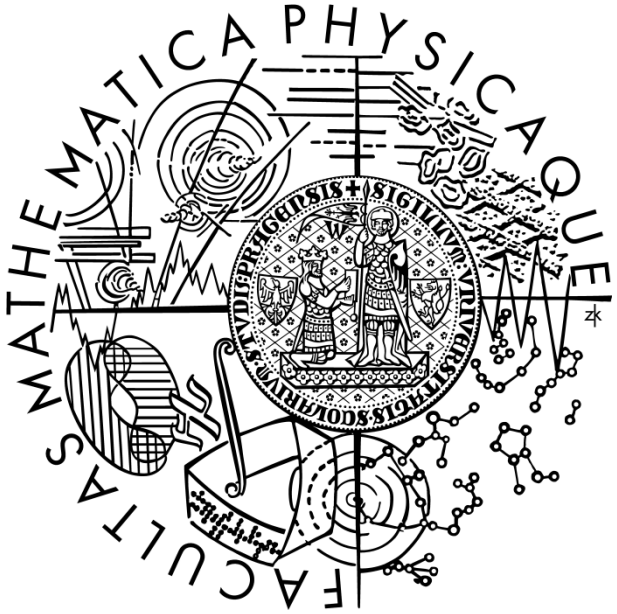
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DISERTAČNÍ PRÁCE



RNDr. Petra Surynková

Analýza bodových množin reprezentujících povrchy technické praxe

Katedra didaktiky matematiky

Vedoucí disertační práce: **Mgr. Šárka Voráčová, Ph.D.**

Studijní program: **Matematika**

Studijní obor: **Obecné otázky matematiky a informatiky**

PRAHA 2013

Děkuji všem, kteří mne při sepisování mé disertační práce podpořili. Děkuji své vedoucí Mgr. Šárce Voráčové, Ph.D. za odborné vedení a spolupráci. Mé zvláštní poděkování patří RNDr. Josefu Pelikánovi z Kabinetu software a výuky informatiky na MFF UK za pomoc při sepisování práce, množství cenných rad a velkou podporu.

Dále bych ráda poděkovala RNDr. Janě Velemínské, Ph.D. z Katedry antropologie a genetiky člověka na Přírodovědecké fakultě UK v Praze za laskavé zapůjčení skenovacích zařízení a RNDr. Václavu Krajíčkovi z Kabinetu software a výuky informatiky na MFF UK za pomoc při 3D skenování.

Prohlašuji, že jsem tuto disertační práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 26. 7. 2013 Petra Surynková

Obsah

Abstrakt **viii**

1 Úvod **1**

1.1 Problém rekonstrukce povrchů v současnosti 1
1.2 Motivace a praktické aplikace 2
1.3 Cíle disertační práce a metodika 2
1.4 Přínos práce k oboru 3
1.5 Použitá experimentální data 5
1.6 Rozvržení práce 5

2 Rekonstrukce povrchů z mračna bodů **7**

2.1 Problém rekonstrukce povrchů 7
2.2 Získávání bodů 3D skenováním reálných povrchů 12
2.3 Registrace 21
2.4 Odstranění nadbytečných bodů a šumů 24
2.5 Aproximace mračna bodů pomocí polygonální sítě 27
2.5.1 Polygonální reprezentace 3D objektů 27
2.5.2 Existující algoritmy pro hledání polygonální sítě 30
2.6 Segmentace 32
2.6.1 Odhad křivostí plochy 33
2.6.2 Segmentační algoritmy 35
2.7 Prokládání povrchu plochou 37
2.7.1 Proložení množiny bodů algebraickou plochou 38

3 Analýza bodové množiny – aproximační metody **41**

3.1 Metody pro zjišťování orientace dat 42
3.2 Aproximace funkce metodou nejmenších čtverců 42
3.3 Ortogonální prokládání dat přímkou 54
3.3.1 Případová studie v eukleidovské rovině E_2 70
3.3.2 Případová studie v eukleidovském prostoru E_3 76
3.3.3 Hledání osových symetrií ortogonálním prokládáním dat přímkou 83
3.4 Aproximace funkce dvou proměnných metodou nejmenších čtverců 89

3.5 Ortogonální prokládání dat rovinou 98
3.6 Analýza hlavních komponent (Principal Component Analysis) 107

4 Analýza bodové množiny – iterační metody 119

4.1 Diferenciální numerické metody 120
4.1.1 Gradientní metody hledání extrému 120
4.2 Hledání osy rotační válcové plochy 127
4.2.1 Případová studie v eukleidovském prostoru E_3 145
4.2.2 Případová studie v eukleidovské rovině E_2 152
4.3 Hledání osy obecné rotační plochy 155
4.3.1 Případová studie v eukleidovském prostoru E_3 167
4.4 Hledání rovinových symetrií 180
4.4.1 Rovinová symetrie s jednou rovinou 181
4.4.2 Rovinové symetrie se dvěma navzájem kolmými rovinami 191
4.4.3 Rovinové symetrie se třemi navzájem kolmými rovinami 204

5 Povrchové analýzy 214

5.1 Odhad tečných rovin a normál bodové množiny 215
5.2 Inkrementální konstrukce polygonální sítě 227
5.2.1 Ošetření speciálních případů 231
5.3 Triangulace povrchu pomocí transformace 243
5.4 Konstrukce povrchové triangulace pomocí rovinových symetrií 255

6 Experimenty s reálnými daty 263

6.1 Reálná data reprezentující povrchy geometrických modelů 264
6.1.1 Model balustrády – optický systém Vectra3D 265
6.1.2 Model balustrády – dotykový skener MicroScribe G2X 270
6.1.3 Model balustrády – laserový skener Roland Picza LPX-1200 275
6.1.4 Model křížové klenby – optický systém Vectra3D 280
6.2 Reálná data reprezentující povrchy skutečných staveb 283

7 Závěr a budoucí práce 286

Literatura 289

A Obsah přiloženého výměnného média 298

Seznam algoritmů

4.1 Výpočet chybové funkce *error_function* pro rotační válcovou plochu 128
4.2 Hledání osy rotační válcové plochy – varianta A 129
4.3 Newtonova metoda tečen 131
4.4 Hledání osy rotační válcové plochy – varianta B 133
4.5 Výpočet chybové funkce *error_function* pro obecnou rotační plochu 157
4.6 Výpočet hodnoty chybové funkce *error_function* pro obecnou rotační plochu v bodě λ 160
4.7 Výpočet chybové funkce *error_function_one_plane* pro rovinovou symetrii 184
4.8 Hledání roviny symetrie bodové množiny 186
4.9 Výpočet chybové funkce *error_function_two_planes* pro rovinové symetrie se dvěma navzájem kolmými rovinami 195
4.10 Hledání dvou navzájem kolmých rovin symetrií bodové množiny 196
4.11 Výpočet chybové funkce *error_function_three_planes* pro rovinové symetrie se třemi navzájem kolmými rovinami 207

Seznam příkladů

2.1 Některé typy 3D skenerů 17
2.2 Příklady některých typů 2-manifoldu a nonmanifoldu 28
3.1 Prokládání dat přímkou metodou nejmenších čtverců 51
3.2 Ortogonální prokládání dat přímkou v rovině a v prostoru 64
3.3 Prokládání dat polynomem dvou proměnných metodou nejmenších čtverců 94
3.4 Ortogonální prokládání dat rovinou 104
3.5 Analýza hlavních komponent 113
4.1 Minimalizace reálné funkce dvou reálných proměnných 123
4.2 Hledání osy rotační válcové plochy 135
4.3 Hledání osy symetrie rovnoběžných přímek 142
4.4 Hledání osy obecné rotační plochy 162
4.5 Hledání roviny symetrie bodové množiny 187
4.6 Hledání dvou navzájem kolmých rovin symetrií bodové množiny 200
4.7 Hledání tří navzájem kolmých rovin symetrií bodové množiny 209
5.1 Odhady tečných rovin a normálových vektorů bodové množiny 218
5.2 Inkrementální konstrukce polygonální sítě 234
5.3 Triangulace povrchu pomocí transformace 247
5.4 Triangulace povrchu pomocí rovinových symetrií 258

Seznam tabulek

3.1 Experimentální data pro aproximaci přímkou metodou nejmenších čtverců 52
3.2 Experimentální data v rovině pro ortogonální prokládaní přímkou 65
3.3 Experimentální data v prostoru pro ortogonální prokládaní přímkou 67
3.4 Experimentální data pro aproximaci rovinou metodou nejmenších čtverců 95
3.5 Experimentální data v prostoru pro ortogonální prokládání dat rovinou 105
5.1 Případy při přidání trojúhelníka do povrchové triangulace 232
7.1 Přehled přiložených programů naimplementovaných ve výpočetním prostředí MATLAB 288
A.1 Obsah přiloženého výměnného média 299

Abstrakt

Název práce:	Analýza bodových množin reprezentujících povrchy technické praxe
Autor:	Petra Surynková
Katedra (ústav):	Katedra didaktiky matematiky
Vedoucí disertační práce:	Mgr. Šárka Voráčová, Ph.D., Fakulta dopravní, České vysoké učení technické v Praze

Abstrakt: Disertační práce *Analýza bodových množin reprezentujících povrchy technické praxe* se zabývá rozvojem a aplikacemi metod digitální rekonstrukce povrchů technické a stavební praxe z mračen bodů. Hlavním výsledkem práce je předložení nových postupů a metod přispívajících k jednotlivým fázím rekonstrukčního procesu vstupních množin bodů. Práce je zaměřena především na analýzu vstupních mračen bodů popisujících speciální typy povrchů. Prezentováno je několik zcela nových algoritmů a vylepšení algoritmů existujících, které řeší dílčí kroky rekonstrukce povrchů. Nové řešící postupy přitom vycházejí více z geometrických vlastností rekonstruovaného objektu. Významným výsledkem disertační práce je rovněž rozbor a zpracování nejen syntetických bodových množin ale především reálných bodových množin, které byly získané vlastnoručním měřením. Podstatným přínosem jsou implementace navržených algoritmů v moderním programovacím jazyku a interaktivním prostředí MATLAB. Aby bylo umožněno reprodukování výsledků, jsou veškerá použitá data a vlastní programy k dispozici na přiloženém výměnném médiu.

Klíčová slova: mračno bodů, rekonstrukce ploch, aproximace, ortogonální prokládání dat, 3D skenování

Title:	Analysis of Point Clouds Representing Surfaces of Engineering Practice
Author:	Petra Surynková
Department:	Department of Mathematics Education
Supervisor:	Mgr. Šárka Voráčová, Ph.D., Faculty of Transportation Sciences, Czech Technical University in Prague

Abstract: The doctoral dissertation *Analysis of Point Clouds Representing Surfaces of Engineering Practice* addresses the development and application of methods of digital reconstruction of surfaces of engineering and construction practice from point clouds. The main outcome of the dissertation is a presentation of new procedures and methods that contribute to each of the stages of the reconstruction process from the input point clouds. The work is mainly focused on the analysis of input clouds that describe special types of surfaces. Several completely new algorithms and improvements of existing algorithms that contribute to individual steps of surface reconstruction are presented. New procedures are based on geometrical characteristics of the reconstructed object. An important result of the dissertation is an analysis of not only synthetically generated point clouds but above all an analysis of real point clouds that have been obtained from measurements of real objects. The significant contribution of the dissertation is also an implementation of all the proposed algorithms in a modern programming language and interactive environment MATLAB. All the data and programs are available on the attached removable media to allow reproducibility of presented results.

Keywords: point clouds, reconstruction of surfaces, approximation, orthogonal data fitting, 3D scanning

Kapitola 1

Úvod

Disertační práce *Analýza bodových množin reprezentujících povrchy technické praxe* představuje prezentaci výsledků výzkumu dosažených během mého doktorského studia na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze. Tématem práce je rozvoj a aplikace metod digitální rekonstrukce povrchů ze vstupní množiny bodů (*mračno bodů*) v oblasti rekonstrukce ploch technické a stavební praxe s cílem analyzovat a dále zpracovávat reálné množiny bodů. Hlavním přínosem práce je předložení nových výsledků a metod založených na geometrických a dalších deskripčních vlastnostech vstupních množin bodů, které reprezentují speciální typy reálných povrchů.

1.1 Problém rekonstrukce povrchů v současnosti

Úloha rekonstrukce povrchů, k jejímuž řešení jsem ve svém výzkumu přispěla, je zadaná pomocí neorganizované konečné množiny bodů (*mračno bodů*) v prostoru na vstupu. Úkolem je přitom vytvořit rekonstruovaný povrch plochy takový, že body vstupní množiny leží na nebo blízko povrchu. Vždy se předpokládá, že vstupní množina bodů odpovídá v prostoru reálné ploše, avšak žádné další informace o jejich struktuře k dispozici v obecné úloze nejsou. Požadavkem je získat výslednou rekonstruovanou plochu v co nejkompaktnějším tvaru, tj. vyjádřenou pokud možno parametricky nebo implicitně. Určení procesu, který vede k získání rekonstruované plochy v kompaktním popisu z neorganizované množiny bodů v prostoru, je předmětem moderní a stále aktuální oblasti výzkumu rekonstrukce povrchů z mračna bodů.

Hledání výsledného zrekonstruovaného povrchu představuje netriviální problém, neexistuje všeobecný řešící postup, který by byl použitelný pro libovolné zadání této úlohy a libovolné mračno bodů. Díky své obtížnosti rekonstrukce povrchů z mračna bodů stále vyžaduje hledání nových způsobů řešení a další inovace. Důležitou motivací pro hledání a rozvíjejí nových řešících technik je právě široké praktické využití této metody. Nejedná se tedy v žádném případě pouze o teoretický problém, nýbrž o problém, který vědecká i praktická komunita potřebuje efektivně řešit. Náročnost obecného problému rekonstrukce povrchů z mračna bodů představuje velkou výzvu k hledání nových univerzálnějších postupů.

Současné přístupy jsou založené na principu rozdělení procesu rekonstrukce do několika na sebe navazujících fází, kde každá fáze úlohu nějakým způsobem zjednodušuje a předzpracovává pro fázi následující. Tento přístup jsem následovala i ve svém výzkumu

a mé výsledky tak přispívají k jednotlivým fázím v současnosti používaném rekonstrukčním procesu.

1.2 Motivace a praktické aplikace

Rekonstrukce povrchů z mračna bodů představuje široce používanou metodu v digitalizaci reálných povrchů. Tato zajímavá problematika se rozvíjí v mnoha odvětvích od počítačové grafiky přes počítačové vidění po architekturu a historické obory, a její aplikovatelnost se neustále rozrůstá.

Velkou inspirací k sepisování disertační práce zabývající se analýzou speciálních povrchů stavební a technické praxe byl také můj zájem o architekturu. Klenby, klenební oblouky, různé formy zastřešení nebo i estetické architektonické prvky jsou hojně využívaným vyjadřovacím prostředkem architektury už od jejích počátků. Lze říci, že architektonická interpretace historie má v mnoha směrech silnou vypovídací hodnotu. Díky technickému pokroku ve stavebnictví a novým materiálům lze realizovat stále novější a rafinovanější stavební plochy. Velké množství ploch využívaných v architektuře je zajímavé také z matematického hlediska. Matematický popis a digitalizace stavebních ploch má svůj smysl jak ve fázi navrhování nových staveb, tak i při popisu a uchování existujícího kulturního dědictví.

Můj zájem o plochy využívané ve stavební praxi se objevil již v rámci mé diplomové práce (Surynková, 2008a), ve které jsem se věnovala matematickému popisu ploch využívaných v architektuře. Plochy byly popisovány prostředky diferenciální a deskriptivní geometrie. Některé typy ploch, jejichž vlastnosti jsou detailně rozebrány v diplomové práci, se objevují i v disertační práci. V tomto smyslu má disertační práce návaznost na moji předchozí práci.

Metody rekonstrukce povrchů nachází uplatnění i mimo architektonické obory. Uveďme například mapování prostředí při navigaci mobilního robota, přenos reálných interiérů a exteriérů do virtuálních světů počítačových her nebo replikaci tvarů skutečných předmětů pomocí 3D tisku.

1.3 Cíle disertační práce a metodika

Předložená disertační práce si kladla za cíl hlavně analýzu vstupního mračna bodů popisujícího nějaký objekt. V širším pohledu ale bylo rovněž cílem zkoumat jednotlivé fáze rekonstrukce povrchů z bodových množin a obecně problém digitální rekonstrukce libovolného reálného objektu. Z toho důvodu jsou v práci představeny již existující metody a postupy rekonstrukce povrchů z mračna bodů využitelné pro různé vstupní množiny. Některé z těchto postupů jsou přímo vhodné pro využití při digitalizaci povrchů technické

praxe. Přehled existujících metod a algoritmů je uveden také pro porovnání existujících a nově navržených postupů.

Hlavní část disertační práce (kapitoly *Analýza bodové množiny – aproximační a iterační metody, Povrchové analýzy a Experimenty s reálnými daty*) se zabývá hledáním nových přístupů a metod hodících se pro dílčí fáze rekonstrukce právě speciálních typů povrchů. Některé nové přístupy jsou založené kromě jiného též na vizuálním posouzení rekonstruované množiny bodů. Některé metody dovolují rovněž interaktivní zásah uživatele, což představuje v řešení rekonstrukce povrchu nové možnosti.

Veškeré navržené algoritmy jsou za účelem verifikace implementovány v moderním programovacím jazyku a interaktivním prostředí MATLAB[1] uzpůsobeném technickým výpočtům. Při testování správnosti postupů jsou využívána počítačově generovaná data nebo data reprezentující menší modely reálných objektů. Ověřování teoretických návrhů pomocí experimentálních implementací je standardní a velmi moderní přístup. Výběr metod, které jsou nejvhodnější pro danou množinu bodů, je prováděn jejich porovnáváním, přičemž se zohledňuje rychlost, spolehlivost a efektivita algoritmu. Navržené metody a známé postupy porovnáváme z hlediska rychlosti a také estetické kvality řešení. Tento aspekt je v problematice hledání rekonstrukce povrchů velmi důležitý.

Jako cíl jsem měla na zřeteli rovněž didaktický rozměr práce, který je naplněn zdůrazňováním geometrických principů, na nichž jsou existující a nově navržené metody založeny, a právě také využitím zmíněných programových implementací v prostředí MATLAB, které umožňují metody snadno vizuálně prezentovat. Rovněž jsou v práci vysvětleny všechny potřebné definice, věty a pojmy, není proto nutná předchozí znalost užívaných termínů a všeobecně dané problematiky.

V některých fázích hledání řešení rekonstrukce povrchu je v hojné míře užíváno též 3D modelování v programu Rhinoceros[2] 4.0 (NURBS modeling for Windows). Tento moderní software je využíván například k předzpracování či dodatečné úpravě zadaného mračna bodů reprezentující rekonstruovaný povrch nebo k tvorbě ilustrací doprovázející text práce.

1.4 Přínos práce k oboru

Přínos disertační práce tkví v rozvoji nových konceptů a přístupů v řešení dílčích fází rekonstrukce povrchů z mračna bodů. V práci se zaměřuji na výzkum této problematiky v oblasti rekonstrukce povrchů technické praxe. Věnuji se konkrétně zpracování bodové množiny nejdříve známými aproximačními technikami, poté navrhuji nové iterační metody založené na diferenciálních numerických metodách. Povrchové reprezentace zkoumaných

[1] Název MATLAB je ochranná známka nebo registrovaná ochranná známka svých vlastníků.
[2] Název Rhinoceros je ochranná známka nebo registrovaná ochranná známka svých vlastníků.

objektů jsou prováděny několika možnými konstrukcemi trojúhelníkové sítě, které jsou v základu založeny na existujících metodách, ty ovšem dále vylepšuji.

Navrhuji celou řadu nových algoritmů (*hledání osy rotační válcové plochy, hledání osy obecné rotační plochy, hledání rovinových symetrií, konstrukce povrchové triangulace pomocí rovinových symetrií*) založených na geometrických principech a vylepšuji nebo doplňuji existující řešící metody (*odhady tečných rovin a normál bodové množiny, inkrementální konstrukce polygonální sítě, triangulace povrchu pomocí transformace*).

Ke každému zpracovávanému tématu přikládám velké množství příkladů, na kterých demonstruji správnost postupů. Nově navržené postupy a algoritmy a modifikované existující metody řešení jsou aplikovány na konkrétní mračna bodů reprezentující povrch. Jedná se jak o počítačově generovaná data, modely reálných objektů, tak i o části reálných staveb. Velkým přínosem práce jsou experimenty s reálnými daty prezentované v závěrečné kapitole. Veškerá reálná data, která jsou také k dispozici na přiloženém výměnném médiu, jsem získala vlastním měřením reálných povrchů profesionálními měřícími systémy.

Všechny navrhované algoritmy jsou implementovány ve výpočetním prostředí MATLAB a programy jsou rovněž součástí práce.

Kromě jiného práce též ukazuje uplatnění znalostí klasické a aplikované geometrie v praktické oblasti počítačové grafiky. Výsledné metody a implementace tedy mohou napomoci ke zvýšení zájmu o studium matematiky a převážně geometrie na vysokých školách. Vizualizace výsledků může sloužit ke zlepšení výuky geometrie. Výstupem práce je také několik programů, které mohou být rovněž využívány ve výuce nebo k dalšímu výzkumu.

Přínos této práce spatřuji také v tom, že dosud neexistuje rozsáhlejší česká publikace věnující se této problematice.

Výsledky prezentované v této disertační práci byly publikovány na těchto konferencích: konference o geometrii a počítačové grafice (GCG), Annual Conference of Doctoral Students – Week of Doctoral Students (WDS), mezinárodní konference Historie matematiky, International Conference on Technology in Mathematics Teaching (ICTMT) a International Conference on Communication, Media, Technology and Design (ICCMTD), viz (Surynková, 2008b, 2009a, 2009b, 2010a, 2011b, 2012a, 2012b; Surynková a Šír, 2010, Surynková *a kol.*, 2009; Surynková a Voráčová, 2011; Voráčová a Surynková, 2011). Článek o využití rekonstrukce v analýze statistických dat a aproximace křivek a ploch metodou postupné evoluce vyšel v časopise South Bohemia Mathematical Letters, (Surynková a Šír, 2011). Plochy stavební praxe a výsledky počítačového modelování byly publikovány v knize Atlas geometrie. Nové metody používané pro rekonstrukci ploch z mračen bodů byly též prezentovány na semináři Speciální seminář z počítačové grafiky na Matematicko-fyzikální fakultě Univerzity Karlovy v Praze.

Další oblasti výzkumu a využití metod převážně v oblasti didaktiky byly prezentovány na konferencích: II. North American GeoGebra Conference (GeoGebra-NA), International GeoGebra Institue Conference, mezinárodní konference Historie matematiky, konference Matematika a reálný svět, konference Jak připravit učitele matematiky (Surynková, 2010b,

2010c, 2011a; Surynková *a kol.*, 2012), a přednáškách a seminářích: Didakticko-historický seminář na Přírodovědecké fakultě Masarykovy Univerzity v Brně, IX. a X. seminář z historie matematiky pro vyučující na středních školách a seminář Aplikace matematiky pro učitele. V průběhu studia byly též veškeré výsledky přednášeny na pravidelných seminářích pro doktorandy na Katedře didaktiky matematiky.

1.5 Použitá experimentální data

V disertační práci používám několik typů vstupních dat. Správnost navržených algoritmů ověřuji na počítačově generovaných datech. Bodové množiny vytvářím buď v 3D modelovacím softwaru Rhinoceros nebo je generuji z výpočetního prostředí MATLAB. Analýza algoritmů na syntetických datech má opodstatnění v tom, že je předem známo, jak má rekonstruovaná plocha vypadat. Dále využití počítačově generovaných bodových množin umožnilo snadnější prototypování, neboť reálné množiny často obsahují velké množství bodů (statisíce až miliony) a jejich zpracování je tedy mnohem obtížnější a časově velmi náročné na dostupné běžné výpočetní technice.

V disertační práci se nejvíce věnujeme bodovým množinám, které vznikly naskenováním povrchů geometrických modelů. Rozměry těchto modelů se pohybovaly v rozmezí od 10 cm do 50 cm. Takto se mohlo pracovat se skutečně reálnými daty získanými skenováním, a přesto jejich velikost dovolovala zpracovat je v rozumném čase. Naskenování modelů nám umožnila Přírodovědecká fakulta Univerzity Karlovy v Praze.

Třetí typ použitých dat bylo několik bodových množin získaných skenováním reálných povrchů kleneb, tedy množin obsahující řádově milióny bodů. Množiny bodů naskenovaných reálných povrchů jsem získala od Stavební fakulty Českého vysokého učení technického v Praze.

1.6 Rozvržení práce

Pro snadnější zorientování čtenáře v textu a přiložených rozšiřujících materiálech uvádím rozvržení disertační práce, které je následující, se stručným popisem obsahu jednotlivých kapitol.

Kapitola 2: Rekonstrukce povrchů z mračna bodů. Tato kapitola je věnována obecnému popisu rekonstrukce povrchů z mračna bodů. Jsou zde představeny hlavní kroky nezbytné pro hledání řešení a uvedeny existující řešící postupy. Vysvětleny jsou všechny pojmy, které jsou potřebné pro následnou práci. Podrobně jsou popsány jednotlivé fáze rekonstrukce a připojen je též přehled existujících algoritmů především pro hledání prvotní polygonální reprezentace.

Kapitola 3: Analýza bodové množiny – aproximační metody. V této kapitole jsou popsány různé způsoby zpracování vstupní bodové množiny aproximačními technikami. Kapitola představuje přehled známých metod převážně z oblastí numerické matematiky a matematické statistiky. I když se jedná o existující postupy, podrobné odvození metody ortogonálního prokládání dat v rovině a v prostoru přímkou a analogická úloha ortogonálního prokládání dat v prostoru rovinou dosud zcela chybělo v české literatuře.

Kapitola 4: Analýza bodové množiny – iterační metody. K hledání osy rotační válcové plochy, osy obecné rotační plochy a rovinových symetrií bodových množin navrhujeme nové iterační algoritmy založené na diferenciálních numerických metodách. Vlastní algoritmy zapisujeme pomocí symbolického kódu a předkládáme celou řadu příkladů s názornými ilustracemi. Naimplementované programy a počítačově generované bodové množiny, na nichž algoritmy testujeme, jsou rovněž součástí disertační práce.

Kapitola 5: Povrchové analýzy. Vymezení hranice rekonstruovaného objektu je předmětem této kapitoly. Jednak je kapitola věnována odhadům tečných rovin a normál bodového mračna v jednotlivých bodech, a jednak různým konstrukcím povrchové triangulace. Vycházíme ze známých postupů, které dále přizpůsobujeme bodovým množinám, jež zpracováváme, a zavádíme nová vylepšení těchto technik. Rovněž navrhujeme vlastní metodu konstrukce povrchové triangulace pomocí rovinových symetrií.

Kapitola 6: Experimenty s reálnými daty. Veškeré nově navržené a popsané řešící postupy testujeme na bodových množinách reprezentující skutečné povrchy. Bodová mračna jsme získali skenováním geometrických modelů třemi různými profesionálními skenovacími zařízeními. Rovněž uvádíme příklad zpracování bodové množiny popisující reálnou stavbu, která obsahuje velké množství bodů.

Kapitola 7: Závěr a budoucí práce. Závěrečná kapitola je věnována shrnutí přínosu disertační práce a diskuzi o dalším možném pokračování ve výzkumu v této oblasti.

Příloha A: Obsah přiloženého výměnného média. Přehled obsahu přiloženého DVD je uveden v této příloze. (Navržené algoritmy implementované ve výpočetním prostředí MATLAB, bodová mračna, text práce ...)

Kapitola 2

Rekonstrukce povrchů z mračna bodů

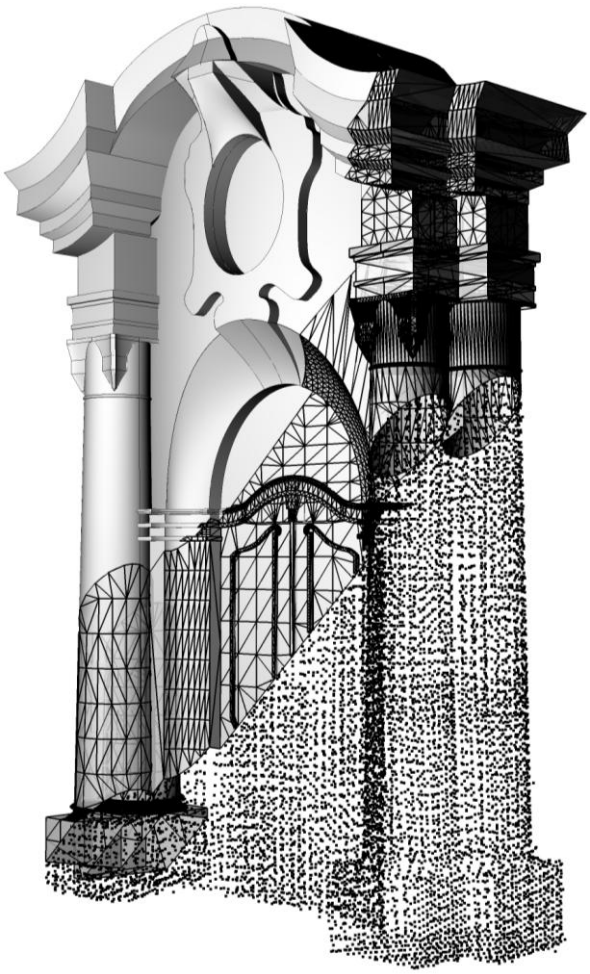
V této kapitole stručně popíšeme principy rekonstrukce povrchů z mračna bodů a představíme hlavní kroky, které jsou nezbytné k získání výsledné rekonstruované plochy. Podáme přehled existujících metod a postupů, čímž zmapujeme současný stav v otázce rekonstrukce povrchů.

2.1 Problém rekonstrukce povrchů

Rekonstrukce reálných geometrických povrchů z neorganizované konečné množiny bodů v prostoru je v současné době velmi aktuální téma. Rekonstrukce povrchů se vyskytuje v řadě odvětví vědy a průmyslu a je nedílnou součástí mnoha vědeckých a technických aplikací v praxi. Jde o moderní oblast výzkumu s dopady na různé disciplíny výpočetní geometrie, počítačové grafiky nebo CAD systémů (*Computer Aided Design*).

Hlavním cílem rekonstrukce je digitální dokumentace reálných povrchů. Jde tedy o proces, který podle fyzické předlohy (např. reálné stavby, fyzické makety nějakého objektu, mechanické součástky) vytváří počítačový model (Pottmann *a kol.*, 2007; Ahn, 2004). Počítačový model představuje matematický popis daného objektu, se kterým je možné dále manipulovat a transformovat jej v matematických a modelovacích počítačových programech. S počítačovými modely získanými rekonstrukcí lze také pracovat při statických výpočtech.

V našem případě se speciálně zaměříme na rekonstrukci ploch s cílem vizualizovat reálná data a dokumentovat reálné objekty. Vyjádříme-li úlohu rekonstrukce matema-



Obrázek 2.1: Ilustrace jednotlivých fází rekonstrukce povrchu z mračna bodů – vstupní množina bodů, trojúhelníková síť reprezentující povrch a výsledný CAD model (výsledná plocha je po částech hladká)

ticky, pak vstup představuje konečná množiny bodů v prostoru, která není nijak dále organizovaná, a cílem je vytvořit takovou aproximující plochu, že body vstupní množiny leží na povrchu nebo v jeho blízkosti. Body vstupní množiny jsou zadány pouze svými prostorovými souřadnicemi, v obecné úloze žádné další informace k dispozici nejsou. Předpokládá se, že vstupní množina bodů odpovídá v prostoru reálné ploše, jinak by úloha neměla rozumné řešení. Pokud je to možné, výslednou rekonstruovanou plochu chceme popsat parametricky či implicitně. Současné metody používají jako mezikrok vhodnou aproximující síť, nejčastěji trojúhelníkovou. Finálním krokem procesu rekonstrukce je tvorba 3D modelu rekonstruovaného objektu. Celý proces rekonstrukce povrchu z mračna bodů ilustruje obrázek 2.1.

Motivace pro získávání počítačových popisů reálných objektů nacházíme ve stavebních oborech. Využívají se při restaurování historických objektů, zaměřování skutečného stavu budov nebo při konzervaci současného kulturního dědictví. Pomocí technik rekonstrukce povrchů je možné digitálně uchovávat dokonce i složitá umělecká díla jako jsou například sochy. Podobu původního objektu je možné díky digitálnímu modelu trvale archivovat, podle modelu lze vyrobit repliky objektu či pomocí počítače model dodatečně upravovat a transformovat. Pro ohrožené architektonické památky představuje digitalizace a rekonstrukce povrchů prostředek, jak je zachovat pro příští generace.

O problematice digitalizace reálných objektů, konkrétně o zaznamenávání chrámových komplexů v Kambodži (např. Bayon Temple), existuje rozsáhlá publikace (Ikeuchi a Miyazaki, 2008). V této knize autoři popisují digitalizaci historických objektů od prvotního měření různými skenovacími zařízeními až po finální tvorbu počítačových modelů a virtuální galerie historických památek.

V našem výzkumu se budeme zabývat především rekonstrukcí geometrických modelů a povrchů využívaných v architektuře, přičemž soustředit se budeme na speciální případy, které často představují z geometrického hlediska zajímavé části architektonických staveb. Konkrétně se budeme věnovat geometrii kleneb, různým formám zastřešení a dalším speciálním plochám stavebně-inženýrské praxe.

Rekonstrukce reálných povrchů prováděná z různých vstupních prvků (mračen bodů, řezů, …) a podmínek se používá i v jiných oblastech jako je lékařství, geodézie, kartografie, archeologie atd. Zajímavou činností, kde se uplatňuje rekonstrukce, je také reverzní inženýrství (*reverse engineering*). Jedná se o proces, který umožňuje odhalit princip fungování a konstrukci zkoumaného předmětu například nějaké mechanické součástky. Na základě měření a určité analýzy struktury se podle původního vyšetřovaného předmětu zhotovuje předmět nejen prostorově (volumetricky) ale také funkčně ekvivalentní. Jedná se tedy o postup opačný k výrobě strojní součástky. Reálný objekt se digitalizuje a na základě jeho CAD modelu se zhotoví objekt nový. Digitalizace se provádí pomocí 3D skenerů. Reverzního inženýrství však bývá také zneužíváno ke kopírování cizích technologií.

Digitální dokumentace se rovněž uplatňuje v počítačovém projektování a navrhování architektonických a designových prvků. Při takovém návrhu nové stavby má architekt možnost pracovat například s počítačovou vizualizací svého nápadu nebo jen s ručními náčrtky, to však nemusí být vždy výhodné a dostačující. Nové možnosti nabízí digitální rekonstrukce

fyzického modelu, který představuje projekt nějaké budoucí stavby. Tento přístup je velmi účinným prostředkem, který může designér, architekt nebo jiný projektant využít. S fyzickým modelem, který je pouze zjednodušenou a zmenšenou formou plánované stavby, se snadněji manipuluje, lze ho měnit, opravovat a to vše s menšími náklady. Díky počítačovému modelu je tedy možné návrh levněji a efektivněji zrealizovat. Převedením fyzického modelu do podoby editovatelné počítačem se urychlí následný proces výroby. Podobný přístup využijeme rovněž v našem výzkumu. V našich experimentech začneme s počítačově generovanými daty a poté budeme pracovat s rekonstrukcí geometrických modelů menších rozměrů. Experimenty s počítačově generovanými daty budou sloužit k ověření správnosti navržených metod, neboť zde budeme vědět, jak má vypadat výstup. Následně přejdeme k rekonstrukcím povrchů reálných objektů nejdříve geometrických modelů poté i částí skutečných staveb. Toto pojetí je přirozené, neboť stavební povrchy jsou popisovány velkým množstvím dat, které nelze zpracovat najednou. Správnost a vlastnosti navrhovaných metod budeme ověřovat experimentálními implementacemi.

Digitální dokumentace má tedy široké uplatnění. Ve většině případů, kde se užívá a které jsme jmenovali, se však jedná už o aplikace této metody. V praxi je většinou proces, který vede od reálného objektu k jeho digitalizaci, v podstatě nedůležitý. Zajímá nás až výsledný produkt. Náš výzkum bude ale spočívat právě v objasnění procesu získání počítačové reprezentace objektu. Tento proces se skládá z několika fází, které budeme řešit jednotlivě. Tyto kroky následně popíšeme a představíme existující ale především námi nově navržené metody vedoucí k jejich řešení. Naše postupy budou speciálně navrhovány pro konkrétní typy povrchů objevujících se v architektuře a ve stavební praxi a přispívat budou k vybraným fázím procesu rekonstrukce.

Popišme nyní jednotlivé kroky rekonstrukce povrchů, tedy postup, který je potřebný k získání rekonstruované plochy v kompaktním popisu. Cílem metody rekonstrukce povrchů, kterou budeme využívat k digitalizaci vybraných objektů, je ze vstupní konečné množiny bodů v prostoru tzv. **mračna bodů** (*point clouds*) zrekonstruovat povrch. Podmínkou je, aby výsledná po částech spojitá plocha co nejlépe popisovala a vyjadřovala dané mračno bodů (Pottmann *a kol.*, 2007; Iske, 2004; Ahn, 2004; Dey, 2007), které reprezentuje nějaký reálný povrch. Cílem řešících metod je zrekonstruovaný povrch popsat parametrickými nebo implicitními rovnicemi a pro další práci je vyžadován i CAD model (Ahn, 2004; Jeměljanov, 2004). V obecné úloze známe pro body vstupní množiny pouze jejich prostorové souřadnice. Mračna bodů jsou získávána 3D skenováním reálných objektů, k čemuž slouží speciální 3D laserové, optické či dotykové skenery (Pottmann *a kol.*, 2007; Ahn, 2004). Výsledkem takového skenování je množina více či méně pravidelně nasnímaných bodů fyzického modelu nebo reálné stavby.

Poznamenejme, že se zde zabýváme pouze 3D skenery, které zachycují body na povrchu objektu. Existují i zařízení, která přinášejí informace také o vnitřní stavbě objektu a využívají se například v lékařství či archeologii. V architektuře, kde cílem je zachytit geometrii povrchu daného objektu, nenacházejí tato zařízení příliš velké uplatnění.

Před samotným skenováním povrchu je nutné znát alespoň některé vlastnosti objektu, abychom mohli rozhodnout, která data jsou pro zachycení významná. Jedná se o povrchovou strukturu, vzhled, komplikovanost povrchu nebo naopak jednoduchost či jiné geometrické vlastnosti (Pottmann *a kol.*, 2007). U některých objektů je žádoucí, aby body mračna, které objekt reprezentují, byly v některých částech povrchu hustší nebo řidší, u jiných je naopak výhodou, aby byly rozmístěny pravidelně. Toto rozhodování je nezbytné, neboť vždy pracujeme s konečnou množinou bodů, která má reprezentovat spojitý povrch.

Naměřená data obsahují velké množství bodů (u reálných povrchů až miliony), přičemž v ideálním případě tyto body leží přímo na povrchu objektu. V reálných aplikacích však může velmi často docházet k chybám a nepřesnostem v měření (Iske, 2004; Ahn, 2004; Dey, 2007). Každé skenovací zařízení pracuje s různě velkou odchylkou měření, je tedy nutné s těmito aspekty počítat. Kromě toho povrch reálného objektu většinou nelze zachytit pouze jedním skenováním z jednoho výchozího bodu. Z jedné pozice skeneru může být totiž přímo viditelná pouze část objektu. Tento problém se řeší tzv. **registrací** (Pottmann a *kol.*, 2007). Skenování se provádí vícekrát z různých stanovišť a získaná mračna bodů se následně slučují. Počet těchto dílčích snímání se odvíjí od členitosti objektu, například obsahuje-li díry nebo je jinak komplikovaný, je nutné provádět snímání vícekrát. Každé takto získané mračno bodů je reprezentováno v různých kartézských soustavách souřadnic. Sloučením bodů do jedné kartézské soustavy souřadnic získáváme mračno bodů reprezentující povrch daného objektu.

Výsledné mračno bodů může obsahovat redundantní data (Iske, 2004; Ahn, 2004; Dey, 2007), tedy body, které nepřináší žádnou novou informaci nebo leží navzájem příliš blízko sebe. Kromě toho se díky nepřesnostem v měření mohou ve výsledné množině objevovat tzv. šumy. K odstranění nadbytečných bodů a k filtrování nežádoucích šumů slouží celá řada tzv. **ztenčujících algoritmů** (*thinning algorithms*) (Iske, 2004). Právě popsané postupy tvoří tzv. **bodovou fázi** (*point phase*) procesu rekonstrukce povrchu (Pottmann *a kol.*, 2007).

V tomto okamžiku máme připravenou vstupní množinu bodů (mračno) pro další zpracování. Odstraněny jsou nadbytečné body a chyby v měření. Následuje tzv. **polygonální fáze** (*polygon phase),* tedy postup (Pottmann *a kol.*, 2007), kterým dané mračno bodů aproximujeme pomocí polygonální sítě (Dey, 2007), nejčastěji sítě trojúhelníků. Výsledná síť je skutečně pouze aproximací tvaru původního povrchu, neboť pro přesný popis rekonstruovaného objektu bychom potřebovali, aby vstupní mračno bodů spojitě opisovalo původní povrch, což technicky není možné. Je tedy opravdu nezbytné dodržet předchozí požadavky při skenování, aby výsledná aproximující síť byla korektní a původní povrch aproximovala co nejlépe. Hledání této sítě představuje obtížný problém. Dosud neexistují uspokojivé univerzální řešící postupy. Lze říci, že pro každou vstupní množinu je nutné použít pro počítání aproximující sítě speciální algoritmus. Je zřejmé, že jiný algoritmus bude vyžadovaný pro objekt s komplikovaným povrchem s otvory a jiný pro jednoduchý povrch popisující objekt málo členitý. Díky vytvoření aproximující sítě, dostáváme prvotní hraniční reprezentaci objektu, se kterou lze dále pracovat. Tento krok je nevyhnutelný, neboť pro mnoho aplikací je pouhá bodová reprezentace povrchu nedostačující.

Na tomto místě bychom mohli rekonstrukci povrchu ukončit. Pokud totiž bude aproximující síť reprezentující povrch dostatečně jemná, v mnoha oblastech počítačové grafiky a výpočetní geometrie nebo v různých grafických aplikacích, je tento popis postačující. Pokud se ale věnujeme povrchům stavebních ploch objevujících se v architektuře či technickým povrchům, je další zpracování důležité.

Pro účely stavitelství, architektury a digitální dokumentace památek je proto do rekonstrukce povrchu přidána ještě tzv. **tvarová fáze** (*shape phase*) (Pottmann *a kol.*, 2007). Tento proces převádí síť aproximující povrch do CAD reprezentace (3D počítačový model objektu), která je vhodná pro další zpracování, vizualizaci nebo různé simulace. Jde tedy o nahrazení trojúhelníkové sítě hladkým povrchem, který aproximuje zadané mračno bodů. Součástí této fáze je též tzv. **segmentace** (*segmentation*), tedy detekce a rozdělení povrchu objektu na oblasti s rozdílnou geometrií (Pottmann *a kol.*, 2007; Ahn, 2004). Jinými slovy jde o nalezení základních geometrických prvků (*geometric primitives*), tj. rovinných částí objektu, částí válcových, kuželových, kulových ploch, jiných speciálních ploch nebo částí obecných ploch (*freeform patches*). Při segmentaci tedy rozdělíme vstupní data naskenovaného objektu na části, které následně aproximujeme daným typem plochy. Nepopisujeme tedy povrch pomocí jedné hladké plochy, což v praxi většinou není ani možné, ale soustavou různých ploch, které na sebe napojujeme. Je tedy nutné rozpoznat také hrany objektu, ve kterých jsou jednotlivé části napojovány ostře. Segmentace není plně automatická, obzvláště ne při rekonstrukci objektů z oblastí jako je architektura. Segmentace může být u daného objektu po aplikaci různých postupů rozdílná. Velkou výhodou je samozřejmě znalost geometrických vlastností daného objektu. Tvarovou fázi uzavírá **proces prokládání povrchem** (*surface fitting*) daným typem plochy určeného při segmentaci (Pottmann *a kol.*, 2007; Ahn, 2004). Při prokládání povrchu získáme také implicitní nebo parametrické vyjádření částí ploch, kterými naskenovaný povrch reprezentujeme.

Rekonstrukce povrchů z mračna bodů se tedy skládá z několika dílčích kroků. Shrňme tyto kroky v následujícím přehledu.

Rekonstrukce povrchů je tvořena

- **bodovou fází**
  - získávání mračna bodů 3D skenováním reálných povrchů (viz oddíl 2.2)
  - registrace (viz oddíl 2.3)
  - odstranění nadbytečných bodů a šumů (viz oddíl 2.4)
- **polygonální fází**
  - aproximace mračna bodů pomocí polygonální sítě (nejčastěji trojúhelníkové) (viz oddíl 2.5)
- **tvarovou fází**
  - segmentace (viz oddíl 2.6)

- prokládání povrchu daným typem plochy určeného při segmentaci (viz oddíl 2.7)

V následujícím textu se budeme podrobněji věnovat jednotlivým částem rekonstrukce povrchů. Uvedeme nejzákladnější typy a rozdělení skenovacích zařízení. U jednotlivých fází rekonstrukce povrchů se zaměříme na teorii, která je nezbytná k pochopení dalších postupů a ze které budeme dále vycházet. Popíšeme současné metody, postupy a algoritmy, které se v jednotlivých krocích rekonstrukce používají. Potřebnou teorii doplníme vhodnými ilustracemi. Názorné obrázky budou sloužit k lepšímu pochopení dané problematiky a mohou být též využity didakticky.

2.2 Získávání bodů 3D skenováním reálných povrchů

Podívejme se nyní podrobněji na bodovou fázi rekonstrukce povrchů, tedy na získávání mračna bodů. Technologie 3D skenování představuje nový přístup v pořizování prostorové informace. Bezkontaktní optické či laserové skenování slouží k určování prostorových souřadnic bodů objektu (Varnuška, 2002; Rožánek, 2008). Tyto 3D skenery poskytují data v takové hustotě a přesnosti, že v některých aspektech neexistuje adekvátní srovnání s jinými technikami měření. Množství přenesených dat je většinou velmi velké, proto je nutné mít ke zpracování výkonnou výpočetní techniku. Existují však i další 3D skenovací zařízení, která jsou založená i na jiných principech. Oblasti využití metody digitalizace se neustále rozrůstají. Jak už jsme se zmínili v předchozím, 3D skenování má uplatnění ve stavebních oborech, v lékařství, ve fyzické antropologii, geodézii, kartografii, značnou oblast využití představuje také strojírenská výroba (*reverse engineering*). Stejně jako ve stavebních oborech, je i ve strojírenství 3D skenování využíváno ve fázi navrhování k nasnímání fyzického modelu například nějaké součástky. Neopomenutelnou oblastí využití je vytvoření modelu staré součástky v případě, že je potřeba tuto součástku vyrobit jako náhradní díl. Možnost získat data z hotového dílu, výrobku nebo nějaké formy, počítačově rekonstruovat opotřebenou plochu a provést opravu nejdříve v CAD reprezentaci a teprve poté v reálu je obrovským přínosem. Ve výrobních oblastech hrají 3D skenery nezastupitelnou roli v získávání a archivaci databází výrobků. 3D skenování lze použít také ke zpětnému ověření přesnosti. Pokud se vyrobí nějaká součástka na základě naskenovaných bodů již existující součástky, lze zpětně ověřit, jak moc se nový výrobek liší od původního opět skenováním. Toto ověřování se provádí porovnáním mračen bodů původního a nového výrobku. S nástupem moderních a kvalitních snímacích zařízení ve spojení s CAD systémy je možné měřit a následně vyrábět tvarově i velmi složité součástky. Tyto součástky bychom tradičními ručními metodami měření (posuvná měřidla, šablony, …) nezvládli popsat.

Díky vysokému nárůstu výkonu počítačů lze pracovat s velkým množstvím dat s vysokým rozlišením v reálném čase. Z toho vyplývá také vyšší výsledná přesnost skenování.

Vysoká hustota naměřených bodů a krátká doba měření představují velké výhody optického a laserového skenování.

Hranice velikosti skenovaného objektu nejsou stanoveny technickým omezením. Lze snímat objekty velkých rozměrů (automobily, letadla, budovy), ale díky vysokému rozlišení také předměty velmi malých rozměrů.

Přejděme nyní ke kategorizaci skenovacích zařízení. Nutno podotknout, že se jedná pouze o stručný přehled, pro další informace odkazuji čtenáře na literaturu a další internetové zdroje (Geovap**,** 2011; Simple3D, 2006; Skoupý, 2007; SolidVision, 2011; Urbánek, 2008).

Digitální snímací zařízení lze třídit z různých hledisek. Nejzákladnější dělení je na *kontaktní (dotykové)* a *bezkontaktní (bezdotykové)* přístroje. V prvním případě jde o souřadnicové měřící systémy (*coordinate measuring machine*) a mechanické 3D skenery. Principem těchto zařízení je dotyk se skenovaným povrchem objektu. Digitalizace předmětů velkých rozměrů může být časově velmi náročná. Navíc mají tyto přístroje většinou nižší produktivitu výsledné množiny bodů. Bezkontaktní systémy měření pracují nejčastěji na laserovém nebo optickém principu a získávají data bezkontaktním způsobem, tedy nedotýkají se skenovaného předmětu. Jak už bylo řečeno, tyto přístroje pracují velmi rychle a vytváří hustou síť bodů. Snímání povrchových bodů objektu se tedy vyznačuje značnou produktivitou. Vhodné jsou pro zpracování objektů velkých i malých rozměrů. Do skupiny bezkontaktních měřících zařízení se dále řadí také ultrazvukové a rentgenové skenery.

Další kritéria pro členění se uvádějí na základě toho, zda jde o *stacionární* nebo *mobilní* zařízení. Stacionární zařízení nelze přemisťovat, proto je nutné skenovaný předmět dopravit ke skenovacímu zařízení, z čehož samozřejmě plynou jistá omezení. Mobilními zařízeními lze skenovat i předměty, které není možné přemístit.

3D skenery můžeme také dělit na *destruktivní* a *nedestruktivní*. U destruktivních skenerů dochází při skenování k nevratnému zničení skenovaného objektu. Tímto způsobem lze však naskenovat i dutiny, tvarově složitý povrch a získat tak popis nejen vnějšího povrchu objektu, ale také informace o vnitřní geometrii.

Další možné dělení skenovacích zařízení je podle stupně dosahované přesnosti na zařízení s vysokou nebo běžnou přesností nebo podle použité technologie na *mechanické, laserové, optické, rentgenové* či *ultrazvukové.*

U jednotlivých 3D skenerů jsou důležité další parametry a to měřící objem, přesnost a rychlost. Měřící objem udává, jak velký prostor můžeme při jednom záběru skenerem změřit. Pokud je měřený objekt větší, než je měřící objem daného skeneru, lze měření provádět vícekrát z různých stanovišť a jednotlivá měření poté sloučit. Jedná se o již zmiňovanou registraci. Rozdíl mezi skutečnou a naměřenou hodnotou se označuje jako přesnost. Dalším uváděným parametrem 3D skenerů je také rychlost, která udává počet naskenovaných bodů za sekundu.

Některé typy 3D skenovacích zařízení mohou patřit do více kategorií zároveň. Nás budou zajímat především přístroje, které jsou vhodné pro snímání vnější geometrie objektu. Systémy pro snímání vnitřní stavby daného objektu, jak už jsme zmínili výše, nemají příliš

velké uplatnění v architektuře. Pojďme se tedy seznámit s některými typy skenovacích zařízení podrobněji.

V následujícím přehledu si shrňme základní kategorizaci skenovacích zařízení.

Dělení 3D skenerů

- **podle způsobu snímání**
  - kontaktní
  - bezkontaktní
- **podle přemístitelnosti**
  - stacionární
  - mobilní
- **podle zachování skenovaného objektu**
  - destruktivní
  - nedestruktivní
- **podle dosahované přesnosti**
  - zařízení s vysokou přesností
  - zařízení s běžnou přesností
- **podle principu digitalizace**
  - mechanické
  - optické
  - laserové
  - rentgenové
  - ultrazvukové

Mechanické 3D skenery. Jedním z přístupů, jak nasnímat povrch nějakého objektu, je použití mechanického ramene s několika stupni volnosti (Pottmann *a kol.*, 2007). Princip tohoto zařízení spočívá v tom, že se daného fyzického objektu dotýkáme hrotem, který je zavěšený na pohyblivém rameni s několika klouby. V každé poloze je zaznamenáno natočení ramene a poloha snímaného bodu je vyhodnocena na základě údajů z kloubů ramene. Tento typ skenerů patří tedy mezi dotykové přístroje. Abychom získali co nejpřesnější digitální obraz fyzického modelu, je nutné nasnímat důležité body na objektu, jejichž počet se odvíjí od složitosti povrchu a požadované přesnosti. Tyto body je nutné označit před vlastním snímáním. Výsledné mračno bodů je poměrně řídké a navíc získávání dat tímto způsobem je časově náročné. Dosahovaná přesnost se zpravidla pohybuje v řádu desetin milimetru. Taková přesnost většinou není dostačující pro užití ve strojírenství, avšak plně vyhovuje v architektuře a designu, při tvorbě počítačových her nebo v počítačovém sochařství – CAS (*Computer*

Aided Sculpting). Přímý dotyk měřícího zařízení s měřeným objektem někdy způsobuje jeho poškození, což je například u vzácných historických artefaktů naprosto nežádoucí. Výhodou této metody je však ruční řízení, designér tedy může přímo ovlivnit, která část objektu se má nasnímat. Z toho důvodu je tato metoda často používána pouze ke snímání některých částí. Povrch objektu může být hodně členitý a dokonce může obsahovat díry. Touto metodou se však nedají získat informace o textuře povrchu snímaného objektu.

Existují též strojově polohovatelné souřadnicové měřící systémy, které jsou založeny opět na principu dotyku se skenovaným povrchem. Jedná se o nemobilní zařízení, která většinou disponují velmi přesným měřením.

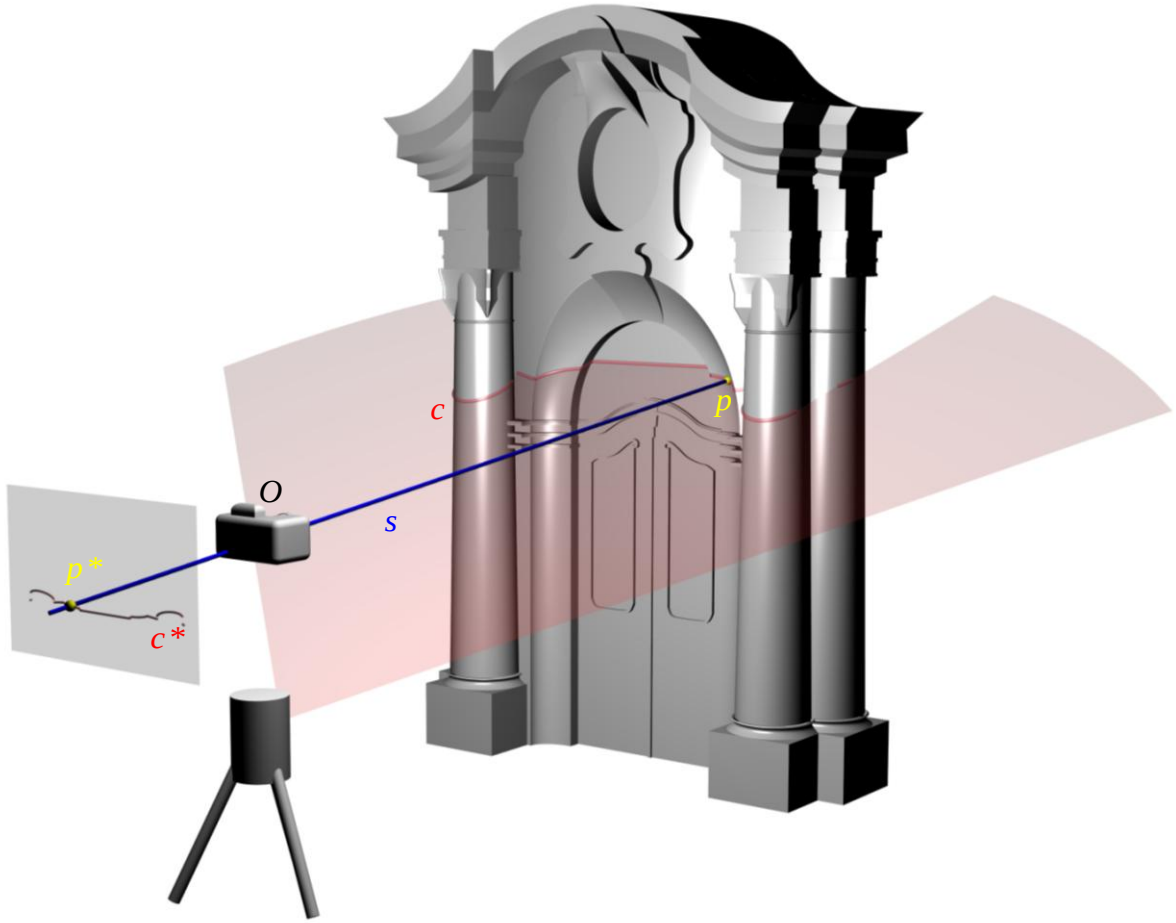
Optické 3D skenery. Moderní 3D skenovací systémy jsou převážně založeny na optických technologiích. Tyto skenery snímají obrysy objektu z několika úhlů pomocí optického zařízení. Při každém natočení se objekt vyfotí a po získání snímků se data následně zpracují v počítači. Otáčení skeneru se provádí ručně nebo pomocí polohovacího zařízení. Snímků musí být dostatečný počet a pořizovány musí být z různých úhlů tak, aby obsahovaly všechny potřebné informace o objektu. Samozřejmě čím bude skenovaný objekt složitější, tím musí být takto pořízených snímků více.

Jednotlivé snímky jsou 2D fotografie, výhodou tedy je, že informace o textuře a barevnosti objektu jsou v datech již obsaženy. Nevýhodou je však tvorba výsledného modelu spojováním jednotlivých snímků, protože touto metodou získáme pouze přibližný povrch objektu. Proto je před samotným skenováním vhodné na objektu vyznačit několik orientačních bodů pro přesnější spojování snímků. Navíc je nutné, aby byly jednotlivé snímky pořízeny proti kontrastnímu pozadí, jinak by objekt na snímku nebyl rozpoznatelný. Můžeme se rovněž setkat s problémem, že je snímaný povrch příliš lesklý, potom je nutné objekt správně nasvítit. Optické skenery většinou nedokážou zachytit větší prohlubně a díry a více členitý povrch. Při vhodné kombinaci s dalšími systémy však mohou digitalizovat veliké objekty.

Laserové 3D skenery. Laserové skenery jsou založeny na bezkontaktním způsobu digitalizace a využívají ke snímání objektů laserového paprsku (Pottmann *a kol.*, 2007). Existuje celá řada typů těchto skenerů, přičemž se dělí na základě užité technologie k určování prostorových souřadnic bodů, vždy je ale využíváno laserového paprsku.

Jedna z možností, jak určit prostorovou polohu bodu objektu, je vyhodnocení doby letu laserového paprsku a jeho odrazu od objektu, typ skenerů s touto technologií je označován jako *lidar* (*light detection and ranging*). Paprsek skener vyšle proti skenovanému předmětu, ten se od něho odrazí a vrátí se zpět do skenovacího zařízení, kde se vyhodnotí. Vzdálenost mezi vysílačem a povrchem objektu je spočítána z času, který uběhne mezi vysláním a přijetím signálu (*time of flight*). Tyto skenery mají velký dosah a používají se k digitalizaci objektů velkých rozměrů např. k tvorbě 3D modelů měst nebo terénu. Prakticky všechny tyto skenery jsou vybaveny též digitální kamerou nebo fotoaparátem pro snímání barvy objektu. Pro jednotlivé body v prostoru je totiž možné vypočítat jejich polohu v digitálním snímku, přenést hodnotu do 3D modelu a doplnit tak chybějící informaci o textuře objektu.

Nejčastěji využívaný typ laserových skenerů pracuje na základě známé polohy skenovacího zařízení a triangulačních metod. Opět je využito laserového paprsku. Skener tedy vysílá záření, které na objektu vytváří bodovou stopu. Pro urychlení procesu měření je místo bodové stopy pomocí vhodného optického prvku tvořícího laserovou rovinu vysílána stopa čárová. Čárová stopa (křivka), je vlastně průnikem laserové roviny a skenovaného objektu, viz obrázek 2.2. Součástí zařízení je i jedna nebo více digitálních kamer (se stejnou základnou jako skenovací zařízení), které detekují stopu laseru na objektu. Ve výsledku tedy pracujeme s obrazem čárové stopy v digitálním snímku a určujeme tak prostorové polohy bodů čárové stopy na objektu. Princip tohoto způsobu skenování je znázorněn na obrázku 2.2.



Obrázek 2.2: Laserový 3D skener kombinovaný s digitální kamerou – princip snímání čárové stopy laserového paprsku na objektu digitální kamerou

Čárová stopa je v obrázku 2.2 označena jako c a její digitální obraz jako $c*$. Každý bod $p*$ křivky $c*$ je spojený se středem promítání O kamery paprskem s, který protíná laserovou rovinu v bodě p křivky c, jehož projekci $p*$ zjišťujeme z digitálního obrazu. Bod p lze jako průsečík paprsku a laserové roviny získat, neboť známe pozici skenovacího zařízení a kamery. Tímto způsobem získáme pro jednu polohu laserové roviny v digitálním obrazu křivku bodů. Dále je laserová rovina posunuta do nové polohy a postup je opakován.

Sjednocením bodů z jednotlivých poloh laserové roviny obdržíme výsledné mračno bodů. (Tyto skenery jsou v některých publikacích označovány též jako triangulační, viz (Pavelka, 2006).) Na rozdíl od klasických optických skenerů jsou laserové skenery velmi přesné a nemají problém ani se členitým povrchem s otvory. Tento typ skenerů má však omezený dosah několika metrů a používá se na objekty do velikosti také jen několika metrů. Problémy se objevují rovněž při skenování příliš tmavých nebo lesklých povrchů.

Opět je digitální kamerou při skenování snímána též barva objektu, čímž je doplněna chybějící informace o textuře.

Laserové skenování představuje neselektivní techniku, tedy nejsou měřeny charakteristické body objektu, ale body jsou rozmístěny pravidelně.

Rentgenové 3D skenery. Pokud je nutné získat informaci o vnitřní stavbě zkoumaného objektu, používají se bezkontaktní rentgenové skenery. Princip těchto zařízení je stejný jako u klasických rentgenů používaných ve zdravotnictví s tím rozdílem, že intenzita záření může být mnohem vyšší. Rentgenové skenery jsou ve většině případů mobilní a používají se například ke kontrole potrubí nebo jiných uzavřených prostor. Tyto skenery se nepoužívají k tvorbě 3D modelů objektů.

Ultrazvukové 3D skenery. Tento způsob digitalizace spočívá na základě bezkontaktního snímání povrchu objektu ultrazvukovou sondou. Sonda vysílá ultrazvukový signál, ten je speciálním zařízením vyhodnocován a získány jsou z něho prostorové souřadnice bodů objektu. Jedná se zřejmě o cenově nejvýhodnější způsob zaznamenávání prostorových souřadnic objektu, ovšem nevýhodou je relativně malá přesnost. V mnoha oblastech je ale i tato digitalizace plně dostačující.

Destruktivní 3D skenery. K popisu jak vnější tak i vnitřní stavbě objektu slouží destruktivní 3D skenery. Při skenování touto metodou dochází ke zničení skenovaného objektu, což je naprosto nežádoucí v oblastech, jako je architektura. Tato metoda je však využívána v reverzním inženýrství při digitalizaci součástek se složitou vnitřní geometrií.

V následujícím příkladě uveďme pro lepší představu nejběžnější zástupce alespoň třech skupin popsaných skenovacích zařízení – mechanické, optické a laserové 3D skenery.

Příklad 2.1: *NĚKTERÉ TYPY 3D SKENERŮ.*

A) Mechanické 3D skenery

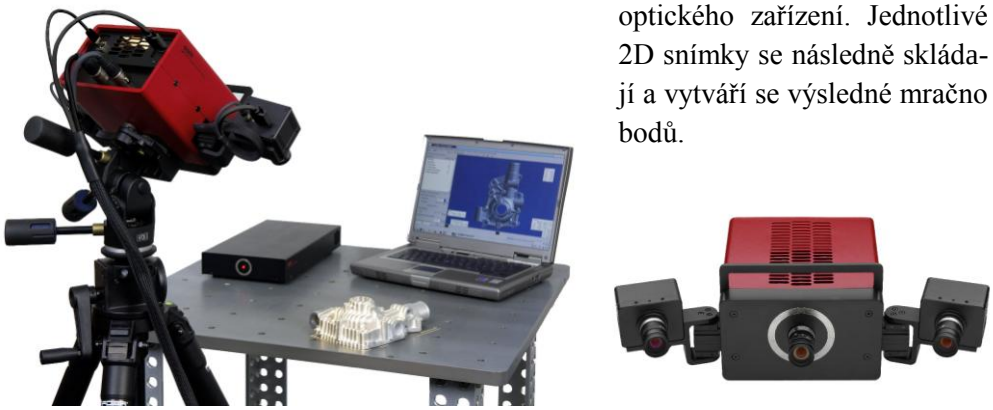
Mechanické 3D skenery pracují na základě dotyku konce mechanického ramene a skenovaného povrchu. Prostorové souřadnice skenovaného bodu se získají vyhodnocením údajů ze všech kloubů ramene.

Obrázek 2.3: 3D mechanický skener MicroScribe
(http://www.3d-microscribe.com)

Na obrázku 2.3 můžeme vidět jeden z mnoha zástupců těchto skenerů mechanický 3D skener *MicroScribe – desktop digitizing system* od americké firmy Immersion. Tento 3D skener je vhodný pro účely počítačové grafiky, herní animace, designérskou a architektonickou tvorbu. Informace o technických parametrech tohoto skeneru je možné nalézt v (Emicroscribe, 2012).

B) Optické 3D skenery

Optické 3D skenery snímají objekt z několika úhlu pomocí optického zařízení. Jednotlivé 2D snímky se následně skládají a vytváří se výsledné mračno bodů.



Obrázek 2.4: 3D optický skener Atos
(http://www.scanare3d.com)

Na obrázku 2.4 je ukázán mobilní bezdotykový optický 3D skener *Atos* od německé firmy GOM. Tento 3D skener pracuje tak, že se na povrch skenovaného objektu promítají pruhy světla, které se na povrchu objektu zdeformují podle jeho tvaru, a pomocí dvou digitálních kamer se povrch objektu snímá. Získají se tak dva obrazy objektu z různých úhlů a dalším zpracováním se vypočte poloha každého obrazového bodu. Objekt se snímá postupně natočením či přesunutím skeneru do další polohy. V nové poloze se musí vždy nasnímat alespoň tři orientační body z předchozího měření, které byly před samotným skenováním vyznačeny. Proces měření je založen na optické triangulaci a fotogrammetrii. Přístroj je výrobcem vybaven softwarem pro automatickou tvorbu výsledného mračna bodů. Tento 3D skener je vhodný například pro reverzní inženýrství, oblasti počítačové grafiky, strojní výrobu. Informace o technických parametrech tohoto skeneru je možné nalézt v (Gom, 2012).

Obrázek 2.5: Sestava 3D optického měřícího systému Tritop (http://www.scanare3d.com)

Na obrázku 2.5 můžeme vidět 3D optický měřící systém *Tritop* od německé firmy GOM. Toto zařízení se skládá z digitálního fotoaparátu s vysokým rozlišením, speciálních optických značek, které se umisťují na skenovaný objekt, kalibrační tyče a vyhodnocovacího softwaru.

Princip získávání prostorových souřadnic bodů na povrchu objektu je založen na metodách fotogrammetrie. Objekt, který je předem označen speciálními optickými značkami, je snímán digitálním fotoaparátem z různých pozic vždy společně s kalibrační tyčí. Je nutné dodržet, aby každá pomocná značka byla obsažena na alespoň třech snímcích. Naměřená data se přenáší k dalšímu zpracování do počítače. Na základě digitálních snímků se spočítá pozice fotoaparátu pro jednotlivé snímky a prostorové souřadnice měřených bodů na povrchu objektu. Prostřednictvím kalibrační tyče se zjišťují správné číselné údaje vzdáleností. Na rozdíl od systému Atos systém Tritop nezkoumá povrch skenovaného objektu, ale pouze souřadnice jednotlivých speciálních optických značek. Oblasti využití tohoto skeneru jsou například reverzní inženýrství nebo deformační analýzy (měří se ve dvou fázích a to nedeformovaný a deformovaný stav objektu). Další informace o technických parametrech tohoto skeneru je možné nalézt v (Gom, 2012).

C) Laserové 3D skenery

Obrázek 2.6: 3D laserový skener HandyScan 3D (http://www.worldnewelectronics.com)

Laserové 3D skenery jsou založeny na bezkontaktním způsobu digitalizace a využívají ke snímání objektů laserového paprsku.

Na obrázku 2.6 můžeme vidět ruční laserový skener *HandyScan 3D* od kanadské firmy Creaform. Toto zařízení pracuje tak, že povrch skenovaného objektu ozáří laserovým křížem, který je snímán pomocí dvou kamer a vyhodnocován triangulačními technikami.

Oblasti využití tohoto skeneru jsou rozsáhlé, jmenujme například průmyslový design, reverzní inženýrství, tvarovou analýzu, architekturu a další umělecké oblasti. Pro další informace odkazuji čtenáře například na (Creafrom, 2012).

Na obrázku 2.7 je ukázán další příklad 3D laserového skeneru *Vivid* od japonského výrobce Konica Minolta. Stejně jako předchozí 3D skener pracuje toto zařízení na principu triangulace. Na povrch skenovaného objektu je vysílán laserový pruh, který je následně snímán pomocí zabudované kamery. Tento typ skeneru je opět využitelný například v reverzním inženýrství nebo v oblastech počítačové grafiky. Uplatňuje se ale také v architektuře, při archivaci historických předmětů, v průmyslovém designu, při vývoji počítačových her a v mnoha dalších odvětvích. Další technické parametry může čtenář nalézt na internetových stránkách výrobce (Konica Minolta, 2012).

Obrázek 2.7: 3D laserový skener Vivid
(http://www.ems-usa.com)

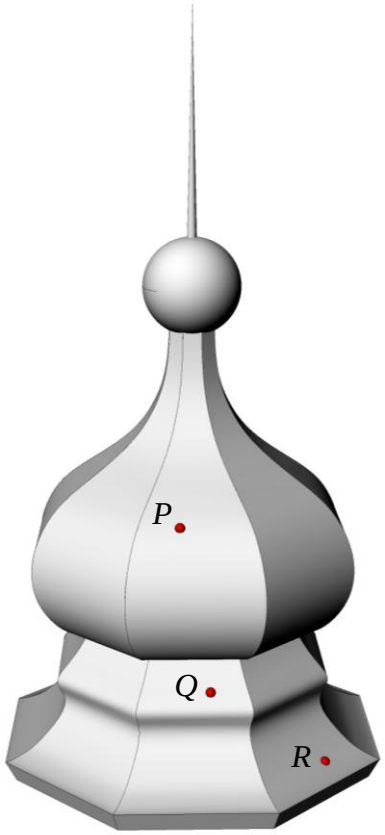
Poznamenejme, že jednotlivé typy skenerů jsme zdaleka nevyčerpali. Existují další skenovací zařízení, která jsou založena na zcela jiných principech. Výběr skenovacího zařízení se samozřejmě odvíjí od aplikace, ve které bude použito získané mračno bodů. Navíc je nutné podotknout, že přesný popis principů jednotlivých skenovací zařízení není cílem této práce. Uvádíme tedy vždy jen stručnou charakteristiku daného skenovacího zařízení bez větších detailů. Podrobné technické informace o 3D skenerech může čtenář nalézt v mnoha internetových zdrojích přímo u daného výrobce.

Díky velkému technickému vývoji v posledních letech byly postupy v digitálním měření zjednodušeny a urychleny a možnosti práce se získanými daty podstatně rozšířeny. Lze očekávat, že digitální dokumentace bude zasahovat do stále více oblastí.

Typy skenovacích zařízení a jednotlivé přístroje, které jsme využili k našim experimentům, popíšeme až u konkrétních příkladů mračen bodů.

2.3 Registrace

V tomto oddíle podrobněji popíšeme proces registrace, který je nutný k získání výsledného mračna bodů. Jak už bylo řečeno v předchozím, povrch objektu (fyzický model nějaké stavby nebo součástky, skutečná stavba, …) většinou nelze zachytit pouze jedním skenováním z jednoho výchozího bodu. Je zřejmé, že z jedné pozice skeneru je přímo viditelná pouze část daného objektu (pokud tedy nejde o nějaký triviální objekt nebo speciální polohu). Z jednoho měření tedy dostáváme body pouze nějaké oblasti povrchu, která je z dané pozice skeneru viditelná. Z toho důvodu se skenování provádí několikrát z různých stanovišť. Jde-li však o mobilní objekt, můžeme měnit také jeho polohy a skenovací zařízení naopak ponechat na místě. Počet těchto jednotlivých skenování se u různých objektů liší a závisí na členitosti objektu a jeho komplikovanosti. Evidentně bude zapotřebí více snímání pro členitý povrch, který obsahuje otvory nebo různé prohlubně. Z dílčích snímání reálného povrchu získáme samostatné množiny bodů (mračna bodů) reprezentované v různých kartézských soustavách souřadnic. Pro další postup je nutné tyto bodové množiny sloučit do jedné a získat tak výsledné mračno bodů, které již bude popsáno v jedné kartézské soustavě souřadnic. Proces slučování jednotlivých mračen bodů se nazývá *registrace* (Pottmann a *kol.*, 2007).



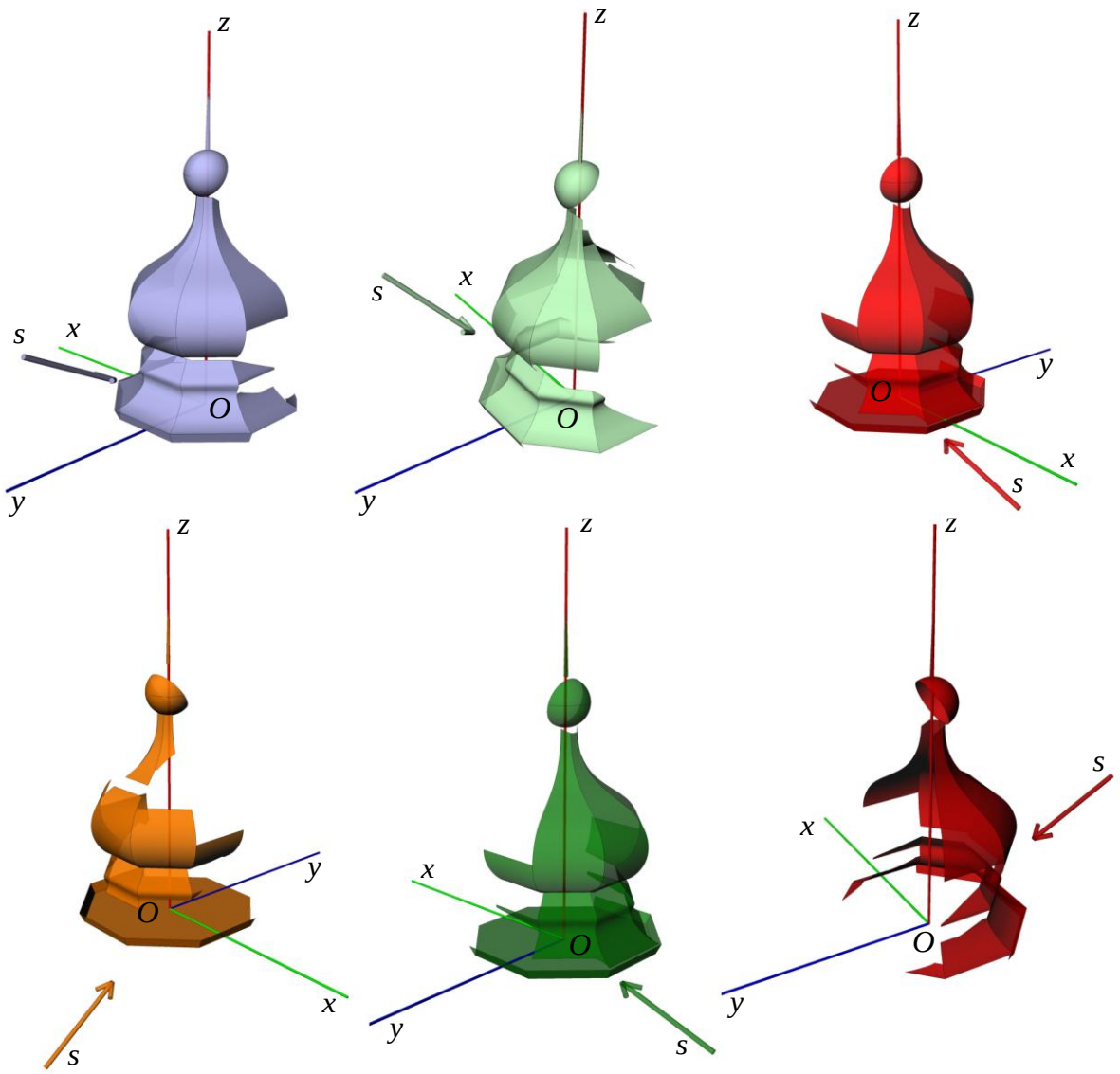
Obrázek 2.8: Model skenované věže kostela s vyznačenými body $P, Q, R, \ldots$, které budou sledovány při registraci

K většině skenovacích zařízení je dodáváno též podpůrné a doplňkové programové vybavení umožňující základní operace s naskenovanými daty – například software pro převod nasnímaných bodů do požadovaného formátu, který je podporovaný konkrétní aplikací pro editaci 3D objektů. Existují též komerční softwary pro tvorbu registrace. Algoritmy pro registraci jsou vesměs založeny na podobných principech. Uveďme tedy stručný popis této metody. Vše si ukážeme na příkladě 3D skenování věže kostela, viz obrázek 2.8.

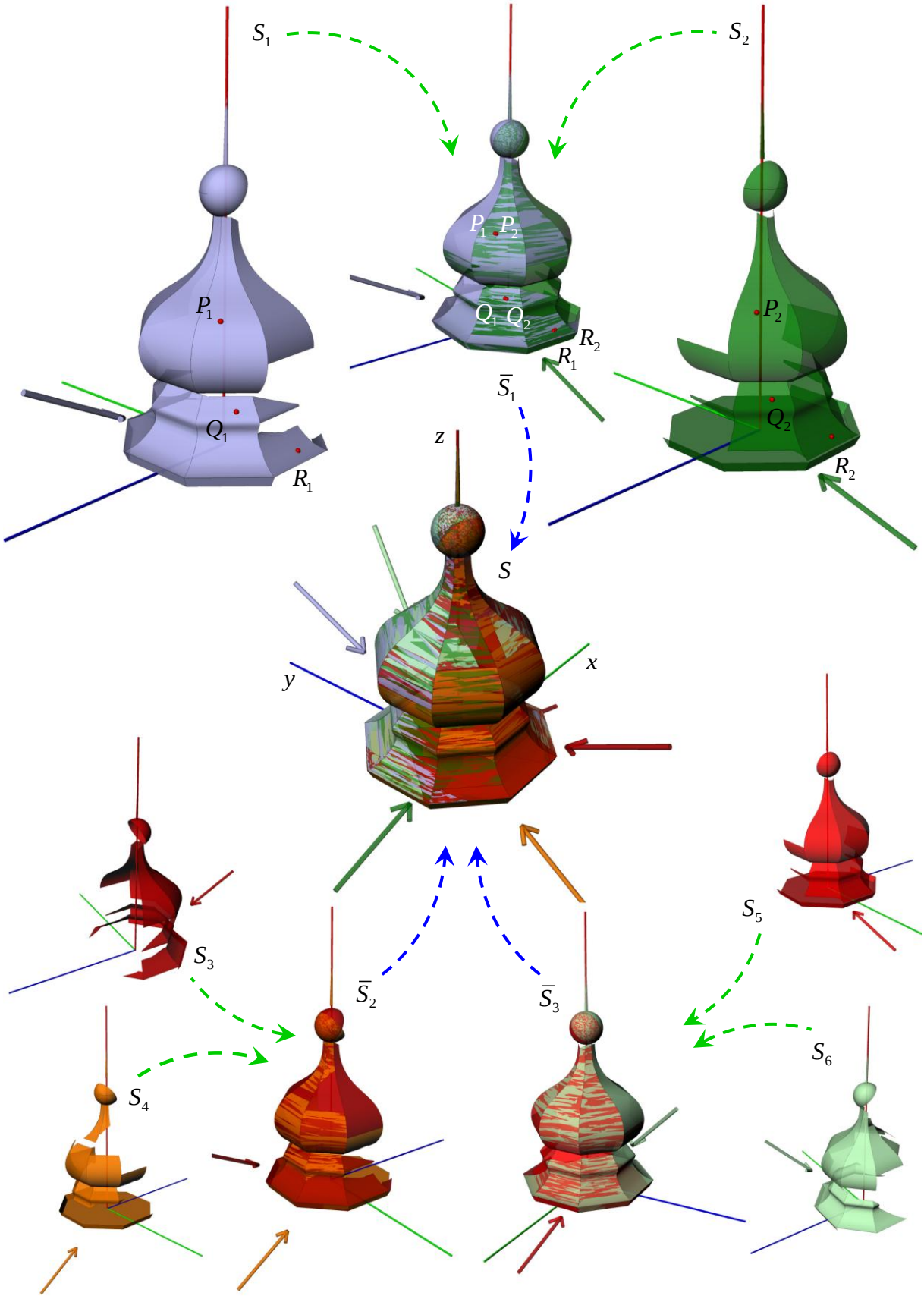
Nejprve je nutné určit na skenovaném povrchu několik bodů, jejichž nasnímané prostorové souřadnice potom budou sledovány v jednotlivých mračnech. Na obrázku 2.8 jsou vyznačeny body P, Q, R (v reálné situaci se samozřejmě určuje bodů více). V každé dílčí bodové množině vzniklé skenováním objektu nalezneme obrazy těchto bodů. Při slučování dílčích bodových množin půjde vždy o to, aby vyznačené body korespondovaly. To znamená, body, které

v naskenované bodové množině reprezentují (alespoň přibližně) ty samé body na reálném povrchu, se ve výsledném bodovém mračnu více či méně shodují. Při slučování půjde vždy pouze o aproximaci, neboť už při 3D skenování může docházet k nepřesnostem v měření.

Na obrázku 2.9 je znázorněno šest bodových množin $S_1, S_2, S_3, S_4, S_5, S_6$ v různých kartézských soustavách souřadnic, ty budeme postupně slučovat. Tyto bodové množiny vznikly skenováním věže z různých pohledů. Hlavní směr pohledu je v obrázku vždy vyznačen šipkou, zakreslena je i kartézská soustava souřadnic. Pro lepší představu jsou jednotlivá mračna bodů znázorněna jen symbolicky jako části ploch. Ve skutečnosti se tedy jedná o bodové množiny. Registrace je ilustrována na dalším obrázku 2.10.



Obrázek 2.9: Šest bodových množin – vyznačená je vždy kartézská soustava souřadnic $\{O, x, y, z\}$ a hlavní směr pohledu s



Obrázek 2.10: Registrace

Mračna bodů, která v prvním kroku slučujeme, jsou označeny S_1, S_2 a dvojice odpovídajících si bodů jsou značeny vždy $(P_1, P_2), (Q_1, Q_2), (R_1, R_2), \ldots$, jsou to naměřené souřadnice bodů $P, Q, R, \ldots$ na povrchu reálného objektu. Díky nepřesnostem v měření po sloučení mračen S_1, S_2 většinou nedochází k úplné shodě, tedy $P_1 \neq P_2, Q_1 \neq Q_2, R_1 \neq R_2, \ldots$, jde pouze o aproximaci (přibližnou shodu) bodů ve sloučeném mračnu popsaném již v jedné kartézské soustavě souřadnic. Aproximace se provádí algoritmicky. Podstata algoritmu je následující. Mračno bodů (například S_2) se přemístí do nové pozice tak, že body $P_2, Q_2, R_2, \ldots$ jsou co nejblíže odpovídajícím bodům $P_1, Q_1, R_1, \ldots$. Je zřejmé, že se tento postup musí aplikovat pro každou dvojici mračen bodů. Na obrázku 2.10 je tedy to samé zopakováno také pro dvojice mračen S_3, S_4 a S_5, S_6. Následně se slučují $\overline{S}_1, \overline{S}_2, \overline{S}_3$, kde $\overline{S}_1$ vzniklo sloučením S_1, S_2, $\overline{S}_2$ vzniklo sloučením S_3, S_4 a $\overline{S}_3$ vzniklo sloučením S_5, S_6 atd. Nakonec tedy získáme výsledné mračno bodů S, které je složeno z jednotlivých mračen bodů $S_1, S_2, S_3, S_4, S_5, S_6$. Toto spojení mračen bodů bývá označováno jako **globální registrace** (*global registration*).

Dalším krokem, který následuje po globální registraci, je **lokální registrace** (*local registration*). Jde o optimalizaci sloučeného mračna, která představuje poměrně komplikovaný problém. Ve výsledném mračnu, které jsme získali sloučením dílčích měření, nyní odpovídá jednomu bodu na reálném povrchu skupina blízkých bodů. Lokální registrace je vícedimenzionální (závisí na počtu dílčích skenování) optimalizační problém hledání příslušného minima. Stručně tuto problematiku popisuje například (Pottmann a *kol.*, 2007).

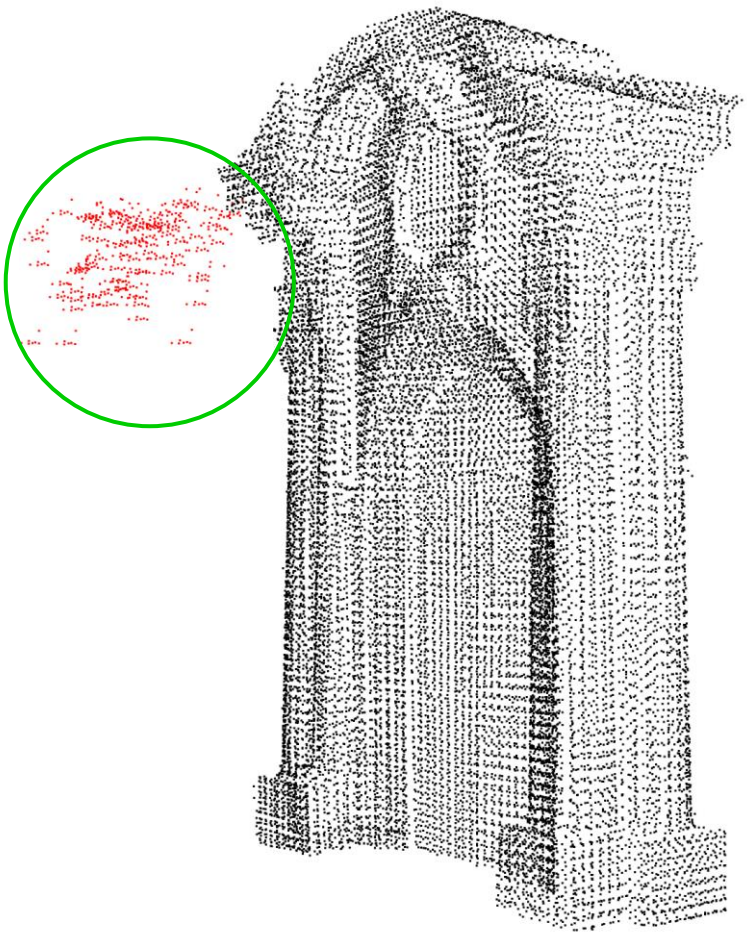
Nebudeme se touto problematikou více zabývat, neboť není hlavním předmětem našeho zájmu. V experimentech máme k dispozici data, která jsou již upravena komerčními softwary dodávanými k danému typu skeneru výrobcem. Pracujeme tedy s mračnem bodů již v jedné kartézské soustavě souřadnic a dále tuto bodovou množinu upravujeme, tj. odstraňujeme nepřesnosti v měření, odlehlé nebo nadbytečné body. Popis procesu registrace zde uvádíme pouze pro úplnost.

2.4 Odstranění nadbytečných bodů a šumů

Nyní předpokládejme, že máme k dispozici mračno bodů v již jedné kartézské soustavě souřadnic, tedy bodovou množinu, která byla zpracována registrací. Jak víme z předchozího, toto výsledné mračno bodů může obsahovat redundantní data (Iske, 2004; Ahn, 2004; Dey, 2007), tedy body, které nepřináší žádnou novou informaci nebo leží navzájem příliš blízko sebe. Může se také stát, že mračno bodů obsahuje části, které nepatří skenovanému objektu. Jedná se například o naskenované okolí nebo u skenování součástky o naskenovanou ruku nebo držák, ve kterém je součástka upevněna apod. Tyto nadbytečné body budeme v dalších částech rekonstrukce odstraňovat.

Mračno bodů, které jsme získali jako výsledek registrace, obsahuje také chyby v měření, které mohou být rozděleny zhruba do dvou kategorií – **odlehlé body** (*outliers*), tj. body, které leží daleko od shluku bodů v mračnu nebo **šumy** (*noise*), které vznikají díky nepřesnostem v měření (Iske, 2004; Pottmann *a kol.*, 2007).

Odlehlé body se v mračnu objevují díky špatným podmínkám při skenování. Jsou velmi jednoduše rozpoznatelné, je možné je tedy detekovat a odstranit nejen algoritmicky, ale také interaktivně. Podobně je tomu tak i s naskenováním částí, které objektu nepatří, lze je odstranit ručně.

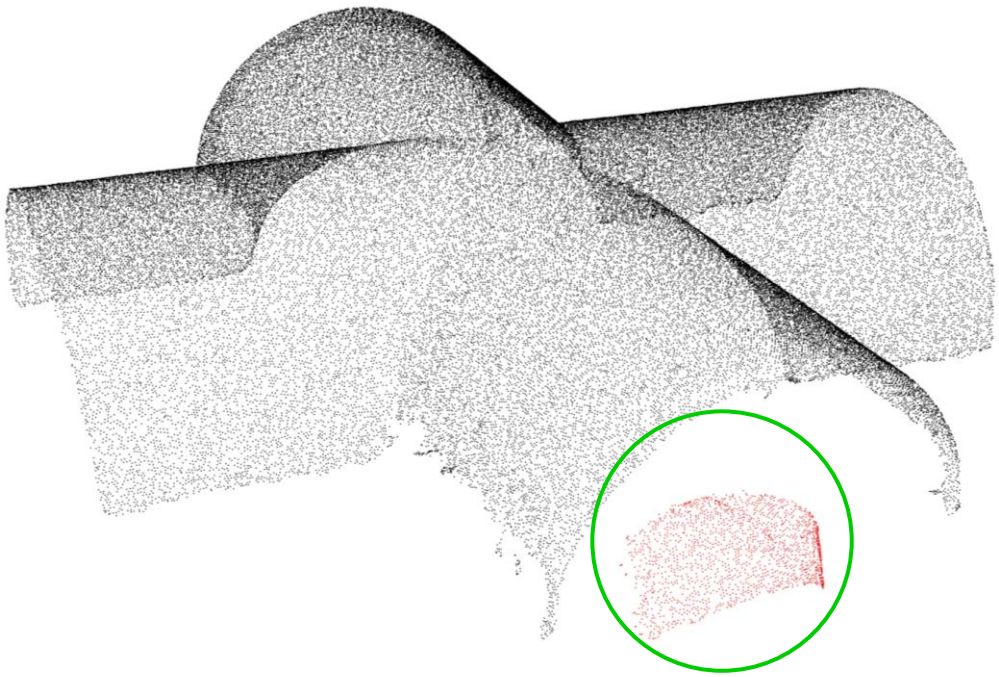


Obrázek 2.11: Mračno bodů reprezentující reálný povrch s odlehlými body (červeně), které jsou odstraněny algoritmicky nebo ručně v modelovacím softwaru.

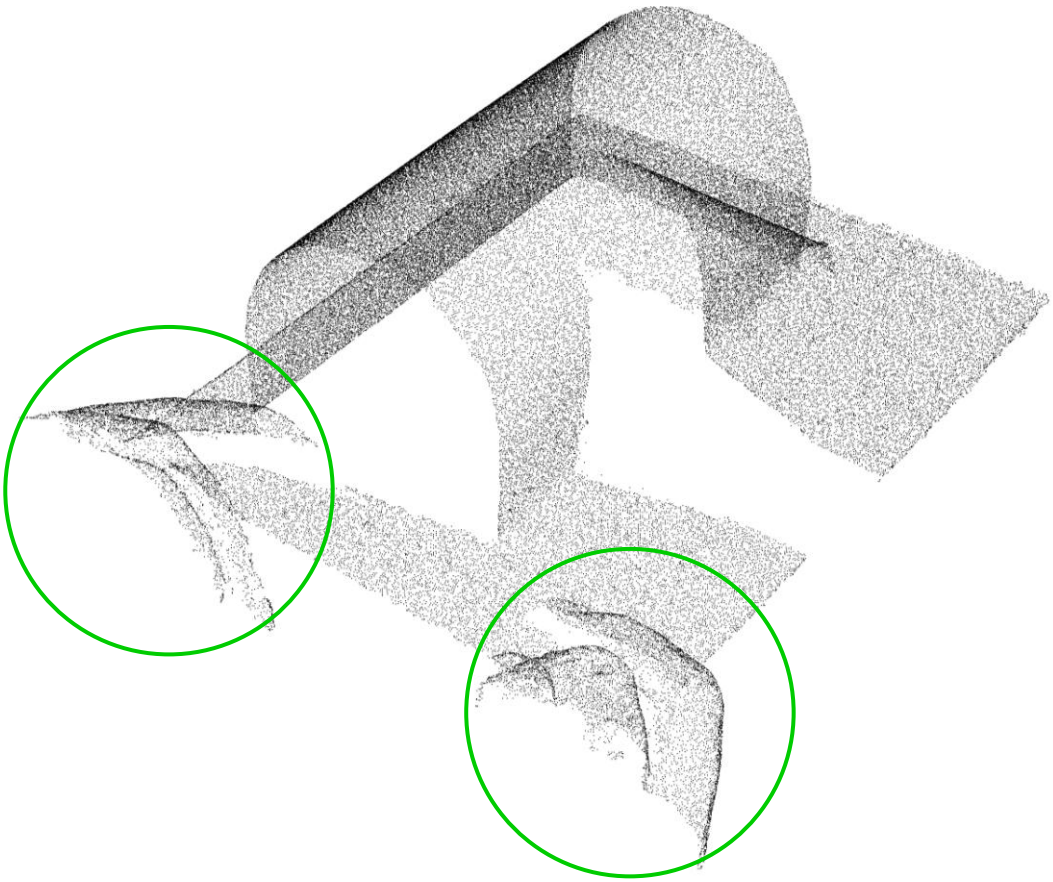
Při 3D skenování velmi často dochází k nepřesnostem v měření. Rozlišení jednotlivých skenovacích zařízení nás totiž limituje v přesnosti. Takto vznikají další chyby v měření, které se vyskytují téměř náhodně, a souhrnně je nazýváme šumy. Jestliže známe přesnost daného skenovacího zařízení, šumy mohou být odstraněny automaticky vhodným algoritmem. Odstraňování šumů může mít za nežádoucí důsledek zaoblování ostrých hran objektů. Je tedy nutné pracovat s odstraňováním bodů z mračna velmi opatrně a odstranění šumů například spojit s vyhlazováním povrchu v polygonální fázi.

Obrázky 2.11-2.13 ilustrují případy nežádoucích bodů ve výsledném mračnu. Na obrázcích 2.11 a 2.12 můžeme vidět odlehlé body, tedy body vzdálené od shluku bodového mračna. Na obrázku 2.13 je potom znázorněna situace, kdy byly při skenování reálného fyzického modelu naskenovány také ruce.

Odstraňování nadbytečných bodů, odlehlých bodů nebo šumů může být prováděno algoritmicky nebo ručně v modelovacím softwaru. V našich experimentech volíme interaktivní odstraňování těchto nepřesností v modelovacím programu Rhinoceros. Algoritmy, které slouží k odfiltrování redundantních bodů, se označují jako ztenčující algoritmy (*thinning algorithms*) a jejich princip je předložen například v (Iske, 2004).



Obrázek 2.12: Mračno bodů reprezentující reálný povrch s odlehlými body (červeně), které jsou odstraněny algoritmicky nebo ručně v modelovacím softwaru.



Obrázek 2.13: Mračno bodů reprezentující reálný geometrický model s body, které nepatří skenovanému objektu

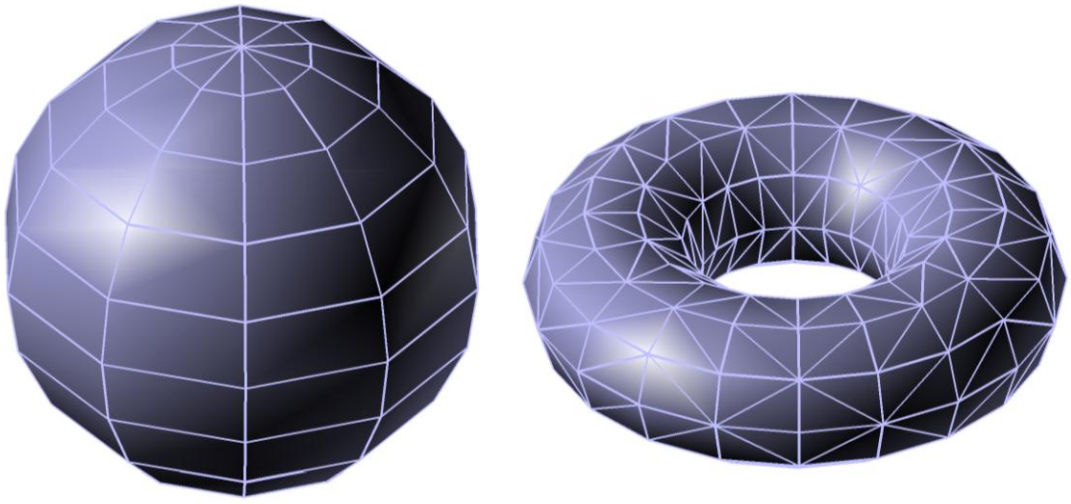
2.5 Aproximace mračna bodů pomocí polygonální sítě

V této části se podrobněji podíváme na polygonální fázi rekonstrukce povrchů z mračna bodů. Vysvětlíme pojem polygonální reprezentace a uvedeme několik známých algoritmů pro hledání polygonální sítě, nejčastěji sítě trojúhelníků, která aproximuje dané mračno bodů.

Jak už bylo řečeno v předchozím, nalezení kvalitní aproximující sítě představuje složitou optimalizační úlohu, která dosud nebyla v plné obecnosti uspokojivě vyřešena (Pottmann a *kol.*, 2007). Tato fáze rekonstrukce povrchu je nepostradatelnou částí, díky níž dostáváme prvotní hraniční reprezentaci naskenovaného mračna, se kterou můžeme dále pracovat.

2.5.1 Polygonální reprezentace 3D objektů

Polygonální (plošková) reprezentace představuje vedle analytické reprezentace jednu z možností, jak je možné vymezit hranici 3D objektu, tedy zachytit informaci o povrchu (vnitřek 3D objektu neuvažujeme). Souhrnně jsou tyto dvě možnosti nazývány jako hraniční reprezentace (*boundary representation*, *B-rep*, *BREP*) a nacházejí svá uplatnění v různých oblastech počítačové grafiky (Žára a *kol.*, 2004). Analytické reprezentaci se budeme věnovat v oddílu 2.6, nyní se zaměřme na reprezentaci polygonální.



Obrázek 2.14: Polygonální sítě – vlevo příklad čtyřúhelníkové sítě aproximující povrch koule, vpravo příklad trojúhelníkové sítě aproximující povrch anuloidu

Povrchy 3D objektů jsou v polygonální reprezentaci aproximovány pomocí sítě rovinných plošek a to nejčastěji trojúhelníků (méně pak čtyřúhelníků a mnohoúhelníků s více vrcholy). Říkáme, že těleso je vyjádřeno polygonální sítí (*polygon mesh*) nebo speciálně sítí trojúhelníků (*triangle mesh*). Jde o pravděpodobně nejvíce používaný způsob zobrazení hranice prostorového modelu. Trojúhelník je volen díky své jednoduchosti – je vždy konvexní

a všechny jeho vrcholy leží v rovině. U mnohoúhelníků je potřeba konvexitu určovat, což může být výpočetně náročné. Polygonální reprezentace zajišťuje snadnou manipulaci s takto vyjádřenými objekty, zobrazování je velice rychlé, neboť většina operací nad těmito objekty vede na lineární interpolaci, která je technicky snadno realizovatelná díky současnému výkonnému hardwaru. Mnoho geometrických výpočtů se provádí právě s objekty reprezentovanými pomocí polygonální sítě. Pokud je nějaký složitý objekt popsán analyticky, převádí se tento model většinou do ploškové reprezentace a výpočty se provádějí s ním. Převod z analytické reprezentace do ploškové bývá velice snadný a rychlý (Žára a *kol.*, 2004).

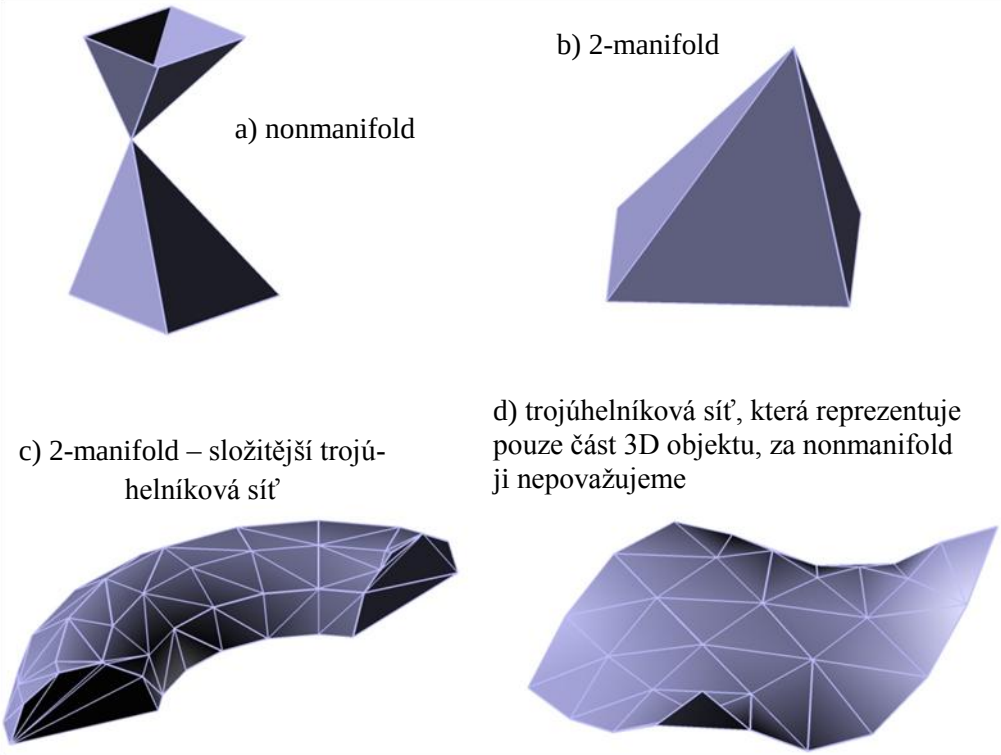
Nevýhodou polygonální reprezentace je to, že zachycuje pouze hrubou geometrii tělesa bez větších detailů. Pro modelování tedy může být nevýhodná.

Polygonální síť je složena z vrcholů, hran a stěn (mnohoúhelníků) tak, že jedna hrana je společná nejvýše dvěma stěnám. Stěna je tvořena uzavřenou posloupností hran. Hrana spojuje vždy dva vrcholy, přičemž z vrcholu vycházejí nejméně dvě hrany. Příklady polygonálních sítí je možné vidět na ilustrujícím obrázku 2.14. Často je potřeba vyloučit nežádoucí konfigurace v síti, které se nazývají *nonmanifold* (Shirley a Marschner, 2009). Zjednodušeně řečeno, nonmanifold je plošková reprezentace, která neodpovídá skutečnému 3D objektu. Například dvě stěny v síti, které mají společný právě jeden vrchol. *2-manifold* potom označuje naopak 3D objekt odpovídající reálnému tělesu. V dalších kapitolách budeme pracovat většinou jen s částí ploškové reprezentace daného 3D objektu, nebudeme však tuto část považovat za nonmanifold. Například budeme-li rekonstruovat klenbu, zaměříme se na rekonstrukci pouze části této plochy, plošková reprezentace tedy nebude reprezentovat těleso. Některé případy 2-manifoldů a nonmanifoldů jsou ukázány v příkladě 2.2.

Datová struktura, která polygonální síť popisuje, bývá rozdělena do dvou částí – *geometrické* a *topologické* (Žára a *kol.*, 2004). V geometrické části jsou zaznamenány souřadnice vrcholů stěn, topologická část obsahuje údaje o tom, které vrcholy tvoří stěnu, případně které stěny spolu sousedí. Toto rozdělení má výhody při práci s polygonálními sítěmi, například při geometrických transformacích. V mnoha aplikacích je také potřeba polygonální síť orientovat, to znamená určit „rub“ a „líc“ plochy, kterou síť popisuje. Vrcholy každé stěny jsou zaznamenány vždy ve stejném pořadí (například proti směru hodinových ručiček). Potom můžeme u každé stěny určit její orientaci. Pokud jsou všechny stěny v polygonální síti shodně orientovány, říkáme, že je síť orientována (Shirley a Marschner, 2009).

Příklad 2.2: *PŘÍKLADY NĚKTERÝCH TYPŮ 2-MANIFOLDU A NONMANIFOLDU.* Sledujme obrázek 2.15. Typický příklad nonmanifoldu ilustruje možnost a), tedy vrchol v síti sdílí několik stěn v takové konfiguraci, jako je ukázáno na obrázku. Tuto síť bychom mohli v reálu vytvořit pouze zdvojením vrcholu. V případě b) jde už o 2-manifold. Složitější příklad polygonální sítě, která je 2-manifoldem ukazuje případ c). Některé algoritmy vyžadují, aby polygonální sítě byly 2-manifoldem (tedy případy b) a c)), tedy takovou polygonální reprezentací, která odpovídá skutečnému 3D objektu. Nicméně, někdy je žádoucí pracovat i s takovými sítěmi, které představují jen část nějaké ploškové reprezentace 3D objektu, nebudeme však tuto část za nonmanifold

považovat. Povolíme tedy, aby polygonální síť obsahovala také hrany, které patří do její hranice (případ d)).



Obrázek 2.15: Příklady typů 2-manifoldu a nonmanifoldu

Výběr reprezentace polygonální sítě v počítači bude vždy záviset na konkrétní aplikaci. Uveďme alespoň nejznámější z nich: *explicitní*, *ukazatele do seznamu vrcholů* a *ukazatele do seznamu hran* (Foley a *kol.*, 1995).

V explicitní reprezentaci je každý mnohoúhelník popsán seznamem souřadnic svých vrcholů. Vrcholy jsou zaznamenány vždy ve stejném pořadí. Mezi po sobě jdoucími vrcholy a mezi prvním a posledním vrcholem je vždy hrana. Nevýhoda této reprezentace spočívá v tom, že souřadnice vrcholů patřících více stěnám jsou uloženy vícekrát. Chybí také informace o sdílených hranách a vrcholech. Například chceme-li zjistit, které hrany vycházejí z daného vrcholu, musíme projít všechny stěny, které tento vrchol obsahují. Problém nastává také při vykreslování, neboť každá sdílená hrana se vykresluje dvakrát.

Polygonální sítě reprezentované pomocí ukazatelů do seznamu vrcholů mají každý vrchol uložený jen jednou v seznamu vrcholů V. Potom je každá stěna polygonální sítě definovaná posloupností indexů do seznamu vrcholů V. Výhoda oproti explicitní reprezentaci je právě v ušetření místa díky ukládání každého vrcholu pouze jednou. Navíc souřadnice vrcholu mohou být snadno změněny. Stále však chybí informace o společných hranách a při vykreslování se sdílené hrany vykreslují dvakrát.

Tyto nedostatky řeší reprezentace pomocí ukazatelů do seznamu hran E. Opět máme k dispozici seznam vrcholů V, každá stěna polygonální sítě je definována posloupností indexů nikoli do seznamu vrcholů, ale do seznamu hran, ve které se každá hrana vyskytuje pouze jednou. Každá hrana v seznamu hran E je určena ukazateli do seznamu V a ukazateli na jednu nebo dvě stěny, které ji sdílejí. Stále není snadné určit hrany vycházející z vrcholu, je nutné projít všechny hrany.

Existují další možné reprezentace, které odstraňují zbylé nedostatky, například *winged-edge* reprezentace nebo *half-edge*. K prostudování těchto možných přístupů odkazujeme na literaturu (Foley a *kol.*, 1995; Shirley a Marschner, 2009).

V dalším textu budeme pracovat výhradně s trojúhelníkovými sítěmi. S nutným teoretickým základem je čtenář obeznámen, další podrobnější informace lze najít v literatuře (Foley a *kol.*, 1995; Pottmann *a kol.*, 2007; Shirley a Marschner, 2009; Žára a *kol.*, 2004).

Jako zajímavost uveďme, že polygonální sítě pronikly také do architektury. V architektuře se polygonální sítě používají velice často například k nahrazení obecných ploch po částech rovinnými ploškami. Velice často jsou tyto plošky zhotovovány ze skla.

Zaměřme se nyní na popis existujících metod a algoritmů pro hledání trojúhelníkové sítě popisující dané mračno bodů. Následující přehled uvádíme pro možnost porovnání s našimi metodami a pro lepší pochopení dané problematiky.

2.5.2 Existující algoritmy pro hledání polygonální sítě

V tomto oddíle předkládáme přehled několika existujících algoritmů pro tvorbu polygonální sítě aproximující povrch, který je reprezentován mračnem bodů. Pokud tomu není jinak, algoritmy a jednotlivé metody pojmenováváme vždy podle jejich tvůrců. Více informací k uvedené klasifikaci je možné nalézt v literatuře (Dey, 2007; Jeměljanov, 2004; Varnuška, 2002). Bližší popis těchto známých algoritmů neuvádíme, pouze u každého algoritmu odkazujeme na základní literaturu, ve které jsou uvedené metody podrobně vysvětleny. Klasifikaci metod provádíme na základě principů, na kterých jsou založeny. V každé skupině algoritmů jmenujeme nejvýznamnější zástupce.

a) Dělení prostoru – povrchové metody

Algoritmus Algorri a Schmitt – (Algorri a Schmitt, 1996)
Algoritmus Hoppe a kol. – (Hoppe a *kol.*, 1992)
Algoritmus Bajaj, Bernardini a kol. – (Bernardini a Bajaj, 1997)
Algoritmus α-shapes – (Edelsbrunner a Mücke, 1994)
Algoritmus Normalizovaná síť – (Attali, 1997)

b) Dělení prostoru – objemové metody

Algoritmus Boissonnat – (Boissonnat, 1984)
Algoritmus Isselhard, Brunett a Schreiber – (Isselhard a *kol.*, 1997)
Algoritmus γ-indicator – (Veltkamp. 1994)
Algoritmus α-solids – (Bajaj a *kol.*, 1997)

c) Znaménková funkce (*distance function*)

Algoritmus Hoppe a kol. – (Hoppe a *kol.*, 1992)
Algoritmus Roth a Wibowoo – (Roth a Wibowoo, 1997)
Algoritmus – rekonstrukce pomocí střední osy (*surface construction by medial axes*) – (Bittar a *kol.*, 1995)

d) Warping

Algoritmus – volná deformace prostoru (*spatial free form warping*) – (Ruprecht a *kol.*, 1995)
Algoritmus Algorri a Schmitt – (Algorri a Schmitt, 1996)

e) Inkrementální konstrukce povrchu

Algoritmus – povrchově orientovaná konstrukce – (Boissonnat, 1984)
Algoritmus Mencl a Müller – (Mencl a Müller, 1998)

f) Clustering

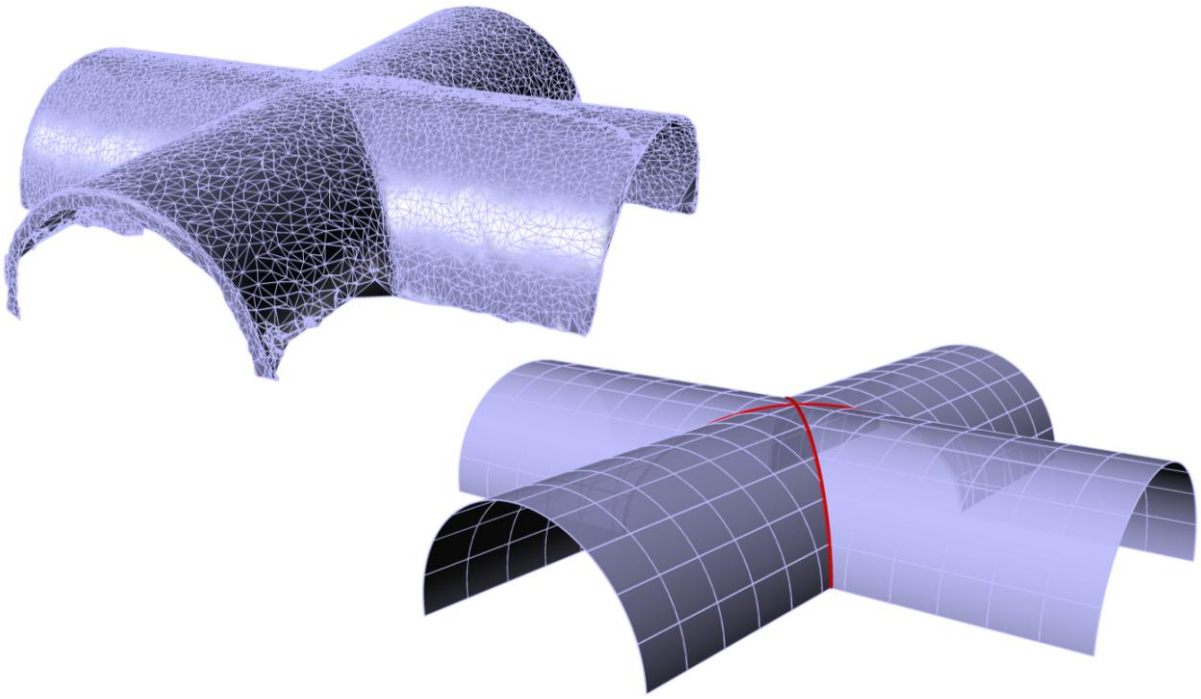
Algoritmus Fua a Sander – (Fua a Sander, 1992)
Algoritmus Mencl a Müller – (Mencl a Müller, 1998)

Přehled algoritmů pro tvorbu polygonální sítě aproximující mračno bodů jsme zdaleka nevyčerpali. Uvedená klasifikace však dostatečně poslouží k orientaci v používaných metodách.

Nevýhody polygonální reprezentace jsou zřejmé – jedná se pouze o po částech lineární aproximaci povrchu. Aby byla tato aproximace dostatečně přesná, je zapotřebí velkého množství dat − hustá množina vrcholů i s incidenčními vazbami. Vstupní data ze 3D skeneru většinou netvoří pravidelnou mřížku, výsledkem zasíťování takového mračna bodů je proto nepravidelná síť trojúhelníků (*Triangular Irregular Network − TIN*) nebo polygonů.

2.6 Segmentace

Mnohem úspornější reprezentace povrchu se provádí pomocí NURBS ploch (*Non Uniform Rational B-Splines*), implicitních ploch nebo primitiv CSG (*Constructive Solid Geometry*). Abychom převedli polygonální reprezentaci na parametrickou, musíme nejdříve jednotlivé části jednoduchých ploch rozpoznat. K tomu slouží tzv. segmentační techniky. Na obrázku 2.16 můžeme vidět příklad povrchové triangulace křížové klenby, jež je tvořena dvěma částmi rotačních válcových ploch se shodnými poloměry a navzájem kolmými osami, a povrch této klenby reprezentovaný válcovými pláty. Ve druhém případě povrchové reprezentace jsou rozpoznány průnikové křivky částí dvou rotačních válcových ploch. To znamená, že ve vstupním mračnu bodů jsou identifikovány body, které leží na tomto průniku, a proloženy jsou hladkou křivkou.



Obrázek 2.16: Křížová klenba, jež je tvořena dvěma částmi rotačních válcových ploch se shodnými poloměry a navzájem kolmými osami – vlevo ukázka povrchové triangulace, vpravo reprezentace povrchu válcovými pláty

Segmentace je nejdůležitější a nejobtížnější součástí tvarové fáze. Jak jsme již v předchozím uvedli, obecně segmentací rozumíme dekompozici mračna bodů na jednotlivé pláty, které můžeme považovat dle nějakého kritéria za diskrétní vzorek části jednoduché plochy (Pottmann *a kol.*, 2007). Jinými slovy jde o nalezení základních geometrických prvků (*geometric primitives*), tj. rovinných částí objektu, částí válcových, kuželových, kulových ploch nebo jiných speciálních ploch. Při segmentaci rozdělíme vstupní data naskenovaného objektu na komponenty, které následně aproximujeme daným typem plochy. Nepopisujeme tedy povrch pomocí jedné plochy, což v praxi většinou není ani možné, ale pomocí plátů, které

na sebe napojujeme. Je důležité správně rozpoznat skutečné hrany objektu, ve kterých budeme jednotlivé části napojovat ostře tj. spojitostí G^0.

Segmentace sleduje dva cíle. V první řadě musí být každá komponenta aproximována částí geometrického primitiva. Za druhé vyžadujeme, aby výsledných částí bylo co možná nejméně a aby oddělující hrany byly skutečné. Tedy dvě sousední komponenty musí představovat různá primitiva s různou geometrií.

Klíčovou roli ve všech segmentačních metodách hraje vektorové pole normál, resp. jeho odhad. Normály určují osvětlení i obrysy plochy a jsou velmi důležité pro vizuální dojem. Ve skutečnosti vnímáme změnu průběhu normál citlivěji než odchylky vzdáleností (Cohen-Steiner *a kol.*, 2004). Míra změny normály plochy určuje, jak se plocha ohýbá, tj. jak rychle se odchyluje od tečné roviny v daném bodě. Nespojitost vektorového pole normál znamená pro plochu nejvýše G^0 spojitost, tj. poukazuje na hrany, přehyby či vzájemné překrývání ploch.

Změnu normály plochy v daném směru v tečné roviny určuje kovariantní derivace vektorového pole normál N ve směru vektoru v, tzv. tvarový operátor (*shape operator*) $S(v) = -\nabla_v N$ (Abbena a *kol.*, 2006). Tento symetrický lineární operátor tečné roviny můžeme reprezentovat maticí S typu (2×2). Poznamenejme, že matice S nemusí být nutně symetrická, protože bázové vektory tečné roviny nemusí být navzájem kolmé. Důležité jsou invarianty operátoru nezávislé na volbě báze. Vlastní čísla matice S jsou *hlavní křivosti* k_1, k_2 a vlastní vektory matice S určují *hlavní směry* (Boček a Kubát, 1983; Budinský a Kepr, 1970). Pomocí matice S tvarového operátoru jednoduše vypočítáme *Gaussovu křivost plochy* K i *střední křivost plochy* H (Boček a Kubát, 1983; Budinský a Kepr, 1970), tj.

$$K = k_1 \cdot k_2 = \det(\mathrm{S}),\ H = \frac{k_1 + k_2}{2} = \frac{1}{2}\,\mathrm{tr}\ S, \tag{2.1}$$

kde tr S je stopa matice S.

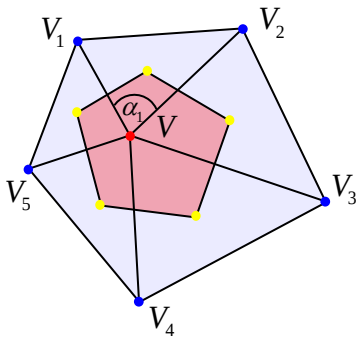
2.6.1 Odhad křivostí plochy

Lokální vlastnosti u spojitých ploch lze dobře popisovat pomocí křivostí plochy, u polygonální reprezentace se musíme spokojit pouze s diskrétním odhadem křivostí. Hlavním problémem při odhadování křivosti plochy ve vrcholu diskrétní reprezentace je, že křivost je lokální vlastnost a jako taková je citlivá na nepřesnosti měření. Cílem je, aby navržený algoritmus byl při rozumné výpočetní složitosti co možná nejrobustnější. Existují desítky heuristických metod pro odhady křivostí plochy (Petitjean, 2002**;** Trucco a Fisher, 1995). Pokud je nám známo, v současné době není propracované žádné srovnávací kritérium, jež by umožnilo klasifikaci metod. Je to zřejmě dáno tím, že na kvalitu odhadu má vliv mnoho faktorů: typ povrchu, šum i způsob diskretizace povrchu (McIvor a Valkenburg, 1997). Mezi technikami pro odhad křivostí je možné vysledovat tři různé přístupy.

a) Diskrétní diferenciální geometrie

Pokud je povrch reprezentován sítí trojúhelníků, odhadujeme křivost plochy v daných bodech na základě nejbližších sousedů. Nechť V je vrchol v povrchové triangulaci a V_i, $i = 1,2,...k$ jsou sousední vrcholy, tj. všechny vrcholy trojúhelníků, které jsou incidentní s vrcholem V.

Nejpoužívanější a nejrychlejší způsob určení odhadu Gaussovy křivosti ve vrcholu V, značme K_p, je využití tzv. úhlového defektu, tedy reálného čísla definovaného jako $2\pi - \sum_{i=1}^{k} \alpha_i$, kde α_i jsou úhly při vrcholu V všech trojúhelníků incidentních s bodem V. Odhad lokální Gaussovy křivosti ve vrcholu V získáme vydělením úhlového defektu vhodnou konstantou λ. Její volba je empirická, záleží na hustotě i důvěryhodnosti dat. V přehledu (McIvor a Valkenburg, 1997) je použit součet obsahu trojúhelníků incidentních s vrcholem V a obsah k-úhelníku s vrcholy v těžištích těchto trojúhelníků, viz obrázek 2.17. Zapíšeme-li odhad lokální Gaussovy křivosti ve vrcholu V symbolicky, dostáváme



Obrázek 2.17: Trojúhelníky v povrchové triangulaci incidentní s vrcholem V

(2.2) $$K_p = \frac{1}{\lambda}\left(2\pi - \sum_{i=1}^{k} \alpha_i\right).$$

b) Aproximace okolí bodu hladkou plochou

Okolí bodu V je proloženo vhodnou, většinou kvadratickou plochou. Nabízí se užít oskulačního paraboloidu, ale empirické výsledky ukazují, že ne vždy je výpočetní náročnost úměrná kvalitě výsledného odhadu (Trucco a Fisher, 1995). Pro konstrukci oskulačního paraboloidu je třeba nejprve odhadnout normálu povrchu reprezentovaného mračnem bodů v bodě V a zapsat souřadnice sousedních vrcholů V_i, $i = 1,2,...k$ ve vhodné bázi. Provedeme-li transformaci kartézské soustavy souřadnic v prostoru do nové polohy tak, že osu z ztotožníme s odhadnutou normálou, osy x a y zvolíme libovolně a počátek splyne s bodem V, bude hledaný paraboloid s vrcholem v bodě V popsán kvadratickou funkcí

(2.3) $$z = \frac{s_1 x^2 + 2 s_3 xy + s_2 y^2}{2},$$

kde s_1, s_2, s_3 jsou reálné koeficienty. Neznámé koeficienty s_1, s_2, s_3 určíme metodou nejmenších čtverců. Dle typu dat je experimentálně určeno, jak velké okolí vrcholu je třeba započítat.

c) Aproximace křivek na ploše

Každý soused V_i vrcholu V rekonstruovaného povrchu daného mračnem bodů určuje jeden směr v tečné rovině, pro který odhadneme *normálovou křivost plochy* (Boček a Kubát, 1983; Budinský a Kepr, 1970) v bodě *V*. Vektor normály plochy v bodě *V* získáme odhadem tečné roviny, tj. roviny aproximující okolí bodu *V*. Normálovou křivost k_n plochy v tomto směru odhadneme jako křivost kružnice procházející body *V* a V_i jejíž střed leží na určené normále. Dosazením všech normálových křivostí do Eulerova vztahu

$$k_n = k_1 \cos^2(\varphi + \alpha_i) + k_2 \sin^2(\varphi + \alpha_i) \tag{2.4}$$

dostáváme přeurčenou soustavu rovnic pro neznámý úhel φ a hledané hlavní křivosti k_1, k_2. Můžeme ji řešit např. metodou nejmenších čtverců.

2.6.2 Segmentační algoritmy

Existuje celá řada segmentačních technik, jejich úspěšnost závisí na geometrii zkoumaného objektu, přesnosti skeneru a v neposlední řadě zpracování bodové fáze. Segmentační techniky můžeme rozdělit do dvou kategorií.

a) Hranová segmentace (*Edge-based*). V bodovém mračnu se identifikují vrcholy, které patří dle nějakého kritéria hraně, a tyto body se následně spojí křivkou. Výsledkem je hladká křivka představující hranici mezi jednotlivými pláty.

b) Plátová segmentace (*Region-based*). Spojují se vrcholy do oblastí, které přísluší základním plochám. Hrany se poté určují jako průnikové křivky těchto ploch.

Oba dva přístupy mají své silné i slabé stránky. Ukazuje se, že robustní segmentace lze dosáhnout kombinací obou postupů.

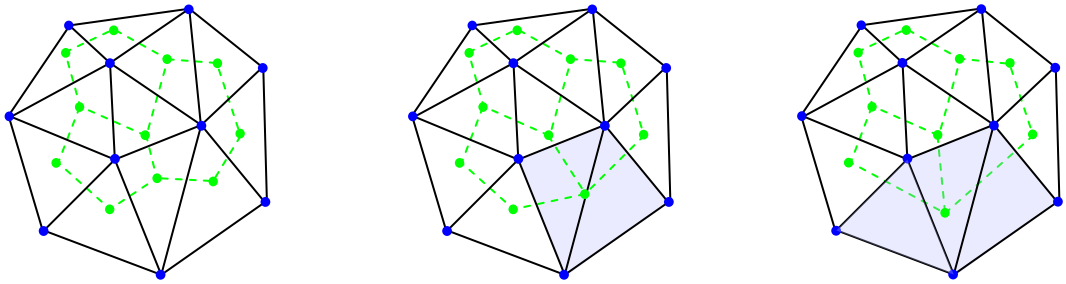
Prvním krokem hranové segmentace je určení bodů zájmu. Jsou to body, jež leží blízko hran − křivek nespojitosti a spojitosti nejvýše G^0. Všechny typy hran se detekují dle průběhu odhadu křivosti plochy. Mezi body zájmu patří body, v nichž nabývá křivost extrému, nebo v nich mění znaménko. Extrémní hodnoty, které jsou nižší, než nastavený práh tolerance, se neuvažují. Práh tolerance nastavíme empiricky dle kvality naměřených bodů, obecně se doporučuje hodnota kolem desetiny největší křivosti. Dalším krokem je interpolovat body

spojitými křivkami − hran napojení plátů. Pro určení incidenčních vazeb nejprve zkoumáme širší okolí každého z bodů a zjišťujeme, zda se křivost v jejich okolí chová podobně. Tím získáme části hran, jež se snažíme v závěrečné fázi spojit a tím definovat jednotlivé pláty segmentace.

Mezi nejznámější plátové segmentace patří *Split and merge* (Faugerass *a kol.*, 1983), či zdola pracující algoritmus *Region growing* (Trucco a Fisher, 1995). Technika *Split and merge* rekurzivně dělí nasnímaný objekt, až zůstanou jen komponenty, jež je možné aproximovat jedním plátem (většinou parametricky určeným). Tím ale vzniknou umělé hrany, proto se v druhém kroku spojují sousední komponenty, jež s danou tolerancí splňují kritéria stejné geometrie. Naproti tomu u přírůstkového algoritmu *Region growing* se začíná se vzorkem bodů (*seed*), jež určují jednotlivé komponenty. K těmto bodům se poté přidávají blízké body, pokud vykazují stejné lokální vlastnosti. Na závěr se kontroluje, zda lze spojit sousední komponenty. Klíčovou roli hraje správná volba počáteční množiny bodů. Většinou se jako kritérium výběru používá odhad křivostí ploch (Trucco a Fisher, 1995).

V posledních deseti letech jsou intenzivně vyvíjeny techniky pro automatickou rekonstrukci 3D objektů. Plně automatická segmentace nebude asi nikdy poskytovat optimální plátování, pokud ale využijeme informaci o typech ploch, můžeme pro kvalitně změřené útvary už dnes dosáhnout velmi dobrých výsledků. Princip takových metod spočívá ve sloučení plátové segmentace a prokládání bodů plátů daným typem plochy. Mnohé kopírují přístup ztenčujících algoritmů polygonální sítě (Attene *a kol.*, 2006), některé využívají shlukové analýzy pro rozhodnutí o počátečním vzorku generujících bodů a počtu segmentů. Za všechny uveďme příklad úspěšné metody pro automatickou segmentaci, která je založena na hierarchickém shlukování stěn (*Hierarchical Face Clustering*), viz obrázek 2.18. Autoři (Cohen-Steiner *a kol.*, 2004) jej použili pro segmentaci po částech rovinných útvarů, v článku (Attene *a kol.*, 2006) je jeho užití rozšířeno i na objekty, které se skládají z části kulových, válcových a kuželových ploch. Princip algoritmu je stejný jako u shlukové analýzy. Základní kritériem pro tvorbu shluků trojúhelníků je podobnost mezi objekty. Měření podobnosti lze provádět pomocí vhodné míry vzdálenosti. Shlukem rozumíme souvislou množinu trojúhelníků, tj. plát segmentace. Na začátku je každý trojúhelník sám o sobě shlukem. Uvažujeme-li duální graf dané triangulace, tj. trojúhelníky reprezentujeme vrcholy, jež spojíme duálními hranami, pokud mají trojúhelníky společnou hranu. Duální hrany ohodnotíme mírou vhodnosti její kontrakce. V každém iteračním kroku vybereme duální hranu s nejvyšší hodnotou a nahradíme je duálním vrcholem. Ten představuje nový sjednocený shluk. Aktualizujeme incidenční vazby nového duálního grafu a přepočítáme ohodnocení duálních hran.

Problémem je volit správný způsob ohodnocení duálních hran pro kontrakci. Zde využijeme informaci o typu ploch. Pro každou duální hranu určíme vhodnou aproximační plochu. Hodnota hrany pro kontrakci pak může být odvozena od součtu čtverců vzdáleností bodů obou shluků od aproximační plochy, podobně jako u metody nejmenších čtverců. Autoři (Cohen-Steiner *a kol.*, 2004) doporučují namísto metriky závislé na vzdálenosti bodů, metriky odvozené od změny normál.



Obrázek 2.18: Hierarchické shlukování stěn – trojúhelníková síť a duální graf (zeleně) a první dvě kontrakce duální hrany

2.7 Prokládání povrchu plochou

Algoritmy, které řeší prokládání bodů v rovině křivkou nebo bodů v prostoru křivkou či plochou (*curve and surface fitting*), viz například (Ahn, 2004), hrají důležitou roli v mnoha technických oblastech a oborech. Při prokládání bodů hledáme nejlepší aproximaci, která ve smyslu zvoleného kritéria vhodně body reprezentuje. Existuje celá řada aproximačních přístupů v závislosti od typu dat i hledané reprezentace plochy. Přehled metod užívaných pro rekonstrukci ploch z naskenovaných bodů je podán např. v (Nielson, 1993).

Ve většině případů je aproximace dat založena na metodě nejmenších čtverců, tedy minimalizaci součtu druhých mocnin nějakých předem definovaných odchylek hledaného modelu (křivky či plochy) od daných dat. Použitou odchylku budeme značit jako *míru chyby* (*error measures*) a součet jejich druhých mocnin jako *chybovou funkci*. Chybová funkce je funkce, jejíž minimum hledáme. Výklad principu metody je podrobně popsán v kapitole 3.

Aproximaci dat metodou nejmenších čtverců můžeme dělit do dvou kategorií – *algebraické prokládání* (*algebraic fitting*) a *geometrické prokládání* (*geometric fitting*), viz (Ahn, 2004). Tyto dva přístupy se liší v definici příslušné míry chyby. Chybová funkce algebraického prokládání je odvozena od reprezentace objektu funkčním předpisem, je tedy závislá na soustavě souřadné. Naproti tomu geometrické prokládání používá reziduum dané eukleidovskou vzdáleností bodů od modelu (křivky či plochy).

Předpokládejme, že jsou naskenovaná data povrchu objektu segmentována do jednotlivých plátů. Nyní je naším cílem vhodným způsobem tyto množiny popsat. V mnohých případech můžeme využít informaci o typu skenované plochy, u některých ploch jsou důležité charakteristiky plátu zjištěny segmentací automaticky (rovina, kulová plocha, válcová plocha).

Prokládání dat se liší dle typu hledané reprezentace plochy. Nejširší uplatnění má parametrické zadání ve formě NURBS *(Non Uniform Rational B-Spline*), zatímco pro rovinné a kvadratické pláty je mnohdy výhodné implicitní zadání. Popis přímé metody nejmenších čtverců pro funkce i pro NURBS najdeme v klasické literatuře CAGD, např. (Hoschek a Lasser, 1993; Foley 1987). Popišme si v dalším textu hledání implicitního předpisu pro algebraický plát.

2.7.1 Proložení množiny bodů algebraickou plochou

Pro algebraickou plochu stupně *m* existuje implicitní polynomiální vyjádření, tj.

$$F(x,y,z)=\sum_{\substack{i,j,k=0\\ i+j+k\le m}}^{m} a_{ijk}x^i y^j z^k = 0, \tag{2.5}$$

kde a_{ijk} jsou reálné koeficienty, z nichž ty, pro něž je $i+j+k=m$, nejsou všechny rovny nule. Koeficienty polynomu jsou tzv. algebraické koeficienty.

Mějme dáno $n+1$ bodů $\{[x_i,y_i,z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3, značme tuto množinu $\{X_i\}_{i=0}^{n}$. Tyto body chceme proložit algebraickou plochou stupně *m*, tedy hledáme vhodné koeficienty a_{ijk} bázových polynomiálních funkcí $F_{ijk}(x,y,z)=x^i y^j z^k$, $i+j+k\le m$. Poznamenejme, že vyjádření (2.5) je pro danou plochu určeno jednoznačně až na nenulový násobek. Můžeme proto bez újmy na obecnosti hledat takové vyjádření, pro něž je $a_{000}=-1$. Jestliže se dosazením souřadnic daného bodu X_i do předpisu (2.5) algebraické plochy anuluje polynom, leží tento bod na ploše, v opačném případě neleží. Pro bod X_i, který neleží na ploše je tedy $F(X_i)\neq 0$. Tato nenulová hodnota polynomu v bodě se nazývá *algebraická vzdálenost* (*algebraic distance*). K určení algebraických koeficientů použijeme minimalizaci součtu druhých mocnin algebraických vzdáleností v každém bodě. Chybová funkce je tedy tvaru

$$H(a)=\sum_{i=0}^{n} F(X_i)^2, \tag{2.6}$$

kde $a\in\mathbb{R}_r$ značí vektor hledaných algebraických koeficientů.

Výhodnost definice chybové funkce snadno nahlédneme při maticovém zápisu soustavy $n+1$ lineárních rovnic, které získáme dosazením bodů $X_i, i=0,1,...,n$ do předpisu (2.5) algebraické plochy, tj.

$$\begin{pmatrix} x_0^m & x_0^{m-1}y_0 & \cdots & y_0^m & y_0^{m-1}z_0 & \cdots & z_0^m & \cdots & z_0x_0^{m-1} & x_0^{m-1} & \cdots & x_0 & y_0 & z_0 \\ x_1^m & x_1^{m-1}y_1 & \cdots & y_1^m & y_1^{m-1}z_1 & \cdots & z_1^m & \cdots & z_1x_1^{m-1} & x_1^{m-1} & \cdots & x_1 & y_1 & z_1 \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ x_n^m & x_n^{m-1}y_n & \cdots & y_n^m & y_n^{m-1}z_n & \cdots & z_n^m & \cdots & z_nx_n^{m-1} & x_n^{m-1} & \cdots & x_n & y_n & z_n \end{pmatrix} \begin{pmatrix} a_{m,0,0} \\ a_{m-1,1,0} \\ \vdots \\ a_{0,0,1} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}. \tag{2.7}$$

Označme matici soustavy *A*, vektor pravých stran $b=(1,1,...,1)^T$. Řešíme-li tuto přeurčenou soustavu ($r<n+1$) lineárních rovnic, potom symbolicky zapsáno

$$Aa=b. \tag{2.8}$$

Přeurčenou soustavu lineárních rovnic (2.8) řešíme metodou nejmenších čtverců, minimalizujeme hodnotu $\|Aa-b\|^2$, tj. chybovou funkci $H(a)$ z (2.6). Pokud plocha neprochází

počátkem a zadané body $\{X_i\}_{i=0}^n$ nejsou v degenerativním rozmístění (kolem jednoho bodu či přímky), je řešení tvaru

$$a = \left(A^T A\right)^{-1} A^T b. \tag{2.9}$$

Poznamenejme, že existuje celá řada numerických přístupů, řešení je možné hledat i v případě singulární či špatně podmíněné matice *A*, viz (Ralston, 1965). Hledání implicitně zadané algebraické plochy má praktické uplatnění především v případě úspěšné segmentace rovinných či kvadratických plátů. Možné přístupy k zohlednění informací o typu kvadratické plochy jsou popsány v článcích (Petitjean, 2002**;** Werghi *a kol.*, 2000).

Výše popsaný postup patří k algebraickému prokládání. Jeho nevýhodou je citlivost na nepřesnosti zdrojových dat a závislost na zvolené soustavě souřadné. Metoda dává dobré výsledky, pokud jsou hodnoty nezávislé veličiny (např. *x, y*) bez chyby a pokud chyba třetí souřadnice splňuje vlastnosti bílého šumu (Cao a Shrikhande, 1991). Takovéto předpoklady jsou ale u reálných dat zřídkakdy splněny. Uvědomme si, že obecně neexistuje praktická interpretace algebraické vzdálenosti a algebraických koeficientů. Například, uvažme prokládání dat rovinou, přičemž algebraická vzdálenost je právě vertikální vzdálenost bodu X_i a roviny. Pokud mají body $\{X_i\}_{i=0}^n$ vyjadřovat vertikální rovinu dostáváme ale nesprávné výsledky. Algebraické prokládání má další nevýhody, které dále popisuje například (Ahn, 2004).

Další možností je použití geometrického prokládání dat. Mezi nejčastěji používané míry chyby, jejichž druhé mocniny potom sčítáme, patří *vzdálenost*, označována také jako *geometrická vzdálenost, kolmá vzdálenost* či *eukleidovská vzdálenost* (*shortest distance, geometric distance, orthogonal distance, Euclidean distance*) bodů od modelu (křivky či plochy). Příslušné prokládání dat křivkou nebo plochou se nazývá geometrické prokládání nebo *ortogonální prokládání* (*orthogonal distance fitting – ODF*). Toto označení uvádí například (Ahn, 2004).

Opět předpokládejme, že je dána množina $n+1$ bodů $\{X_i\}_{i=0}^n$. Chybovou funkci, jejíž minimum hledáme, definujeme jako součet druhých mocnin vzdáleností daných bodů $\{X_i\}_{i=0}^n$ od hledaného modelu, tj.

$$G(a) = \sum_{i=0}^{n} \left\| X_i - X_i' \right\|^2 , \tag{2.10}$$

kde $\{X_i'\}_{i=0}^n$ jsou nejbližší body hledaného modelu od daných bodů $\{X_i\}_{i=0}^n$ a $\|\cdot\|$ značí eukleidovskou normu. Při minimalizaci chybové funkce v (2.10) aktualizujeme vektor parametrů $a \in \mathbb{R}_r$ daného modelu (plochy).

Geometrický význam prokládání dat založeného na minimalizaci sumy čtverců vzdáleností od daného modelu je zřejmý. Geometrické prokládání není tak citlivé na nepřesnosti vstupních dat, na rozdíl od algebraického prokládání dává dobré výsledky i v případě, že jsou všechny souřadnice zatíženy chybou. Výpočet geometrického prokládání je obecně mnohem složitější než při řešení algebraického prokládání dat, jak uvádí (Ahn, 2004). Výjimku tvoří

prokládání dat přímkou nebo rovinou, neboť zde závisí geometrická vzdálenost na parametrech přímky nebo roviny lineárně. V ostatních případech jde o nelineární metodu nejmenších čtverců, která se řeší iteračními metodami (Press *a kol.*, 1992).

Současné přístupy rekonstrukce povrchů jsou založené na popsaném rozdělení procesu rekonstrukce do několika na sebe navazujících fází, kde každá fáze úlohu nějakým způsobem zjednodušuje a předzpracovává pro fázi následující. Tento přístup jsme následovali i v našem výzkumu a výsledky předkládané v této práci tak přispívají k jednotlivým fázím v současnosti používaném rekonstrukčním procesu. V dalších kapitolách se věnujeme konkrétním krokům rekonstrukce povrchů.

Kapitola 3

Analýza bodové množiny – aproximační metody

V této kapitole se věnujeme různým způsobům zpracování bodové množiny v eukleidovské rovině E_2 a v eukleidovském prostoru E_3 známými aproximačními technikami. Předkládáme ukázky použití vybraných metod na konkrétních bodových množinách pro zjišťování jejich orientace.

Zaměřujeme se na podrobný popis známých metod převážně z oblastí numerické matematiky a matematické statistiky, které dále používáme v experimentech. Jedná se o různé typy aproximací dat v rovině přímkou a v prostoru přímkou a rovinou (případně speciálními plochami).

Zcela přirozeně začínáme s aproximací funkcí jedné proměnné a to metodou nejmenších čtverců. Předvedení této základní aproximační techniky považujeme za důležité z několika důvodů. Jedná se o všeobecně známou metodu numerické matematiky užívanou k analýze a zpracování jednorozměrných dat, nejčastěji výsledků nějakého měření. S použitím této metody lze proto jednoduše demonstrovat způsoby výpočtů a dosahování numerické stability, které je možné uplatnit i u jiných technik. Aproximace metodou nejmenších čtverců představuje rovněž výchozí bod pro odvozování dalších metod.

Dále se v této kapitole věnujeme ortogonálnímu prokládání dat přímkou v rovině a v prostoru, které vychází rovněž z principu metody nejmenších čtverců, lze jej však uplatnit v širším okruhu aplikací.

Analogicky popisujeme také aproximaci funkcí dvou proměnných metodou nejmenších čtverců a ortogonální prokládání dat rovinou. Prvně zmiňovaná metoda představuje doplnění teorie o aproximacích. Ortogonální prokládání dat rovinou je pak využito v dalších fázích rekonstrukce povrchů, a proto uvádíme jeho podrobný popis.

Technika ortogonálního prokládání dat přímkou a rovinou je sice známá, není však většinou součástí základních kurzů numerické matematiky a v české literatuře navíc chybí její důkladné odvození s názornou geometrickou interpretací. Z didaktických důvodů tedy ortogonální prokládání dat podrobně odvozujeme.

Závěr kapitoly je věnován analýze hlavních komponent (*Principal Component Analysis)*, kterou uvádíme do souvislosti s ortogonálním prokládáním dat.

Popsané metody jsou použity k analýze bodových množin reprezentující počítačově generovaná data (syntetická), o kterých můžeme předem předpokládat určité specifické geometrické vlastnosti. Podle předem známých vlastností těchto vstupních množin také rozhodujeme, který řešící postup daného podproblému bude zvolen.

3.1 Metody pro zjišťování orientace dat

V této části uvedeme existující metody pro zjišťování orientace dat, které budeme dále porovnávat s vlastními odvozenými metodami. Zabývat se budeme *aproximováním* (přibližným prokládáním, vyjádřením) bodů v rovině a v prostoru přímkou a bodů v prostoru rovinou. Představíme několik možných přístupů založených na minimalizaci součtu druhých mocnin nějakých předem definovaných odchylek hledaného modelu (přímky či roviny) od daných dat. V této fázi pracujeme pouze s bodovou množinou a naším cílem je tuto bodovou množinu popsat a zjistit její charakteristicky. Vstupem úlohy tedy bude množina $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i]\}_{i=0}^{n}$ v eukleidovské rovině E_2 (případně $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3), značme tyto množiny $\{X_i\}_{i=0}^{n}$ pro oba případy. Bodová množina je rigorózně matematicky posloupnost bodů (může obsahovat více totožných bodů). Pořadí bodů v posloupnosti pro nás typicky není důležité, proto hovoříme o množině. Datová reprezentace ve výpočetním prostředí MATLAB však pořadí bodů rozlišuje (body jsou zadány jako seznam reprezentovaný maticí).

Ukážeme základní typ aproximace metodou nejmenších čtverců a následně tuto metodu porovnáme s ortogonálním prokládáním dat. Tyto postupy zmiňujeme proto, abychom se na ně mohli dále odkazovat v našich experimentech. Pro zjištění orientace dat používáme také statistickou analýzu hlavních komponent (PCA). Zdůrazněn bude geometrický význam PCA, který je v literatuře často opomíjen.

3.2 Aproximace funkce metodou nejmenších čtverců

Hlavní úlohou aproximace je proložení dat vhodnou funkcí, která bude data dobře reprezentovat ve smyslu zvoleného kritéria. *Metoda nejmenších čtverců* (*least squares method*) se používá při zpracování nepřesných dat, typicky výsledků nějakého měření, které je zatíženo chybou, a obecně slouží k eliminaci chyby a k získání explicitního vyjádření extenzionální reprezentace dané funkční závislosti. Předpokládáme tedy, že data reprezentují funkční závislost a jsou získána měřením, chyba se přitom projevuje pouze v závisle proměnné (v pozorování). V klasické metodě nejmenších čtverců jde nejčastěji o hledání explicitně vyjádřené aproximující funkce pro naměřená data. Je zřejmé, že hledání funkční závislosti, která by interpolovala naměřená data, není žádoucí, neboť takto bychom chybu v měření propagovali do výsledné explicitní reprezentace. Eliminace chyby je prováděna vzhledem k pevně danému kritériu. Podstata přístupu spočívá v tom, že při daném kritériu můžeme posoudit kvalitu aproximace vzhledem k naměřeným hodnotám. Vždy je předpokládáno, že chyby v různých bodech jsou na sobě navzájem nezávislé.

Pro lepší názornost uveďme použití metody nejmenších čtverců při hledání aproximující funkce pro hodnoty získané měřením, kdy je předem známo, o jakou funkční závislost se jedná (tj. je známý typ explicitní reprezentace aproximující funkce). Vyjdeme přitom z nejpo-

užívanějšího kritéria, čímž bude minimalizace součtu druhých mocnin odchylek aproximujících funkčních hodnot a původních naměřených hodnot.

Nechť je dána množina $\mathcal{X}$ $n+1$ bodů $[x_i, y_i]$, $i = 0,1,...,n$ v eukleidovské rovině E_2, zapisovat budeme $\mathcal{X} = \{[x_i, y_i]\}_{i=0}^{n}$ nebo zkráceně $\mathcal{X} = \{X_i\}_{i=0}^{n}$. Dále je dána množina $k+1$ funkcí φ_j, $j = 0,1,...,k$, kde $k \leq n$, definovaných alespoň ve všech bodech x_i. Za třídu aproximujících funkcí bude brána množina lineárních kombinací

$$\{a_0\varphi_0(x) + a_1\varphi_1(x) + ... + a_k\varphi_k(x); a_j \in \mathbb{R}, j = 0,1,...,k\}. \tag{3.1}$$

Z této množiny vybereme takovou funkci

$$\varphi(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + ... + a_k\varphi_k(x) = \sum_{j=0}^{k} a_j\varphi_j(x), \tag{3.2}$$

pro kterou funkce

$$H(a_0, a_1, ..., a_k) = \sum_{i=0}^{n} (\varphi(x_i) - y_i)^2 \tag{3.3}$$

nabývá minimální hodnoty. Aproximace funkcí metodou nejmenších čtverců je numerická metoda, kde místo spojitého prostoru E_2 uvažujeme pouze konečnou množinu bodů. Tedy zjednodušeně řečeno integrál nahrazujeme sumou, neboť k funkcím vyskytujících se v praxi (například funkce dvou proměnných popisující plochu) je těžké najít primitivní funkci a spočítat určitý integrál. Kdybychom tedy pracovali se spojitým prostorem, funkce (3.3) by byla vyjádřená

$$H(a_0, a_1, ..., a_k) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} (\varphi(x) - y)\, dydx. \tag{3.4}$$

Označme dále $e_i = \varphi(x_i) - y_i$ chybu (reziduum) aproximace v *i*-tém bodě. Reziduum tedy můžeme chápat jako velikost chyby, které se v příslušném bodě při odhadu dopouštíme.

Kdybychom znali přesnou funkci, kterou nyní pomocí konečného množství hodnot navíc zatížených chybami aproximujeme, předpokládáme, že pro nějaké neznámé $k \in \mathbb{N}$ lze tuto funkci vyjádřit jako konečnou lineární kombinaci množiny funkcí φ_j, $j = 0,1,...,k$. V případech, ve kterých využíváme aproximaci metodou nejmenších čtverců, se většinou vyskytuje aproximace přímkou. Předem tedy víme, že se má jednat o lineární funkční závislost, v důsledku chyb vzniklých měřením ale nedokážeme tuto přesnou funkci odhalit. Pokud zvolíme funkce φ_j, $j = 0,1,...,k$ jako mocniny x, funkce $\varphi(x)$ je lineární kombinací polynomů x^j, $j = 0,1,...,k$ s reálnými koeficienty a_j, $j = 0,1,...,k$, aproximační přímku dostáváme v případě $k = 1$.

Poznámka 3.1: Pro volbu $k \geq n$ se jedná o speciální případ *interpolace*, tj. graf výsledné funkce φ přesně prochází body množiny $\mathcal{X}$. Pro případ $k = n$ je tato interpolace jednoznačná, pro $k > n$ je nutné přidat další podmínky (jaké podmínky, závisí na aplikaci). Dále připusťme pouze případ $k = n$. Kromě toho volbu k ovlivňuje také výběr typu funkcí φ_j, $j = 0,1,...,k$. Všimněme si také, že na hodnoty x_i není kladen žádný požadavek, jako je tomu u interpolace, kde je nutné klást hodnoty x_i navzájem různé. Jednotlivé naměřené hodnoty mohou být také různě přesné. V tom případě můžeme požadovat, aby aproximující funkce naměřené hodnoty vyjadřovala v naměřených bodech přesněji. Toho docílíme tím, že jednotlivým bodům přidáme váhy ω_i a zobecníme výraz (3.3) na

$$H(a_0, a_1, ..., a_k) = \sum_{i=0}^{n} \omega_i \left(\varphi(x_i) - y_i\right)^2 , \tag{3.5}$$

kde $\omega_i \in (0, \infty)$, $i = 0,1,...,n$. ■

Hledání minima funkce H z (3.3) závislé na parametrech $a_0, a_1, ..., a_k$ lze řešit standardním postupem. Vypočteme parciální derivace funkce H podle jednotlivých parametrů a ty položíme rovny nule, tj.

$$\frac{\partial H(a_0, a_1, ..., a_k)}{\partial a_j} = 2\sum_{i=0}^{n} \left(\varphi(x_i) - y_i\right)\varphi_j(x_i) = 0 , \tag{3.6}$$

kde $j = 0,1,...,k$. Hledáme tedy lokální minimum funkce H, nutnou podmínkou pro jeho existenci je právě splnění rovnosti (3.6). Po úpravě výrazu (3.6) získáme soustavu $k+1$ lineárních rovnic (tzv. *normálních rovnic*) pro neznámé $a_0, a_1, ..., a_k$, tedy

$$a_0 \sum_{i=0}^{n} \varphi_0(x_i)\varphi_j(x_i) + a_1 \sum_{i=0}^{n} \varphi_1(x_i)\varphi_j(x_i) + ... + a_k \sum_{i=0}^{n} \varphi_k(x_i)\varphi_j(x_i) = \sum_{i=0}^{n} y_i \varphi_j(x_i) , \tag{3.7}$$

kde $j = 0,1,...,k$. Soustavu lineárních rovnic (3.7) můžeme zapsat následovně

$$A = \begin{pmatrix} \sum_{i=0}^{n} \varphi_0(x_i)\varphi_0(x_i) & \sum_{i=0}^{n} \varphi_1(x_i)\varphi_0(x_i) & \cdots & \sum_{i=0}^{n} \varphi_k(x_i)\varphi_0(x_i) \\ \sum_{i=0}^{n} \varphi_0(x_i)\varphi_1(x_i) & \sum_{i=0}^{n} \varphi_1(x_i)\varphi_1(x_i) & \cdots & \sum_{i=0}^{n} \varphi_k(x_i)\varphi_1(x_i) \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^{n} \varphi_0(x_i)\varphi_k(x_i) & \sum_{i=0}^{n} \varphi_1(x_i)\varphi_k(x_i) & \cdots & \sum_{i=0}^{n} \varphi_k(x_i)\varphi_k(x_i) \end{pmatrix}, \; b = \begin{pmatrix} \sum_{i=0}^{n} y_i\varphi_0(x_i) \\ \sum_{i=0}^{n} y_i\varphi_1(x_i) \\ \vdots \\ \sum_{i=0}^{n} y_i\varphi_k(x_i) \end{pmatrix}, \; x = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix}, \tag{3.8}$$

kde $Ax = b$. Je-li matice A regulární, má soustava $Ax = b$ právě jedno řešení.

Definice 3.1 (*Lineární závislost a nezávislost funkcí*). Nechť jsou funkce φ_j, $j = 0,1,...,k$, definovány alespoň ve všech bodech x_i, $i = 0,1,...,n$. Řekneme, že funkce φ_j jsou *lineárně nezávislé* na množině bodů x_i, $i = 0,1,...,n$, jestliže rovnost

$$c_0\varphi_0(x_i) + c_1\varphi_1(x_i) + ... + c_k\varphi_k(x_i) = 0,$$

platí pro všechna x_i, $i = 0,1,...,n$ právě jen pro případ $c_0 = c_1 = ... = c_k = 0$. V opačném případě jsou funkce φ_j *lineárně závislé*. ■

Lze ukázat, že při volbě lineárně nezávislého systému funkcí φ_j, $j = 0,1,...,k$ má úloha aproximace vždy právě jedno řešení, které minimalizuje funkci (3.3) (podmínky (3.6) jsou pouze nutné podmínky pro extrém). Existenci a jednoznačnost řešení soustavy normálních rovnic (3.8), které minimalizuje funkci (3.3), objasňuje následující věta.

Věta 3.1. Je-li systém funkcí φ_j, $j = 0,1,...,k$ lineárně nezávislý, pak existuje právě jedna aproximující funkce φ,

$$\varphi(x) = \sum_{j=0}^{k} a_j\varphi_j(x),$$

která minimalizuje výraz

$$H(a_0, a_1, ..., a_k) = \sum_{i=0}^{n} (\varphi(x_i) - y_i)^2 .$$

Koeficienty $a_0, a_1, ..., a_k$ se naleznou jako jediné řešení soustavy normálních rovnic (3.8). ■

Obdobná věta je uvedena například v (Přikryl, 1988) nebo (Hämmerlin a Hoffmann, 1991), kde je připojen také důkaz.

Uveďme si ještě další možné přístupy v hledání řešení aproximace funkce metodou nejmenších čtverců. Řešení aproximační úlohy lineárně závisí na naměřených hodnotách y_i, $i = 0,1,...,n$. Tyto hodnoty tvoří vektor $y = (y_0, y_1, ..., y_n)^{\mathrm{T}} \in \mathbb{R}_{n+1}$. Jinými slovy výsledná aproximující funkce φ je pro účely aproximace dostatečně reprezentována vektorem svých hodnot v bodech x_i ve tvaru $\varphi = (\varphi(x_0), \varphi(x_1), ..., \varphi(x_n))^{\mathrm{T}} \in \mathbb{R}_{n+1}$. Všimněme si, že v aproximační úloze vystupují funkce φ_p, pro $p = 0,1,...,k$ pouze skrze hodnoty v bodech x_i nikoliv pomocí svého předpisu. Jiné hodnoty v úloze o aproximaci nefigurují, nemohou tedy ovlivnit řešení. Označme dále vektory $\varphi_p = (\varphi_p(x_0), \varphi_p(x_1), ..., \varphi_p(x_n))^{\mathrm{T}} \in \mathbb{R}_{n+1}$, pro $p = 0,1,...,k$. Výraz (3.2) platí v libovolném bodě definičního oboru funkce, speciálně v bodech x_i, a tedy pro odpovídající vektory

$$\varphi = \sum_{p=0}^{k} a_p\varphi_p . \tag{3.9}$$

Aproximační úlohu můžeme nyní formulovat tak, že k danému vektoru $y \in \mathbb{R}_{n+1}$ hledáme vektor φ z lineárního podprostoru generovaného známými vektory φ_p, $p = 0,1,...,k$. S tímto zavedením můžeme také přepsat výraz (3.3) a to následovně

(3.10) $$H = (\varphi - y)\cdot(\varphi - y) = \|\varphi - y\|^2 ,$$

kde $\|\cdot\|$ je eukleidovská norma indukovaná skalárním součinem. Skalární součin vektorů je zaveden obvyklým způsobem, tj. pro vektory $u = (u_0, u_1, ..., u_n)^{\mathrm{T}}$ a $v = (v_0, v_1, ..., v_n)^{\mathrm{T}}$ je skalární součin číslo $u \cdot v = u_0 v_0 + u_1 v_1 + ... + u_n v_n$. Maticově skalární součin vyjádříme jako $u \cdot v = u^{\mathrm{T}} v = v^{\mathrm{T}} u$.

Úlohu minimalizace kritéria nejmenších čtverců můžeme nyní řešit také jinak a to metodami lineární algebry. Platí, že vektor φ, který hledáme, je ortogonální projekcí vektoru y do příslušného podprostoru, jak říká následující tvrzení. Obdobné tvrzení je možné nalézt v (Bečvář, 2002).

Tvrzení 3.1 (*O nejlepší aproximaci*). Nechť W je konečně dimenzionální podprostor prostoru V s pevně zvoleným skalárním součinem. Ortogonální projekce φ vektoru $y \in W$ na podprostor W je *nelepší aproximací* tohoto vektoru v podprostoru W, tj. norma vektoru $\varphi - y$ je menší než norma vektoru $\bar{\varphi} - y$ pro každé $\bar{\varphi} \in W$, $\bar{\varphi} \neq \varphi$, zapsáno symbolicky

$$\forall \bar{\varphi} \in W,\ \bar{\varphi} \neq \varphi,\ \|\varphi - y\| < \|\bar{\varphi} - y\| . \blacksquare$$

Důkaz: Jestliže je $\bar{\varphi} \in W$ a $\bar{\varphi} \neq \varphi$, potom je $o \neq \varphi - \bar{\varphi} \in W$ a vektory $\varphi - y$ a $\varphi - \bar{\varphi}$ jsou *ortogonální*, tj. jejich skalární součin je roven nule. Z Pythagorovy věty plyne

$$\|\bar{\varphi} - y\|^2 = \|\varphi - y\|^2 + \|\varphi - \bar{\varphi}\|^2 > \|\varphi - y\|^2$$
$$\|\bar{\varphi} - y\| > \|\varphi - y\| . \blacksquare$$

Pokud je tedy vektor φ kolmým průmětem, splňuje soustavu rovnic

(3.11) $$(\varphi - y) \perp \varphi_j ,\ j = 0,1,...,k .$$

Po úpravě dostaneme

(3.12) $$(\varphi - y)\cdot\varphi_j = 0 ,\ j = 0,1,...,k .$$

Další úpravou a dosazením za φ získáme soustavu $k+1$ normálních rovnic pro neznámé $a_0, a_1, ..., a_k$, tedy

(3.13) $$\sum_{j=0}^{k} a_p \left(\varphi_p \cdot \varphi_j\right) = y \cdot \varphi_j ,$$

kde $j = 0,1,...,k$. Jsou-li vektory φ_p, $p = 0,1,...,k$ lineárně nezávislé, má soustava (3.13) jednoznačné řešení a tedy existuje právě jedna aproximace metodou nejmenších čtverců danými hodnotami. Důkaz lze najít např. v (Segethová, 1998).

K soustavě (3.8) lze dojít také alternativní úvahou. Dosadíme-li do předpisu (3.2) hodnoty jednotlivých složek bodů $\{[x_i, y_i]\}_{i=0}^{n}$, dostáváme *přeurčenou* soustavu $n+1$ rovnic, tj. soustava rovnic, která obsahuje více (určujících) rovnic než neznámých

$$\begin{aligned}
&\varphi(x_0) = y_0 = a_0\varphi_0(x_0) + a_1\varphi_1(x_0) + \ldots + a_k\varphi_k(x_0),\\
&\varphi(x_1) = y_1 = a_0\varphi_0(x_1) + a_1\varphi_1(x_1) + \ldots + a_k\varphi_k(x_1),\\
&\vdots\\
&\varphi(x_n) = y_n = a_0\varphi_0(x_n) + a_1\varphi_1(x_n) + \ldots + a_k\varphi_k(x_n),
\end{aligned} \tag{3.14}$$

zapišme ji jako $Bx = y$, tedy

$$B = \begin{pmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_k(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_k(x_1) \\ \vdots & \vdots & & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_k(x_n) \end{pmatrix},\ y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix},\ x = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix}. \tag{3.15}$$

Jsou-li sloupce matice B lineárně nezávislé, má soustava (3.15) právě jedno řešení. Soustavu normálních rovnic (3.8) získáme ze soustavy (3.15) vynásobením obou stran rovnice zleva maticí B^{T}, tedy dostáváme $B^{\mathrm{T}}Bx = B^{\mathrm{T}}y$, kde $A = B^{\mathrm{T}}B$ a $b = B^{\mathrm{T}}y$. Vektor x neznámých koeficientů můžeme vyjádřit jako $x = \left(B^{\mathrm{T}}B\right)^{-1}B^{\mathrm{T}}y$ za předpokladu, že existuje inverzní matice $\left(B^{\mathrm{T}}B\right)^{-1}$. Tento postup je uveden například v (Jeffrey, 2010). Při vysoké přeurčenosti soustavy (3.15) se však výpočet vektoru x takto neprovádí kvůli numerické stabilitě, ale užívá se LU rozkladu, Choleského rozkladu, QR rozkladu nebo SVD rozkladu. Tyto přístupy rozebírá například (Demmel, 1997). Použité výpočetní prostředí MATLAB tyto metody implementuje. Numerickou stabilitu si demonstrujme na QR rozkladu.

Věta 3.2 (*QR rozklad*). Nechť $B \in \mathbb{R}_{m\times n}$ je matice s hodností $r(B) = n$, kde $m \geq n$, potom jednoznačně existuje ortonormální (někdy též ortogonální) matice $Q \in \mathbb{R}_{m\times n}$ ($Q^{\mathrm{T}}Q = E$) a invertibilní horní trojúhelníková matice $R \in \mathbb{R}_{n\times n}$ tak, že $B = QR$. ■

Důkaz věty vynecháme, uvádí jej například Demmel v (1997). Existuje několik postupů, jak zkonstruovat QR rozklad pro danou matici, zmiňme alespoň *Householderovy transformace* a klasický nebo modifikovaný *Gramův-Schmidtův ortogonalizační* proces (zkráceně G-S proces). Zvolený postup konstrukce QR rozkladu také ovlivňuje přesnost a rychlost výpočtu. Hlavní výhodou QR rozkladu je numerická stabilita při výpočtech kvůli použití ortonormální matice Q.

K výpočtu QR rozkladu podle věty 3.2 využijme Gramův-Schmidtův ortogonalizační proces. Návod na konstrukci QR rozkladu je zároveň ideou důkazu věty 3.2. Nechť jsou splněny předpoklady věty 3.2. Označme sloupce matice $B \in \mathbb{R}_{m\times n}$ jako b_i, tedy $B = (b_1, b_2, \ldots, b_n)$,

a sloupce matice $Q \in \mathbb{R}_{m\times n}$ jako q_i, tedy $Q = (q_1, q_2, ..., q_n)$. Na sloupce matice B aplikujme zleva doprava G-S proces (viz například (Bečvář, 2002)), tj.

$$
\begin{aligned}
q_1' &= b_1, \\
q_2' &= b_2 - \frac{b_2 \cdot q_1'}{\|q_1'\|^2} q_1', \\
q_3' &= b_3 - \frac{b_3 \cdot q_1'}{\|q_1'\|^2} q_1' - \frac{b_3 \cdot q_2'}{\|q_2'\|^2} q_2', \\
&\vdots \\
q_n' &= b_n - \frac{b_n \cdot q_1'}{\|q_1'\|^2} q_1' - \frac{b_n \cdot q_2'}{\|q_2'\|^2} q_2' - \ldots - \frac{b_n \cdot q_{n-1}'}{\|q_{n-1}'\|^2} q_{n-1}'.
\end{aligned} \tag{3.16}
$$

Dostaneme tak množinu *ortogonálních vektorů* q_i'. To znamená, že každé dva různé vektory q_i', q_j', $i \neq j$ jsou ortogonální, tedy jejich skalární součin je roven nule. Z ortogonálních vektorů q_i' získáme znormováním (tj. $q_i = q_i' / \|q_i'\|$) *ortonormální vektory* q_i, tedy sloupce matice Q. Nyní můžeme G-S proces přepsat následovně

$$
\begin{aligned}
q_1' &= b_1, \\
q_2' &= b_2 - \frac{b_2 \cdot q_1}{\|q_1'\|} q_1', \\
q_3' &= b_3 - \frac{b_3 \cdot q_1}{\|q_1'\|} q_1' - \frac{b_3 \cdot q_2}{\|q_2'\|} q_2', \\
&\vdots \\
q_n' &= b_n - \frac{b_n \cdot q_1}{\|q_1'\|} q_1' - \frac{b_n \cdot q_2}{\|q_2'\|} q_2' - \ldots - \frac{b_n \cdot q_{n-1}}{\|q_{n-1}'\|} q_{n-1}'.
\end{aligned} \tag{3.17}
$$

Každý vektor b_i (sloupec matice B) lze vyjádřit jako lineární kombinaci vektorů $q_1, q_2, ..., q_n$ s koeficienty $r_{ij} = q_i^{\mathrm{T}} b_j$. Koeficienty r_{ij} této lineární kombinace G-S proces rovněž počítá, jak je patrné z (3.17). Sloupce matice B tedy můžeme vyjádřit

$$
b_j = \sum_{i=1}^{n} r_{ij} q_i . \tag{3.18}
$$

Koeficienty r_{ij} jsou prvky matice $R \in \mathbb{R}_{n\times n}$, tedy $R = (r_{ij})$. Lze dokázat, že prvky matice R pod hlavní diagonálou jsou všechny rovny nule, tj. $r_{ij} = 0$, pro $i > j$.

Numerická stabilita G-S procesu lze dále vylepšit jeho speciální modifikací. V tomto případě již odkazujeme na literaturu (Demmel, 1997).

Použijeme-li QR rozklad při hledání aproximace funkce metodou nejmenších čtverců na matici $B \in \mathbb{R}_{(n+1)\times(k+1)}$ (předpokládejme, že má matice B lineárně nezávislé sloupce) v přeurčené soustavy rovnic $Bx = y$ z (3.15), potom lze příslušnou soustavu normálních rovnic $B^{\mathrm{T}}Bx = B^{\mathrm{T}}y$ řešit jako $Rx = Q^{\mathrm{T}}y$. Dosazení součinu QR za matici B si rozepišme

$$\begin{aligned} B^{\mathrm{T}}Bx &= B^{\mathrm{T}}y \Rightarrow \\ \Rightarrow \left(QR\right)^{\mathrm{T}}QRx &= \left(QR\right)^{\mathrm{T}}y \Rightarrow \\ \Rightarrow R^{\mathrm{T}}Q^{\mathrm{T}}QRx &= R^{\mathrm{T}}Q^{\mathrm{T}}y. \end{aligned} \tag{3.19}$$

Matice Q je ortonormální, víme tedy, že platí $Q^{\mathrm{T}}Q = E$. Po dosazení do rovnosti (3.19) získáme

$$\begin{aligned} R^{\mathrm{T}}ERx &= R^{\mathrm{T}}Q^{\mathrm{T}}y \Rightarrow \\ \Rightarrow R^{\mathrm{T}}Rx &= R^{\mathrm{T}}Q^{\mathrm{T}}y. \end{aligned} \tag{3.20}$$

Víme, že matice R je podle věty 3.2 invertibilní, tedy R^{T} je rovněž invertibilní maticí. Můžeme vynásobit obě strany rovnice (3.20) zleva maticí $\left(R^{\mathrm{T}}\right)^{-1}$ a dostaneme

$$Rx = Q^{\mathrm{T}}y. \tag{3.21}$$

Soustavu normálních rovnic (3.8) jsme převedli na soustavu (3.21) s horní trojúhelníkovou maticí R. Taková soustava lineárních rovnic je snadno řešitelná, neboť řešení získáme pouhým zpětným dosazením. Později vše ukážeme v příkladě 3.1.

Podívejme se nyní na případ, kdy funkce $\varphi(x)$ je polynomem stupně k, tj. funkce $\varphi(x)$ je lineární kombinací polynomů x^j, $j = 0,1,...,k$ s reálnými koeficienty a_j, $j = 0,1,...,k$. Předpokládáme tedy, že přesná funkce by byla též polynomem stupně k, kvůli chybám v měření však získáme pouze její aproximaci. Přepišme (3.2) následovně

$$\varphi(x) = a_0 + a_1x + a_2x^2 ... + a_kx^k = \sum_{j=0}^{k} a_jx^j \,. \tag{3.22}$$

V tomto speciálním případě má funkce z (3.3) tvar

$$H\left(a_0, a_1, ..., a_k\right) = \sum_{i=0}^{n}\left(a_0 + a_1x_i + a_2x_i^2 + ... + a_kx_i^k - y_i\right)^2 . \tag{3.23}$$

Hledání minima funkce H závislé na parametrech $a_0, a_1, ..., a_k$ řešíme stejným postupem jako v obecném případě. Po úpravě tedy opět získáme soustavu $k+1$ lineárních rovnic pro neznámé $a_0, a_1, ..., a_k$

$$a_0\sum_{i=0}^{n} x_i^j + a_1\sum_{i=0}^{n} x_ix_i^j + a_2\sum_{i=0}^{n} x_i^2x_i^j + ... + a_k\sum_{i=0}^{n} x_i^kx_i^j = \sum_{i=0}^{n} y_ix_i^j \,, \tag{3.24}$$

pro $j = 0,1,...,k$. Obdobně bychom mohli soustavu lineárních rovnic rozepsat jako v případě (3.8).

Poznámka 3.2: Matice koeficientů soustavy lineárních rovnic (3.24) pro polynomiální funkci $\varphi(x)$ bývá často *špatně podmíněná*, tj. při klasickém výpočtu kvůli zaokrouhlování vzniká numerická chyba, obzvláště při volbě polynomů vyšších stupňů. Pro malé hodnoty k, řekněme do 5 nebo 6, zkušenost ukazuje, že řešením soustavy (3.24) získáme poměrně dobré aproximace. Avšak čím větší hodnoty k volíme, tím horší aproximace získáváme řešením soustavy (3.24). A to platí dokonce nezávisle na zvoleném postupu řešení příslušné soustavy rovnic, (Ralston, 1965). Důsledkem špatné podmíněnosti matice je, že při výpočtu soustavy (3.24) libovolnou metodou způsobí každá zaokrouhlovací chyba, které se dopustíme, mnohonásobně větší chybu v řešení. Pro přesnější řešení soustavy lineárních rovnic, jejichž matice koeficientů je špatně podmíněná, musíme počítat s velkým počtem desetinných míst. Kvůli těmto problémům je vhodné používat polynomiální funkci pouze v těch případech, kdy je hodnota k velmi malá. V našem případě však používáme nejčastěji polynom stupně jedna.

Jedna z možností, jak vylepšit přesnost řešení je aproximace pomocí ortogonálních polynomů místo výpočtu s mocninami x jako ve (3.22), kde je funkce $\varphi(x)$ lineární kombinací polynomů x^j, $j=0,1,...,k$ s reálnými koeficienty a_j, $j=0,1,...,k$. V případě užití ortogonálních polynomů je matice soustavy normálních rovnic diagonální (mimo hlavní diagonálu má samé nulové prvky). Tento postup ukazuje například Ralston v (1965). ■

Otázkou zůstává, jak zvolit hodnotu k, tedy stupeň aproximujícího polynomu, známe-li pouze n počet bodů, které máme aproximovat, a víme, že přesná funkce je polynom stupně $< n$, ale není předem známo, jaký tento stupeň je. Kdybychom stupeň polynomu znali, řekněme, že by to byla hodnota K, a aproximovali bychom hodnoty polynomem stupně $K+1$, koeficient a_{K+1} u nejvyšší mocniny tohoto polynomu by byl statisticky roven nule. Pokud bychom měli přesnou funkci, tj. měření by nebylo zatíženo chybami, koeficient a_{K+1} by byl roven nule, v důsledku chyb však roven nule není. Základ metody pro nalezení K je tedy následující. Při výpočtu normálních rovnic (3.24) začneme s $K=1$ a postupně budeme tuto hodnotu zvyšovat. Pokud je poslední koeficient a_K statisticky nenulový, hodnotu K zvýšíme o jedna, pokud je statisticky roven nule, našli jsme stupeň polynomu, který je o jedna menší, tedy $K-1$. Problémem zůstává, jak otestovat, zda se koeficient statisticky rovná nule. Protože je měření zatíženo chybami, musíme použít robustnější metodu. Aplikujeme testování statistické hypotézy, že $a_K = 0$. Učiníme ještě další předpoklad, že chyby $e_i = \varphi(x_i) - y_i$ mají normální rozdělení s nulovou střední hodnotou. Označme dále

$$H_K = \sum_{i=0}^{n} \left(\sum_{j=0}^{K} a_j x_i^j - y_i \right)^2 . \tag{3.25}$$

Pro hodnoty $K = 1,2,3,...$ budeme postupně počítat veličinu $H_K / (n-K)$ tak dlouho, dokud její hodnota významně klesá. Jakmile dosáhneme hodnoty K, po které již nenásleduje výrazný pokles veličiny $H_K / (n-K)$, je K hledaným stupněm aproximujícího polynomu. Tímto postupem tedy určíme stupeň polynomu a získáme tak nejlepší aproximaci metodou nejmenších čtverců.

Jakým způsobem lze určit nejlepší aproximaci je uvedeno například v (Ralston, 1965). V našem případě však předem víme, že aproximující polynom bude stupně jedna (prokládáme naměřené hodnoty přímkou), podrobněji tedy volbu stupně polynomu v obecném případě aproximace nerozebíráme a odkazujeme se zde na literaturu.

Pro lepší názornost ukažme vše na příkladě, kde budeme data aproximovat přímkou, tedy polynomem stupně jedna. Na menším počtu vstupních bodů nejdříve znázorníme princip metody nejmenších čtverců a podrobně rozepíšeme výpočet, ve druhém případě ukážeme metodu nejmenších čtverců v praktické aplikaci. V dalších experimentech budeme rovněž pracovat s prokládáním dat přímkou, proto se nyní omezujeme pouze na tyto speciální případy. Tuto základní úlohu aproximace dat uvádíme také proto, abychom ji mohli porovnat s dalšími typy prokládání dat přímkou.

Příklad 3.1: *PROKLÁDÁNÍ DAT PŘÍMKOU METODOU NEJMENŠÍCH ČTVERCŮ.* Popsanou metodu nyní ukážeme na dvou příkladech prokládání dat přímkou. Mějme dánu množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i]\}_{i=0}^{n}$ v eukleidovské rovině E_2 a předpokládejme, že byly tyto body získány měřením lineární závislosti (např. z měření fyzikálního jevu, který lineárně závisí na nějakém parametru například času). To znamená, že charakter měřené závislosti je možno s dostatečnou přesností vystihnout polynomem prvního stupně.

Na prvním případě předveďme princip metody nejmenších čtverců na menším počtu vstupních bodů. Experimentální data pro demonstraci aproximace jsou uvedena v tabulce 3.1, přičemž se nejedná o data, která by odpovídala nějakému skutečnému měření. Dále je naznačen postup výpočtu s přeurčenou soustavou rovnic. Uvedena je rovněž alternativní metoda výpočtu s QR rozkladem, k jehož konstrukci je využit Gramův-Schmidtův ortogonalizační proces popsaný v teoretické části. (Číselné hodnoty jsou při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Na obrázku 3.1 je potom znázorněna výsledná aproximující přímka, pro kterou vyžadujeme, aby součet obsahů vyznačených čtverců byl minimální (zobrazeny jsou právě ty čtverce, pro které je tento součet nejmenší možný).

Obrázek 3.2 již ukazuje praktickou úlohu, kdy body odpovídají měření lineární závislosti zatíženému chybami. Můžeme vidět, že výsledná aproximující přímka leží blízko přesné funkce a má i velmi podobnou směrnici. Zde tedy předpokládáme, že přesnou funkci známe. Konkrétní hodnoty v tomto v případě nevypisujeme, protože se jedná již o rozsáhlejší data.

Výstupy jsou získány z výpočetního prostředí MATLAB. V programech řešíme aproximaci dat přímkou a polynomem libovolného stupně metodou nejmenších čtverců pomocí odvozených vztahů, přičemž výsledná soustava normálních rovnic je vypočtena s použitím funkcí MATLAbu. K řešení problému prokládání dat polynomem metodou nejmenších čtverců lze také přímo využít funkci `polyfit`, která je k dispozici v prostředí MATLAB.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu): `mnc_primka.m` (programy/mnc2d),
`mnc_polynom.m` (programy/mnc2d).

---

x_i	-2	0	1	2	2,5	3	4	4,5	5	5,5	
y_i	-3	-4	-2	1	-2	-3	1	-2	1,5	3	
x_i	5,5	6	6	7	8	8,5	9	9	9	10	10
y_i	-1,5	1	2	4	1,5	4	1	4,5	5	4	6

Tabulka 3.1: Experimentální data pro aproximaci přímkou metodou nejmenších čtverců

$$B^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & 0 & 1 & 2 & 2{,}5 & 3 & 4 & 4{,}5 & 5 & 5{,}5 & 5{,}5 & 6 & 6 & 7 & 8 & 8{,}5 & 9 & 9 & 9 & 10 & 10 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 2{,}5 \\ 1 & 3 \\ 1 & 4 \\ 1 & 4{,}5 \\ 1 & 5 \\ 1 & 5{,}5 \\ 1 & 5{,}5 \\ 1 & 6 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 8{,}5 \\ 1 & 9 \\ 1 & 9 \\ 1 & 9 \\ 1 & 10 \\ 1 & 10 \end{pmatrix} \qquad y = \begin{pmatrix} -3 \\ -4 \\ -2 \\ 1 \\ -2 \\ -3 \\ 1 \\ -2 \\ 1{,}5 \\ 3 \\ -1{,}5 \\ 1 \\ 2 \\ 4 \\ 1{,}5 \\ 4 \\ 1 \\ 4{,}5 \\ 5 \\ 4 \\ 6 \end{pmatrix}$$

$$B^{\mathrm{T}}B = \begin{pmatrix} 21 & 113{,}5 \\ 113{,}5 & 846{,}25 \end{pmatrix} \qquad B^{\mathrm{T}}y = \begin{pmatrix} 22 \\ 289{,}25 \end{pmatrix}$$

$$\begin{pmatrix} 21 & 113{,}5 \\ 113{,}5 & 846{,}25 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 22 \\ 289{,}25 \end{pmatrix}$$

$$B^{\mathrm{T}}Bx = B^{\mathrm{T}}y \qquad x = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

řešení soustavy

$$a_0 = -2{,}90701 \qquad a_1 = 0{,}73169$$

$$y = 0{,}73169x + 2{,}90701$$

rovnice aproximující přímky

$$H(a_0, a_1) = 51{,}31186$$

QR rozklad

$$b_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad b_2 = \begin{pmatrix} -2 \\ 0 \\ 1 \\ 2 \\ 2{,}5 \\ 3 \\ 4 \\ 4{,}5 \\ 5 \\ 5{,}5 \\ 5{,}5 \\ 6 \\ 6 \\ 7 \\ 8 \\ 8{,}5 \\ 9 \\ 9 \\ 9 \\ 10 \\ 10 \end{pmatrix}$$

$$B = (b_1, b_2)$$

Gramův-Schmidtův ortogonalizační proces

$$q_1' = b_1$$

$$q_2' = b_2 - \frac{b_2 \cdot q_1'}{\|q_1'\|^2} q_1'$$

$$q_1' = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad q_2' = \begin{pmatrix} -2 \\ 0 \\ 1 \\ 2 \\ 2{,}5 \\ 3 \\ 4 \\ 4{,}5 \\ 5 \\ 5{,}5 \\ 5{,}5 \\ 6 \\ 6 \\ 7 \\ 8 \\ 8{,}5 \\ 9 \\ 9 \\ 9 \\ 10 \\ 10 \end{pmatrix} - \frac{227}{42} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -7{,}40476 \\ -5{,}40476 \\ -4{,}40476 \\ -3{,}40476 \\ -2{,}90476 \\ -2{,}40476 \\ -1{,}40476 \\ -0{,}90476 \\ 0{,}40476 \\ 0{,}09524 \\ 0{,}09524 \\ 0{,}59524 \\ 0{,}59524 \\ 1{,}59524 \\ 2{,}59524 \\ 3{,}09524 \\ 3{,}59524 \\ 3{,}59524 \\ 3{,}59524 \\ 4{,}59524 \\ 4{,}59524 \end{pmatrix}$$

Znormování vektorů

$$q_1 = q_1' / \|q_1'\|$$

$$q_2 = q_2' / \|q_2'\|$$

$$\|q_1'\| = \sqrt{21}$$

$$\|q_1'\| = \sqrt{232{,}80952}$$

$$q_1 = \begin{pmatrix} 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \\ 0{,}21822 \end{pmatrix} \quad q_2 = \begin{pmatrix} -0{,}4853 \\ -0{,}35422 \\ -0{,}28868 \\ -0{,}22314 \\ -0{,}19038 \\ -0{,}15761 \\ -0{,}09207 \\ -0{,}0593 \\ -0{,}02653 \\ 0{,}00624 \\ 0{,}00624 \\ 0{,}03901 \\ 0{,}03901 \\ 0{,}10455 \\ 0{,}17009 \\ 0{,}20286 \\ 0{,}23563 \\ 0{,}23563 \\ 0{,}23563 \\ 0{,}30117 \\ 0{,}30117 \end{pmatrix}$$

$$R = \begin{pmatrix} q_1 \cdot b_1 & q_1 \cdot b_2 \\ q_2 \cdot b_1 & q_2 \cdot b_2 \end{pmatrix} \quad Q = (q_1, q_2) \quad Q^{\mathrm{T}} = \begin{pmatrix} q_1^{\mathrm{T}} \\ q_2^{\mathrm{T}} \end{pmatrix}$$

$$\begin{pmatrix} 4{,}58258 & 24{,}76773 \\ 0 & 15{,}2581 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 4{,}80079 \\ 11{,}16425 \end{pmatrix}$$

$$x = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

$$Rx = Q^{\mathrm{T}} y$$

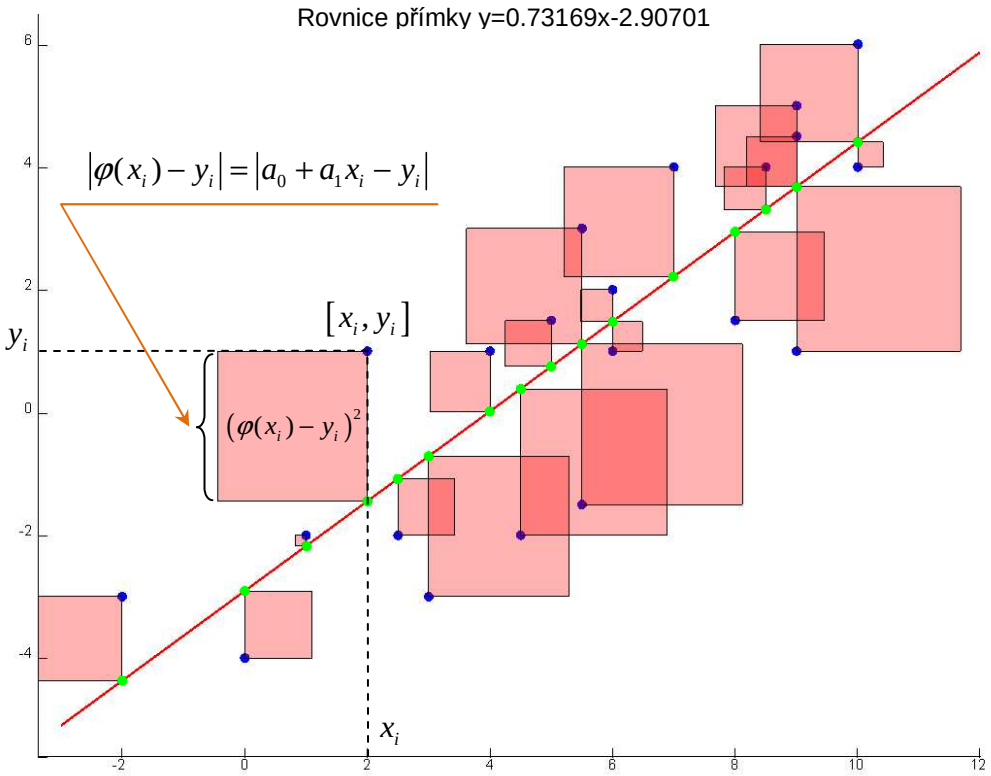
řešení soustavy

$$a_0 = -2{,}90701 \qquad a_1 = 0{,}73169$$

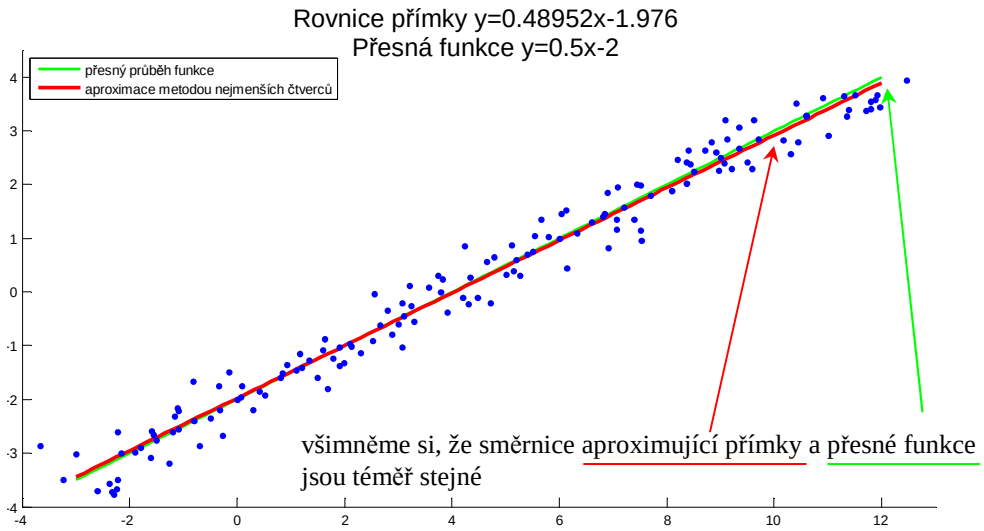
$$H(a_0, a_1) = 51{,}31186$$

$$y = 0{,}73169x + 2{,}90701$$

rovnice aproximující přímky



Obrázek 3.1: Prokládání dat metodou nejmenších čtverců – princip metody



Obrázek 3.2: Prokládání dat metodou nejmenších čtverců – praktická úloha, aproximující přímka červeně, přesný průběh funkce zeleně

Klasická metoda nejmenších čtverců představuje jednu z nejpoužívanějších a nejznámějších typů aproximací funkcí objevující se v celé řadě aplikací. Je však zřejmé, že s jejím použitím jsme kvůli funkční závislosti značně omezeni. Proto zavedeme další obecnější přístupy, které s touto metodou porovnáme.

3.3 Ortogonální prokládání dat přímkou

Metoda ortogonálního prokládání dat v rovině nebo v prostoru přímkou (*orthogonal line fitting*) je sice méně známá, na vysokých školách v ČR se v základních kurzech numerické matematiky většinou nevyučuje, ale je mnohem více uplatnitelná v praktických aplikacích.

Následující odvození lze rozšířit do libovolné dimenze, s ohledem na naše aplikace se omezíme pouze na dimenzi dvě a tři. Předpokládejme, že je dána množina $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3 (případně množina $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i]\}_{i=0}^{n}$ v eukleidovské rovině E_2), značme tyto množiny $\mathcal{X} = \{X_i\}_{i=0}^{n}$ pro oba případy. Souřadnice bodů X_i budeme někdy označovat též $\{[X_i^x, X_i^y, X_i^z]\}_{i=0}^{n}$ (případně $\{[X_i^x, X_i^y]\}_{i=0}^{n}$).

Hledáme nyní přímku, která bude dobře aproximovat zadané body, přičemž volíme přirozené následující geometrické kritérium. Budeme minimalizovat součet druhých mocnin vzdáleností bodů $\{X_i\}_{i=0}^{n}$ od hledané přímky. Hledanou přímku q vyjádříme parametricky

$$q(t) = A + ut\,, \tag{3.26}$$

kde $t \in \mathbb{R}$, A je bod přímky a u je její jednotkový směrový vektor, tj. $\|u\| = 1$. Pro libovolný bod X_i platí

(3.27) $$X_i - A = d_i u + p_i u^{\perp},$$

kde $d_i = u \cdot (X_i - A) = u^{\mathrm{T}} (X_i - A)$ je skalární součin vektorů u a $X_i - A$, p_i je reálný koeficient a $u^{\perp}$ je libovolný jednotkový vektor kolmý na vektor u. Označme dále vektor $w_i = X_i - A$. Z rovnosti $u \cdot w_i = \|u\| \|w_i\| \cos\varphi$, kde $\varphi \in \langle 0, \pi \rangle$ je úhel, který vektory u a w_i svírají, plyne, že $d_i = u \cdot w_i = \|w_i\| \cos\varphi$, neboť $\|u\| = 1$. Vektor $X_i - X_i'$, kde bod X_i' je ortogonální průmět bodu X_i na přímku q, lze zapsat následovně

(3.28) $$X_i - X_i' = w_i - d_i u = p_i u^{\perp}.$$

Velikost vektoru (3.28) je ortogonální vzdálenost bodu X_i od přímky q, podívejme se na obrázek 3.3, ze kterého je odvozený vztah dobře pochopitelný.

Při hledání přímky q budeme opět postupovat na základě metody nejmenších čtverců, nyní ale funkci, označme ji σ^2, jejíž minimum hledáme, definujeme jako součet druhých mocnin vzdáleností daných bodů $\{X_i\}_{i=0}^{n}$ od přímky q, tj.

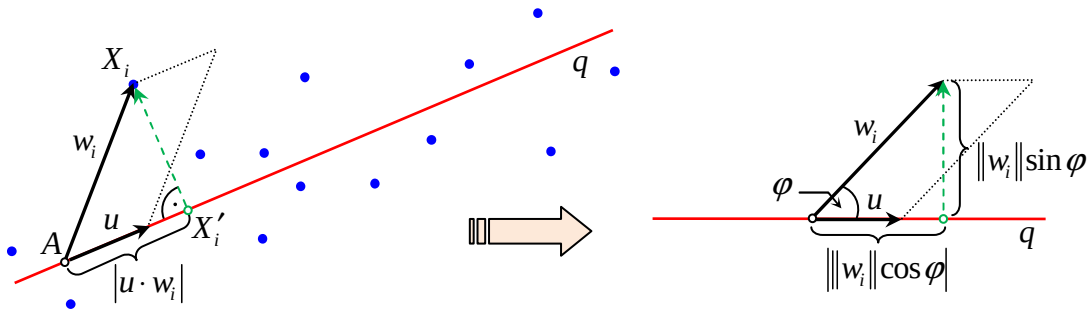
(3.29) $$\sigma^2 = \sum_{i=0}^{n} \|w_i - d_i u\|^2 = \sum_{i=0}^{n} p_i^2 .$$

Lze také odvodit, že vztah (3.29) je možné zapsat následovně

(3.30) $$\sigma^2 = \sum_{i=0}^{n} \|u \times w_i\|^2 ,$$

neboť víme, že platí rovnost $\|u \times w_i\| = \|u\| \|w_i\| \sin\varphi$ a jelikož je $\|u\| = 1$, dostáváme $\|u \times w_i\| = \|w_i\| \sin\varphi$, což je výška v rovnoběžníku určeného vektory u a w_i, tedy vzdálenost bodu X_i od přímky q. Symbol $\times$ značí vektorový součin. Pro vektory $u = (u^x, u^y, u^z)^{\mathrm{T}}$ a $v = (v^x, v^y, v^z)^{\mathrm{T}}$ je vektorový součin vektor o souřadnicích

(3.31) $$u \times v = \left(\begin{vmatrix} u^y & u^z \\ v^y & v^z \end{vmatrix}, -\begin{vmatrix} u^x & u^z \\ v^x & v^z \end{vmatrix}, \begin{vmatrix} u^x & u^y \\ v^x & v^y \end{vmatrix} \right)^{\mathrm{T}} .$$



Obrázek 3.3: Geometrický význam použitých vektorů a vzdáleností při ortogonálním prokládání dat v rovině přímkou

Pro situaci v eukleidovské rovině E_2 volíme třetí souřadnici vektorů u a v nulovou. Geometrický význam popsaných vztahů je objasněn na obrázku 3.3.

Nyní hledáme minimum funkce σ^2 z (3.29) závislé na parametrech A a u (bod a směrový vektor přímky q). Začněme s výpočtem souřadnic bodu A. Využijeme k tomu standardní postup, tj. potřebujeme spočítat parciální derivaci funkce σ^2 podle bodu A. Pro zjednodušení výpočtu vyjádříme funkci σ^2 v alternativním tvaru

$$\sigma^2 = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(E - uu^{\mathrm{T}} \right) w_i \right]. \tag{3.32}$$

Výraz (3.29) upravme, abychom ukázali, že (3.32) je skutečně ekvivalentní zápis stejné funkce σ^2, tj.

$$\begin{aligned} \sigma^2 &= \sum_{i=0}^{n} \left\| w_i - d_i u \right\|^2 = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} w_i - 2 d_i u^{\mathrm{T}} w_i + d_i^2 \left\| u \right\|^2 \right) = \\ &= \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} w_i - d_i u^{\mathrm{T}} w_i - d_i u^{\mathrm{T}} w_i + d_i^2 \right) = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} w_i - d_i u^{\mathrm{T}} w_i - d_i^2 + d_i^2 \right) = \\ &= \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} w_i - d_i u^{\mathrm{T}} w_i \right) = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} w_i - w_i^{\mathrm{T}} u u^{\mathrm{T}} w_i \right) = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(w_i - u u^{\mathrm{T}} w_i \right) \right] = \\ &= \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(E - u u^{\mathrm{T}} \right) w_i \right]. \end{aligned} \tag{3.33}$$

Pro lepší názornost tvar funkce σ^2 z (3.32) rozepišme po souřadnicích pro případ v eukleidovském prostoru E_3, přičemž horní index u vektoru značí *x*-ovou, *y*-ovou nebo *z*-ovou souřadnici, tj. $w_i = (w_i^x, w_i^y, w_i^z)^{\mathrm{T}}$ a $u = (u^x, u^y, u^z)^{\mathrm{T}}$. Uvádíme také, s jakými typy matic pracujeme.

$$\begin{aligned} \sigma^2 &= \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(E - u u^{\mathrm{T}} \right) w_i \right] = \\ &= \sum_{i=0}^{n} \left[\begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} u^x \\ u^y \\ u^z \end{pmatrix} \begin{pmatrix} u^x & u^y & u^z \end{pmatrix} \right) \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right], \end{aligned} \tag{3.34}$$

kde jeden člen sumy představuje součin matic typů $(1\times3)\left[(3\times3)-(3\times1)(1\times3)\right](3\times1)$.

Čtvercovou matici $E - uu^{\mathrm{T}}$ z (3.32) a (3.34) označme jako U a prvky této matice u_{kl}, tedy $U = (u_{kl})$. Vyjádřeme prvky matice U opět pro případ v eukleidovském prostoru E_3

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} = \begin{pmatrix} 1-(u^x)^2 & -u^x u^y & -u^x u^z \\ -u^x u^y & 1-(u^y)^2 & -u^y u^z \\ -u^x u^z & -u^y u^z & 1-(u^z)^2 \end{pmatrix}. \tag{3.35}$$

Je zřejmé, že matice U je symetrická, neboť $U = U^{\mathrm{T}}$, tj. pro každé $k,l = 1,2,3$ platí $u_{kl} = u_{lk}$.

S použitím zavedeného značení přepišme tvar funkce σ^2 z (3.34) a postupně rozná-sobme jednotlivé matice uvnitř sumy

$$\sigma^2 = \sum_{i=0}^{n}\left[w_i^{\mathrm{T}} U w_i \right] = \sum_{i=0}^{n}\left[\begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right] =$$

$$\sum_{i=0}^{n}\left[\left(w_i^x u_{11} + w_i^y u_{21} + w_i^z u_{31}, w_i^x u_{12} + w_i^y u_{22} + w_i^z u_{32}, w_i^x u_{13} + w_i^y u_{23} + w_i^z u_{33} \right) \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right] =$$

(3.36)
$$\begin{aligned} &= \sum_{i=0}^{n}\Big[(w_i^x)^2 u_{11} + w_i^x w_i^y u_{21} + w_i^x w_i^z u_{31} + \\ &+ w_i^x w_i^y u_{12} + (w_i^y)^2 u_{22} + w_i^y w_i^z u_{32} + \\ &+ w_i^x w_i^z u_{13} + w_i^y w_i^z u_{23} + (w_i^z)^2 u_{33} \Big]. \end{aligned}$$

Dostáváme tak hodnotu funkce σ^2 vyjádřenou v závislosti na souřadnicích vstupních parametrů.

Přejděme nyní k výpočtu parciální derivace funkce σ^2 podle bodu A a využijme k tomu tvar funkce σ^2 z (3.32). Parciální derivaci funkce σ^2 podle bodu A definujeme jako vektor parciálních derivací funkce σ^2 podle jednotlivých souřadnic (pracujeme stále v eukleidovském prostoru E_3) a zapisujeme následovně

(3.37)
$$\frac{\partial \sigma^2}{\partial A} = \left(\frac{\partial \sigma^2}{\partial A^x}, \frac{\partial \sigma^2}{\partial A^y}, \frac{\partial \sigma^2}{\partial A^z} \right)^{\mathrm{T}}.$$

Víme, že $w_i = X_i - A$, vyjádřeno v souřadnicích $w_i = (X_i^x - A^x, X_i^y - A^y, X_i^z - A^z)^{\mathrm{T}}$. Funkce σ^2, jak je vidět z výsledného tvaru ve (3.36), je složená funkce, pro složky parciální derivace funkce σ^2 podle bodu A tedy platí následující

(3.38)
$$\frac{\partial \sigma^2}{\partial A^x} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^x} \frac{\partial w_i^x}{\partial A^x}, \quad \frac{\partial \sigma^2}{\partial A^y} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^y} \frac{\partial w_i^y}{\partial A^y}, \quad \frac{\partial \sigma^2}{\partial A^z} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^z} \frac{\partial w_i^z}{\partial A^z}.$$

Spočtěme parciální derivaci funkce σ^2 podle bodu A a rozepišme výpočty pro jednotlivé složky, vycházejme přitom z tvaru (3.36), kde jsme vše rozepsali po souřadnicích.

Pro *x*-ovou souřadnici parciální derivace funkce σ^2 podle bodu A dostáváme

$$\frac{\partial \sigma^2}{\partial w_i^x} = \sum_{i=0}^{n} \left(2w_i^x u_{11} + w_i^y u_{21} + w_i^z u_{31} + w_i^y u_{12} + w_i^z u_{13} \right), \frac{\partial w_i^x}{\partial A^x} = -1$$

$$\Downarrow \tag{3.39}$$

$$\frac{\partial \sigma^2}{\partial A^x} = -\sum_{i=0}^{n} \left(2w_i^x u_{11} + w_i^y u_{21} + w_i^z u_{31} + w_i^y u_{12} + w_i^z u_{13} \right).$$

Důležitý poznatek, který je třeba si nyní uvědomit je, že matice U je symetrická, tedy pro každé $k,l = 1,2,3$ platí $u_{kl} = u_{lk}$. Můžeme tedy *x*-souřadnici parciální derivace funkce σ^2 podle bodu A zjednodušit na

$$\frac{\partial \sigma^2}{\partial A^x} = -2\sum_{i=0}^{n} \left(w_i^x u_{11} + w_i^y u_{12} + w_i^z u_{13} \right), \tag{3.40}$$

což lze ještě přepsat na tvar

$$\frac{\partial \sigma^2}{\partial A^x} = -2\left(u_{11}, u_{12}, u_{13} \right) \sum_{i=0}^{n} \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix}, \tag{3.41}$$

tj. skalární součin prvního řádku matice U a vektoru, který vznikne jako součet všech vektorů w_i pro $i = 0,1,...,n$, vynásobený konstantou -2.

Analogicky budeme postupovat pro *y*-ovou a *z*-ovou souřadnici parciální derivace funkce σ^2 podle bodu A, tj.

$$\frac{\partial \sigma^2}{\partial w_i^y} = \sum_{i=0}^{n} \left(w_i^x u_{21} + w_i^x u_{12} + 2w_i^y u_{22} + w_i^z u_{32} + w_i^z u_{23} \right), \frac{\partial w_i^y}{\partial A^y} = -1$$

$$\Downarrow \tag{3.42}$$

$$\frac{\partial \sigma^2}{\partial A^y} = -\sum_{i=0}^{n} \left(w_i^x u_{21} + w_i^x u_{12} + 2w_i^y u_{22} + w_i^z u_{32} + w_i^z u_{23} \right)$$

a

$$\frac{\partial \sigma^2}{\partial w_i^z} = \sum_{i=0}^{n} \left(w_i^x u_{31} + w_i^y u_{32} + w_i^x u_{13} + w_i^y u_{23} + 2w_i^z u_{33} \right), \frac{\partial w_i^z}{\partial A^z} = -1$$

$$\Downarrow \tag{3.43}$$

$$\frac{\partial \sigma^2}{\partial A^z} = -\sum_{i=0}^{n} \left(w_i^x u_{31} + w_i^y u_{32} + w_i^x u_{13} + w_i^y u_{23} + 2w_i^z u_{33} \right).$$

Opět využijeme toho, že je matice U je symetrická a přepíšeme (3.42) a (3.43)

(3.44) $$\frac{\partial \sigma^2}{\partial A^y} = -2\sum_{i=0}^{n}\left(w_i^x u_{21} + w_i^y u_{22} + w_i^z u_{23}\right)$$

a

(3.45) $$\frac{\partial \sigma^2}{\partial A^z} = -2\sum_{i=0}^{n}\left(w_i^x u_{31} + w_i^y u_{32} + w_i^z u_{33}\right).$$

Obě parciální derivace (3.44) a (3.45) můžeme opět vyjádřit pomocí skalárního součinu příslušného řádku matice U a vektoru, který vznikne jako součet všech vektorů w_i pro $i = 0,1,...,n$, vynásobený konstantou -2, tj.

(3.46) $$\frac{\partial \sigma^2}{\partial A^y} = -2\left(u_{21}, u_{22}, u_{23}\right)\sum_{i=0}^{n}\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix}$$

a

(3.47) $$\frac{\partial \sigma^2}{\partial A^z} = -2\left(u_{31}, u_{32}, u_{33}\right)\sum_{i=0}^{n}\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix}.$$

Symbolicky můžeme zapsat parciální derivaci funkce σ^2 podle bodu A následovně

(3.48) $$\frac{\partial \sigma^2}{\partial A} = -2\left(E - uu^{\mathrm{T}}\right)\sum_{i=0}^{n} w_i ,$$

kde $E - uu^{\mathrm{T}} = U$.

Je zřejmé, že parciální derivace funkce σ^2 podle bodu A z (3.48) je rovna nulovému vektoru právě tehdy, když

(3.49) $$\sum_{i=0}^{n} w_i = o ,$$

to samé zapsáno po souřadnicích

(3.50) $$\sum_{i=0}^{n} w_i^x = 0 , \ \sum_{i=0}^{n} w_i^y = 0 , \ \sum_{i=0}^{n} w_i^z = 0 .$$

Dosadíme-li do výrazu (3.49) za $w_i = X_i - A$, dostáváme vzorec pro výpočet bodu A

(3.51) $$A = \frac{1}{n+1}\sum_{i=0}^{n} X_i .$$

Bod A tedy spočítáme jako aritmetický průměr bodů $\{X_i\}_{i=0}^n$. Rozepišme si výsledek (3.51) opět po souřadnicích

$$A^x = \frac{1}{n+1}\sum_{i=0}^{n} X_i^x,\ A^y = \frac{1}{n+1}\sum_{i=0}^{n} X_i^y,\ A^z = \frac{1}{n+1}\sum_{i=0}^{n} X_i^z. \tag{3.52}$$

Veškeré výpočty se souřadnicemi jsme uváděli pro případ v eukleidovském prostoru E_3, případ v eukleidovské rovině E_2 je analogický a přepis jednotlivých vzorců je jednoduchý.

Posuňme se ve výpočtech dále. Bod A hledané přímky q jsme určili, nyní zbývá spočítat vektor u, což je jednotkový směrový vektor přímky q. K tomu použijeme další alternativní přepis funkce σ^2 a to

$$\sigma^2 = u^{\mathrm{T}}\left(\sum_{i=0}^{n}\left[(w_i \cdot w_i)E - w_i w_i^{\mathrm{T}}\right]\right)u = u^{\mathrm{T}}Mu, \tag{3.53}$$

kde $w_i \cdot w_i$ je skalární součin vektorů w_i a w_i, E je jednotková matice. Čtvercová matice M značí sumu v závorce.

Dokažme nejdříve, že výraz (3.53) je opravdu ekvivalentní zápis stejné funkce σ^2 definované v (3.29). Z úprav uvedených v (3.33), víme, že funkce σ^2 lze přepsat na

$$\sigma^2 = \sum_{i=0}^{n}\left\|w_i - d_i u\right\|^2 = \sum_{i=0}^{n}\left(w_i^{\mathrm{T}}w_i - d_i u^{\mathrm{T}}w_i\right). \tag{3.54}$$

Tento tvar dále upravíme a dostáváme

$$\begin{aligned}
\sigma^2 &= \sum_{i=0}^{n}\left(w_i^{\mathrm{T}}w_i - u^{\mathrm{T}}w_i w_i^{\mathrm{T}}u\right) = \sum_{i=0}^{n}\left[(w_i \cdot w_i)u^{\mathrm{T}}u - u^{\mathrm{T}}w_i w_i^{\mathrm{T}}u\right] = \\
&\sum_{i=0}^{n}\left[u^{\mathrm{T}}(w_i \cdot w_i)u - u^{\mathrm{T}}w_i w_i^{\mathrm{T}}u\right] = u^{\mathrm{T}}\sum_{i=0}^{n}\left[(w_i \cdot w_i)u - w_i w_i^{\mathrm{T}}u\right] = \\
&= u^{\mathrm{T}}\left(\sum_{i=0}^{n}\left[(w_i \cdot w_i)E - w_i w_i^{\mathrm{T}}\right]\right)u = \\
&= u^{\mathrm{T}}Mu.
\end{aligned} \tag{3.55}$$

Tím jsme ukázali, že (3.53) je skutečně pouze jiným zápisem stejné funkce σ^2 z (3.29).

Pro lepší názornost tvar funkce σ^2 z (3.55) opět rozepišme po souřadnicích pro případ v eukleidovském prostoru E_3. Jeden člen sumy představuje matici, která vznikne jako rozdíl matic typů (3×3), zleva je matice vynásobená řádkovým vektorem (1×3) a zprava sloupcovým vektorem (3×1), tj.

$$\sigma^2 = u^{\mathrm{T}} \sum_{i=0}^{n} \left[(w_i \cdot w_i) E - w_i w_i^{\mathrm{T}} \right] u =$$

$$= \begin{pmatrix} u^x & u^y & u^z \end{pmatrix} \left(\sum_{i=0}^{n} \left[\left((w_i^x)^2 + (w_i^y)^2 + (w_i^z)^2 \right) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \right] \right) \begin{pmatrix} u^x \\ u^y \\ u^z \end{pmatrix} = \tag{3.56}$$

$$= u^{\mathrm{T}} M u .$$

Rozepišme dále, jak vypadá čtvercová matice M a spočtěme její prvky

$$M = \sum_{i=0}^{n} \left[\begin{pmatrix} (w_i^x)^2 + (w_i^y)^2 + (w_i^z)^2 & 0 & 0 \\ 0 & (w_i^x)^2 + (w_i^y)^2 + (w_i^z)^2 & 0 \\ 0 & 0 & (w_i^x)^2 + (w_i^y)^2 + (w_i^z)^2 \end{pmatrix} - \right.$$

$$\left. - \begin{pmatrix} (w_i^x)^2 & w_i^x w_i^y & w_i^x w_i^z \\ w_i^x w_i^y & (w_i^y)^2 & w_i^y w_i^z \\ w_i^x w_i^z & w_i^y w_i^z & (w_i^z)^2 \end{pmatrix} \right] = \sum_{i=0}^{n} \begin{pmatrix} (w_i^y)^2 + (w_i^z)^2 & -w_i^x w_i^y & -w_i^x w_i^z \\ -w_i^x w_i^y & (w_i^x)^2 + (w_i^z)^2 & w_i^y w_i^z \\ -w_i^x w_i^z & w_i^y w_i^z & (w_i^x)^2 + (w_i^y)^2 \end{pmatrix} = \tag{3.57}$$

$$= \begin{pmatrix} \sum_{i=0}^{n} \left[(w_i^y)^2 + (w_i^z)^2 \right] & -\sum_{i=0}^{n} w_i^x w_i^y & -\sum_{i=0}^{n} w_i^x w_i^z \\ -\sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} \left[(w_i^x)^2 + (w_i^z)^2 \right] & -\sum_{i=0}^{n} w_i^y w_i^z \\ -\sum_{i=0}^{n} w_i^x w_i^z & -\sum_{i=0}^{n} w_i^y w_i^z & \sum_{i=0}^{n} \left[(w_i^x)^2 + (w_i^y)^2 \right] \end{pmatrix}.$$

Označme prvky této matice m_{kl}, tedy $M = (m_{kl})$. Je zřejmé, že matice M je symetrická, neboť pro každé $k, l = 1,2,3$ platí $m_{kl} = m_{lk}$.

Počítáme-li ortogonální prokládání dat přímkou ve výpočetním prostředí MATLAB, používáme jiný tvar matice M. Nevyjadřujeme jednotlivé prvky matice M, neboť to je pro čtenáře přehlednější, ale ponecháváme matici M v tomto tvaru

$$M = \left[\sum_{i=0}^{n} (w_i^x)^2 + \sum_{i=0}^{n} (w_i^y)^2 + \sum_{i=0}^{n} (w_i^z)^2 \right] \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \sum_{i=0}^{n} (w_i^x)^2 & \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} w_i^x w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} (w_i^y)^2 & \sum_{i=0}^{n} w_i^y w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^z & \sum_{i=0}^{n} w_i^y w_i^z & \sum_{i=0}^{n} (w_i^z)^2 \end{pmatrix}. \tag{3.58}$$

Opět vidíme, jak by se veškeré výpočty se souřadnicemi změnily pro případ v eukleidovské rovině E_2.

Určeme nyní jednotkový směrový vektor u hledané přímky q. Pro dané A spočítáme vektor u jako *vlastní vektor* (Dym, 2007) příslušný nejmenšímu *vlastnímu číslu* (Dym, 2007) matice M. Zobrazení $\sigma^2 = u^{\mathrm{T}} M u$ z (3.53) je kvadratická forma, jejíž minimum hledáme. Máme-li totiž libovolnou čtvercovou matici $C \in \mathbb{R}_{n\times n}$, tedy $C = (c_{ij})$, potom zobrazení f, které vektoru $v = (v_1, v_2, ..., v_n)^{\mathrm{T}} \in \mathbb{R}_n$ přiřadí skalár

$$f(v) = \sum_{i,j=1}^{n} c_{ij} v_i v_j = (v_1, v_2, ..., v_n) C \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = v^{\mathrm{T}} C v, \tag{3.59}$$

je kvadratická forma na prostoru $\mathbb{R}_n$.

Hledání nejmenšího vlastního čísla matice M a jemu příslušného vlastního vektoru je správným postupem, neboť platí, že minimum kvadratické formy $\sigma^2 = u^{\mathrm{T}} M u$ odpovídá nejmenšímu vlastnímu číslu matice M. Uveďme následující větu, viz například (Dym, 2007), která nám celou situaci objasní.

Věta 3.3 (*Courant-Fisher*). Nechť $\lambda_1 \le \lambda_2 \le ... \le \lambda_n$ jsou vlastní čísla symetrické matice $C \in \mathbb{R}_{n\times n}$. Potom

$$\lambda_1 = \min_{v:\|v\|=1} v^{\mathrm{T}} C v,\ \lambda_n = \max_{v:\|v\|=1} v^{\mathrm{T}} C v\ .\ \blacksquare$$

K důkazu věty 3.3 budeme potřebovat ještě větu následující, její znění a důkaz je možné nalézt například v (Poole, 2006).

Věta 3.4 (*Spektrální rozklad symetrické matice*). Pro každou symetrickou matici $C \in \mathbb{R}_{n\times n}$ existuje ortonormální matice $Q \in \mathbb{R}_{n\times n}$ ($Q^{\mathrm{T}} Q = E$) a diagonální matice $\Lambda \in \mathbb{R}_{n\times n}$ (mimo hlavní diagonálu má samé nulové prvky) tak, že $C = Q \Lambda Q^{\mathrm{T}}$. $\blacksquare$

Diagonální matici $\Lambda \in \mathbb{R}_{n\times n}$ z věty 3.4 značme $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$, kde $\lambda_i, i = 1, 2, ..., n$ jsou prvky na diagonále. Je zřejmé, že vlastní čísla diagonální matice jsou právě prvky její diagonály. Lze ukázat, že prvky na diagonále λ_i, $i = 1, 2, ..., n$ matice Λ jsou rovněž všechna vlastní čísla matice C. Sloupce ortonormální matice Q z věty 3.4 jsou ortonormální vlastní vektory matice C příslušné vlastním číslům λ_i, $i = 1, 2, ..., n$. To lze ukázat tak, že rozepíšeme jednotlivé matice, z čehož vyplynou popsané závislosti. Případně odkazujeme čtenáře na literaturu (Abadir a Magnus, 2005) nebo (Poole, 2006), kde jsou tvrzení podrobně dokázána a připojeny jsou rovněž názorné příklady.

Přejděme nyní k důkazu věty 3.3.

Důkaz věty 3.3: Buď $s_n \in \mathbb{R}_n$ normovaný vlastní vektor, tedy $\|s_n\| = 1$, matice C příslušný největšímu vlastnímu číslu λ_n. Podle definice vlastního vektoru, který přísluší danému vlastnímu číslu matice C, platí $C s_n = \lambda_n s_n$. Následující úpravou dostáváme důkaz nerovnosti $\lambda_n \le \max_{v:\|v\|=1} v^{\mathrm{T}} C v$, tedy

$$s_n^{\mathrm{T}} C s_n = s_n^{\mathrm{T}} \lambda_n s_n$$
$$s_n^{\mathrm{T}} C s_n = \lambda_n s_n^{\mathrm{T}} s_n$$
$$(s_n^{\mathrm{T}} s_n = \|s_n\|^2 = 1)$$
$$\lambda_n = s_n^{\mathrm{T}} C s_n \le \max_{v:\|v\|=1} v^{\mathrm{T}} C v.$$

Pokračujme důkazem druhé nerovnosti

$$\lambda_n \ge \max_{v:\|v\|=1} v^{\mathrm{T}} C v .$$

Buď $v \in \mathbb{R}_n$ libovolný normovaný vektor, tedy $\|v\| = 1$. Pro důkaz druhé nerovnosti použijeme spektrální rozklad symetrické matice C podle věty 3.4, tedy $C = Q \Lambda Q^{\mathrm{T}}$, kde $Q \in \mathbb{R}_{n \times n}$ je ortonormální matice a $\Lambda \in \mathbb{R}_{n \times n}$ diagonální matice. Dále víme, že prvky na diagonále $\lambda_i, i = 1,2,...,n$ matice Λ jsou rovněž všechna vlastní čísla matice C. Označme vektor $y = Q^{\mathrm{T}} v$, $y = (y_1, y_2, ..., y_n)^{\mathrm{T}} \in \mathbb{R}_n$. Snadno se ukáže, že $\|y\| = 1$, neboť platí $\|Q^{\mathrm{T}} v\| = \|v\|$ pro každý vektor $v \in \mathbb{R}_n$. Předpokládáme, že pro vlastní čísla $\lambda_i, i = 1,2,...,n$ matice C platí $\lambda_1 \le \lambda_2 \le ... \le \lambda_n$, můžeme pro největší vlastní číslo λ_n odvodit následující

$$v^{\mathrm{T}} C v = v^{\mathrm{T}} Q \Lambda Q^{\mathrm{T}} v = y^{\mathrm{T}} \Lambda y$$
$$y^{\mathrm{T}} \Lambda y = \sum_{i=1}^{n} \lambda_i y_i^2$$
$$\Downarrow$$
$$\sum_{i=1}^{n} \lambda_i y_i^2 \le \sum_{i=1}^{n} \lambda_n y_i^2 = \lambda_n \|y\|^2 = \lambda_n .$$

Tím jsme ukázali, že platí obě nerovnosti

$$\lambda_n \le \max_{v:\|v\|=1} v^{\mathrm{T}} C v \text{ a } \max_{v:\|v\|=1} v^{\mathrm{T}} C v \le \lambda_n$$

Platí tedy rovnost

$$\lambda_n = \max_{v:\|v\|=1} v^{\mathrm{T}} C v .$$

Důkaz druhé rovnosti $\lambda_1 = \min_{v:\|v\|=1} v^{\mathrm{T}} C v$ je analogický, nebudeme ho tedy rozepisovat. ■

Větu 3.3 uvádíme ve zjednodušeném znění, které postačuje našim účelům. Obecně lze uvést vztahy také pro vlastní čísla $\lambda_2, \lambda_3, ..., \lambda_{n-1}$. Obecnější formulaci věty 3.3 je možné nalézt v (Dym, 2007).

Vraťme se zpět k původnímu problému, tedy k hledání nejmenšího vlastního čísla matice M a jemu příslušného vlastního vektoru. Předpoklady věty 3.3 jsou splněny, matice M je symetrická, pro případ v eukleidovském prostoru E_3 je typu (3×3), pro případ v eukleidovské rovině E_2 je typu (2×2). Dále víme, že vektor u v předpisu kvadratické formy $\sigma^2 = u^{\mathrm{T}} M u$ je jednotkový. K hledání vlastních čísel matice M a jim příslušných vlast-

ních vektorů budeme využívat standardních postupů, viz například (Abadir a Magnus, 2005; Bečvář, 2002; Dym, 2007). Výpočet ukážeme později na konkrétním příkladě.

Nalezením nejmenšího vlastního čísla matice M završíme úlohu ortogonálního prokládání dat přímkou, neboť tak nacházíme minimum kvadratické formy $\sigma^2 = u^{\mathrm{T}} M u$. Příslušný jednotkový vlastní vektor u je potom směrovým vektorem hledané přímky q, neboť tuto kvadratickou formu minimalizuje. Popis přímky q, jak už jsme uvedli na začátku tohoto oddílu, získáme v parametrickém tvaru

$$q(t) = A + ut, \tag{3.60}$$

kde $t \in \mathbb{R}$ a u je jednotkový směrový vektor přímky, tj. $\|u\| = 1$.

Výsledky, ke kterým jsme dospěli, uvádí například Ahn v (2004) nebo je čtenář může nalézt na webových stránkách (Geometric Tools, 2012). Předloženy jsou ale pouze výsledky a závěry, které řeší úlohu ortogonálního prokládání dat přímkou. V žádném jmenovaném zdroji nejsou uvedené postupy výpočtů, tvrzení, věty a nutná matematická teorie, na které jsou výpočty a odvození založeny. V české literatuře důkladné odvození této metody s názornou geometrickou interpretací rovněž chybí. Tímto jsme doplnili dostupnou literaturu o nutnou teorii, důkazy a výpočetní postupy, které jsou potřebné k pochopení dané problematiky.

Shrňme nyní odvozené postupy a demonstrujme jejich princip na názorných příkladech jak v rovině, tak v prostoru. Předložme rovněž případovou studii ortogonálního prokládání dat přímkou pro různé vstupní množiny. Metodu ortogonálního prokládání dat přímkou použijeme v bodové fázi rekonstrukce povrchů z množiny bodů.

Příklad 3.2: *ORTOGONÁLNÍ PROKLÁDÁNÍ DAT PŘÍMKOU V ROVINĚ A V PROSTORU*. Popsanou metodu nyní předveďme na příkladech. Nejdříve se věnujme ortogonálnímu prokládání dat přímkou v rovině. Mějme dánu množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i]\}_{i=0}^{n}$ v eukleidovské rovině E_2. Princip metody demonstrujme na menším počtu vstupních bodů. Experimentální data pro ortogonální prokládání jsou uvedena v tabulce 3.2, přičemž se jedná o stejná data, která byla použita také v ukázce aproximace dat metodou nejmenších čtverců v příkladě 3.1. Stejná data jsme volili proto, abychom mohli výsledné aproximace porovnat.

Dále je podrobně rozepsán postup výpočtu vedoucí k nalezení aproximující přímky $q(t) = A + ut$. Odvozeny jsou souřadnice bodu A a jednotkového směrového vektoru u přímky q. Naznačen je rovněž postup výpočtu vlastních čísel a příslušných vlastních vektorů matice M.

Na obrázku 3.4 je potom znázorněna výsledná přímka, pro kterou vyžadujeme, aby součet druhých mocnin vyznačených vzdáleností zadaných bodů $\{[x_i, y_i]\}_{i=0}^{n}$ od přímky byl nejmenší možný.

Ve druhém případě se věnujeme metodě ortogonálního prokládání dat přímkou v prostoru. Nyní tedy máme zadanou množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3. Experimentální data pro ortogonální prokládání jsou

uvedena v tabulce 3.3. Použili jsme data z tabulky 3.2, ke kterým jsme přidali z-ovou souřadnici. I když se jedná o stejný princip, je pro případ v eukleidovském prostoru E_3 také připojen postup výpočtu vedoucí k nalezení aproximující přímky $q(t) = A + ut$. Opět jsou odvozeny souřadnice bodu A a jednotkového směrového vektoru u přímky q a uveden je též výpočet vlastních čísel a příslušných vlastních vektorů matice M. (Ve všech případech jsou číselné hodnoty při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Obrázek 3.5 ukazuje výslednou přímku v prostoru. Opět jsou vyznačeny také vzdálenosti zadaných bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ od přímky q. Doporučujeme čtenáři prohlédnout si výstup přímo ve výpočetním prostředí MATLAB, kde lze s výsledným obrázkem pohybovat. Prostorová situace je tak názornější.

Výstupy jsou získány z výpočetního prostředí MATLAB. Programy, které řeší ortogonální prokládání dat přímkou v rovině a v prostoru, jsou k dispozici na přiloženém médiu. Ortogonální prokládání dat přímkou je řešeno pomocí odvozených vztahů, přičemž k výpočtu vlastních vektorů a příslušných vlastních čísel jsou použity funkce MATLABu. K řešení problému ortogonálního prokládání dat přímkou lze také přímo využít funkci `mldivide`, která je v prostředí MATLAB k dispozici.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu): `primka_orto_2d.m` (programy/orto_primka2d), `primka_orto_3d.m` (programy/orto_primka3d).

x_i	-2	0	1	2	2,5	3	4	4,5	5	5,5	
y_i	-3	-4	-2	1	-2	-3	1	-2	1,5	3	
x_i	5,5	6	6	7	8	8,5	9	9	9	10	10
y_i	-1,5	1	2	4	1,5	4	1	4,5	5	4	6

Tabulka 3.2: Experimentální data v rovině pro ortogonální prokládaní přímkou

Parametrické vyjádření přímky $q(t) = A + ut, t \in \mathbb{R}$

Výpočet souřadnic bodu A

$$A = \frac{1}{n+1}\sum_{i=0}^{n} X_i$$

$$A = \left[A^x, A^y\right]$$

$$A^x = \frac{1}{n+1}\sum_{i=0}^{n} X_i^x = \frac{1}{21}\cdot 113{,}5 = 5{,}40476$$

$$A^y = \frac{1}{n+1}\sum_{i=0}^{n} X_i^y = \frac{1}{21}\cdot 22 = 1{,}04762$$

$$A = [5{,}40476; 1{,}04762]$$

Výpočet souřadnic směrového vektoru u

$$u = (u^x, u^y)^{\mathrm{T}}$$

Funkce (součet druhých mocnin vzdáleností bodů vstupní množiny od přímky q), jejíž minimum hledáme

$$\sigma^2 = u^{\mathrm{T}}\sum_{i=0}^{n}\left[(w_i \cdot w_i)E - w_i w_i^{\mathrm{T}}\right]u = u^{\mathrm{T}}Mu$$

$$M = \left[\sum_{i=0}^{n}(w_i^x)^2 + \sum_{i=0}^{n}(w_i^y)^2\right]\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} - \begin{pmatrix}\sum_{i=0}^{n}(w_i^x)^2 & \sum_{i=0}^{n}w_i^x w_i^y\\ \sum_{i=0}^{n}w_i^x w_i^y & \sum_{i=0}^{n}(w_i^y)^2\end{pmatrix} = \begin{pmatrix}\sum_{i=0}^{n}(w_i^y)^2 & -\sum_{i=0}^{n}w_i^x w_i^y\\ -\sum_{i=0}^{n}w_i^x w_i^y & \sum_{i=0}^{n}(w_i^x)^2\end{pmatrix}$$

$$M = [232{,}80952 + 175{,}95238]\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} - \begin{pmatrix}232{,}80952 & 170{,}34524\\ 170{,}34524 & 175{,}95238\end{pmatrix}$$

$$M = \begin{pmatrix}175{,}95238 & -170{,}34524\\ -170{,}34524 & 232{,}80952\end{pmatrix}$$

Výpočet vlastních čísel a příslušných vlastních vektorů matice M

$$\lambda E - M = \begin{pmatrix}\lambda - 175{,}95238 & 170{,}34524\\ 170{,}34524 & \lambda - 232{,}80952\end{pmatrix}$$

$$\det(\lambda E - M) = (\lambda - 175{,}95238)(\lambda - 232{,}80952) - 29017{,}50014 =$$
$$= \lambda^2 - 408{,}7619\lambda - 11945{,}88988$$

charakteristický polynom

kořeny charakteristického polynomu
$\lambda^2 - 408{,}7619\lambda - 11945{,}88988 = 0$

Vlastní čísla

$\lambda_1 = 31{,}67981 \qquad \lambda_2 = 377{,}08209$

jednotkový vlastní vektor příslušný menšímu vlastnímu číslu
je netriviálním řešením homogenní soustavy s maticí $\lambda_1 E - M$

$$\begin{pmatrix}-144{,}27257 & 170{,}34524\\ 170{,}34524 & -201{,}12971\end{pmatrix}\begin{pmatrix}u^x\\ u^y\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}$$

$(\lambda_1 E - M)u = o$

řešení soustavy

$u^x = -0{,}76309 \qquad u^y = -0{,}64629$

jednotkový směrový vektor přímky q

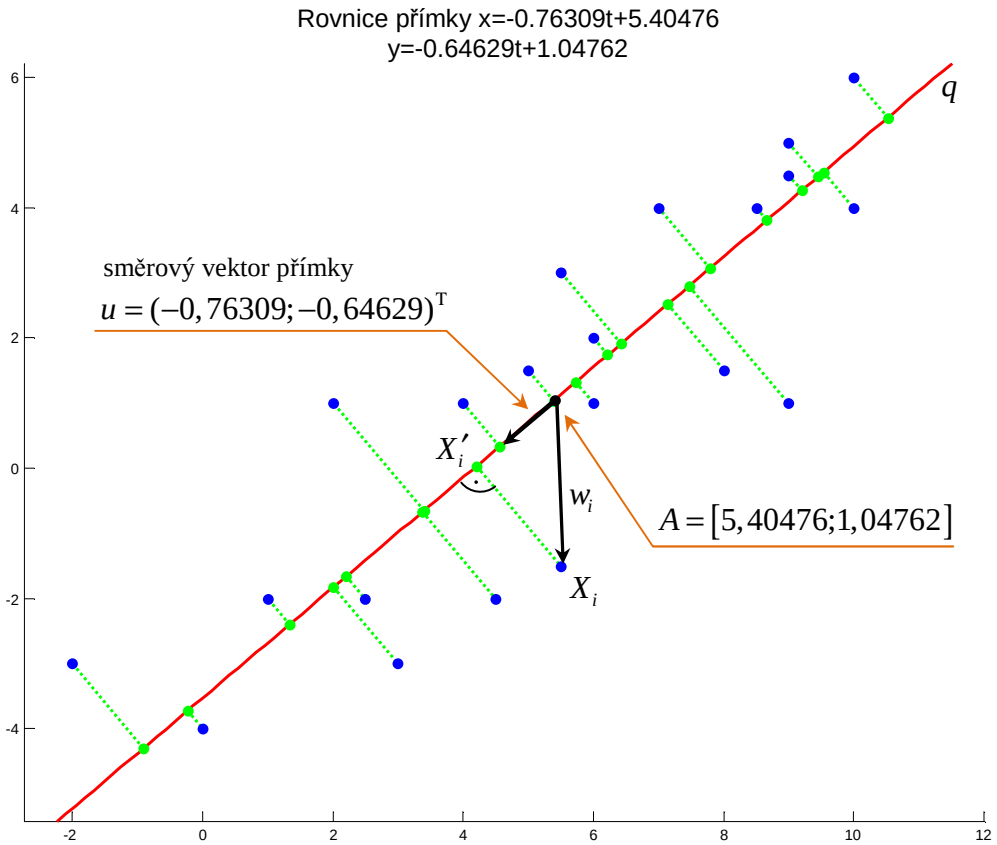
$u = (-0{,}76309; -0{,}64629)^{\mathrm{T}}$

$$\begin{aligned} x &= -0{,}76309t + 5{,}40476\\ y &= -0{,}64629t + 1{,}04762 \end{aligned} \qquad t \in \mathbb{R}$$

parametrické vyjádření přímky q

součet druhých mocnin vzdáleností daných bodů od přímky q

$\sigma^2 = u^{\mathrm{T}} M u = 31{,}67981$



Obrázek 3.4: Ortogonální prokládání dat přímkou v rovině – princip metody

x_i	-2	0	1	2	2,5	3	4	4,5	5	5,5	
y_i	-3	-4	-2	1	-2	-3	1	-2	1,5	3	
z_i	10	8	7	8,5	5	4	5	6	3.5	2	
x_i	5,5	6	6	7	8	8,5	9	9	9	10	10
y_i	-1,5	1	2	4	1,5	4	1	4,5	5	4	6
z_i	1,5	0	0,5	-2	-3	-2	-3,5	-4	-5	-5,5	-7

Tabulka 3.3: Experimentální data v prostoru pro ortogonální prokládání přímkou

Parametrické vyjádření přímky $q(t) = A + ut, t \in \mathbb{R}$

Výpočet souřadnic bodu A

$$A = \frac{1}{n+1}\sum_{i=0}^{n} X_i$$

$$A = \left[A^x, A^y, A^z\right]$$

$$A^x = \frac{1}{n+1}\sum_{i=0}^{n} X_i^x = \frac{1}{21}\cdot 113,5 = 5,40476$$

$$A^y = \frac{1}{n+1}\sum_{i=0}^{n} X_i^y = \frac{1}{21}\cdot 22 = 1,04762$$

$$A^z = \frac{1}{n+1}\sum_{i=0}^{n} X_i^z = \frac{1}{21}\cdot 29 = 1,38095$$

$$A = [5,40476; 1,04762; 1,38095]$$

$$M = \left[\sum_{i=0}^{n}(w_i^x)^2 + \sum_{i=0}^{n}(w_i^y)^2 + \sum_{i=0}^{n}(w_i^z)^2\right]\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} - \begin{pmatrix}\sum_{i=0}^{n}(w_i^x)^2 & \sum_{i=0}^{n}w_i^x w_i^y & \sum_{i=0}^{n}w_i^x w_i^z\\ \sum_{i=0}^{n}w_i^x w_i^y & \sum_{i=0}^{n}(w_i^y)^2 & \sum_{i=0}^{n}w_i^y w_i^z\\ \sum_{i=0}^{n}w_i^x w_i^z & \sum_{i=0}^{n}w_i^y w_i^z & \sum_{i=0}^{n}(w_i^z)^2\end{pmatrix} =$$

$$= \begin{pmatrix}\sum_{i=0}^{n}\left[(w_i^y)^2+(w_i^z)^2\right] & -\sum_{i=0}^{n}w_i^x w_i^y & -\sum_{i=0}^{n}w_i^x w_i^z\\ -\sum_{i=0}^{n}w_i^x w_i^y & \sum_{i=0}^{n}\left[(w_i^x)^2+(w_i^z)^2\right] & -\sum_{i=0}^{n}w_i^y w_i^z\\ -\sum_{i=0}^{n}w_i^x w_i^z & -\sum_{i=0}^{n}w_i^y w_i^z & \sum_{i=0}^{n}\left[(w_i^x)^2+(w_i^y)^2\right]\end{pmatrix}$$

$$M = \left[232{,}80952 + 175{,}95238 + 515{,}45238\right]\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} - \begin{pmatrix}232{,}80952 & 170{,}34524 & -333{,}9881\\ 170{,}34524 & 175{,}95238 & -247{,}88095\\ -333{,}9881 & -247{,}88095 & 515{,}45238\end{pmatrix}$$

$$M = \begin{pmatrix}691{,}40477 & -170{,}34524 & 333{,}9881\\ -170{,}34524 & 748{,}2619 & 247{,}88095\\ 333{,}9881 & 247{,}88095 & 408{,}7619\end{pmatrix}$$

Výpočet vlastních čísel a příslušných vlastních vektorů matice M

$$\lambda E - M = \begin{pmatrix}\lambda - 691{,}40477 & 170{,}34524 & -333{,}9881\\ 170{,}34524 & \lambda - 748{,}2619 & -247{,}88095\\ -333{,}9881 & -247{,}88095 & \lambda - 408{,}7619\end{pmatrix}$$

$$\det(\lambda E - M) = (\lambda - 691{,}40477)(\lambda - 748{,}2619)(\lambda - 408{,}7619) + 28205521{,}65631 -$$
$$-111548{,}04776(\lambda - 748{,}2619) -$$
$$-29017{,}50014(\lambda - 408{,}7619) -$$
$$-61444{,}96655(\lambda - 691{,}40477) =$$
$$= \lambda^3 - 1848{,}42857\lambda^2 + 903822{,}2185\lambda - 45456457{,}73$$

charakteristický polynom

kořeny charakteristického polynomu

$$\lambda^3 - 1848{,}42857\lambda^2 + 903822{,}2185\lambda - 45456457{,}73 = 0$$

Vlastní čísla

$$\lambda_1 = 56{,}65731 \qquad \lambda_2 = 878{,}39897 \qquad \lambda_3 = 913{,}37229$$

jednotkový vlastní vektor příslušný nejmenšímu vlastnímu číslu je netriviálním řešením homogenní soustavy s maticí $\lambda_1 E - M$

$$\begin{pmatrix} -634{,}74745 & 170{,}34524 & -333{,}9881 \\ 170{,}34524 & -691{,}6046 & -247{,}88095 \\ -333{,}9881 & -247{,}88095 & -352{,}1046 \end{pmatrix} \begin{pmatrix} u^x \\ u^y \\ u^z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$(\lambda_1 E - M)u = o$$

řešení soustavy

$u^x = -0{,}50855 \quad u^y = -0{,}39877 \quad u^z = 0{,}76312$

jednotkový směrový vektor přímky q

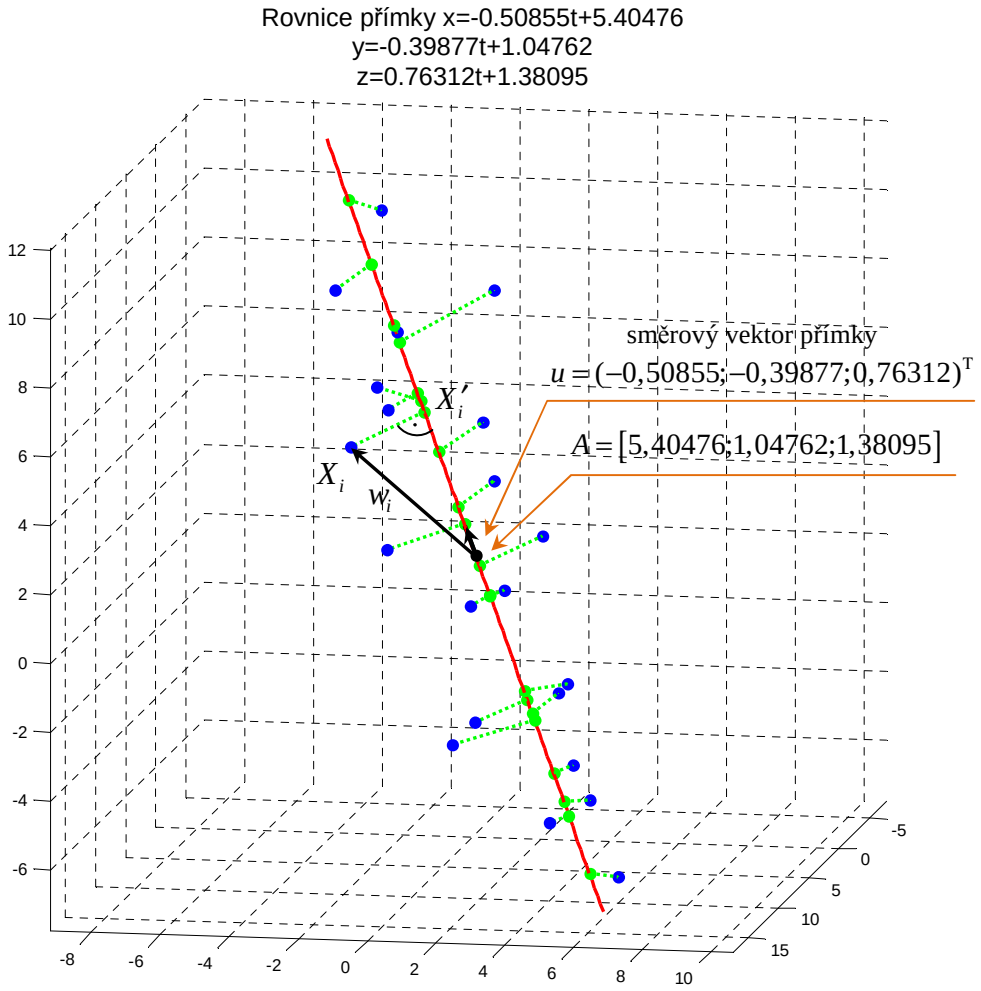
$u = (-0{,}50855; -0{,}39877; 0{,}76312)^{\mathrm{T}}$

$$\begin{aligned} x &= -0{,}50855t + 5{,}40476 \\ y &= -0{,}39877t + 1{,}04762 \quad t \in \mathbb{R} \\ z &= 0{,}76312t + 1{,}38095 \end{aligned}$$

parametrické vyjádření přímky q

součet druhých mocnin vzdáleností daných bodů od přímky q

$\sigma^2 = u^{\mathrm{T}} M u = 56{,}65731$



Obrázek 3.5: Ortogonální prokládání dat přímkou v prostoru – princip metody

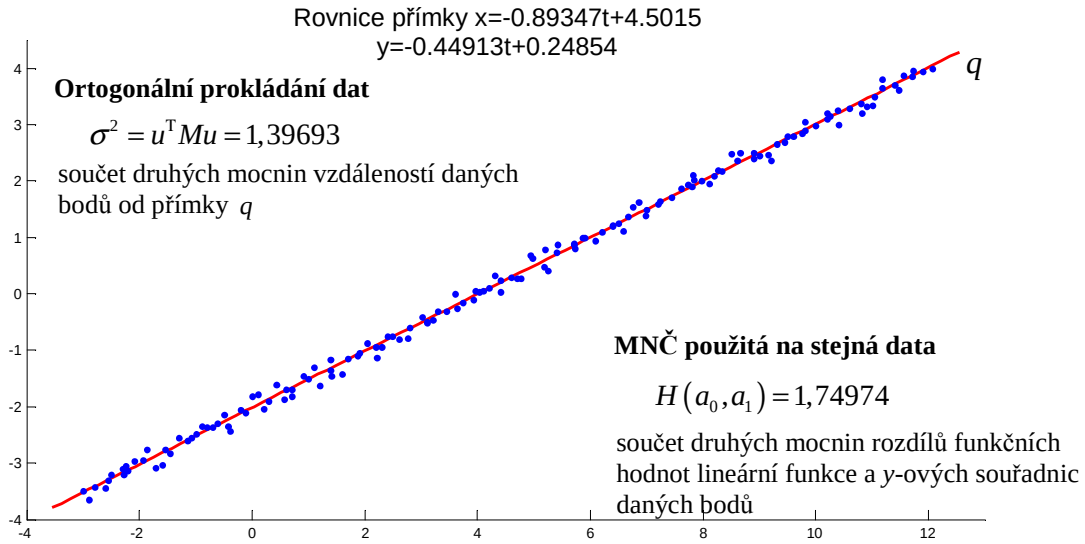
3.3.1 Případová studie v eukleidovské rovině E_2

Zaměřme se nyní na rozbor několika různých vstupních bodových množin a sledujme výsledky ortogonálního prokládání dat přímkou v rovině. Cílem je prověřit fungování odvozeného typu aproximace a ukázat výsledky, které metoda dává. Nyní již nebudeme uvádět jednotlivé výpočty. Číselné údaje a výsledky výpočtů lze získat spuštěním programu vytvořeného ve výpočetním prostředí MATLAB s názvem `primka_orto_2d.m`, kde lze měnit vstupní bodové množiny. Testovací bodové množiny jsou k dispozici na přiloženém médiu pod názvy `body01.txt`, `body02.txt`, … (cesta: bodove_mnoziny/orto_primka2d).

Ortogonální prokládání dat přímkou představuje široce užívanou metodu aproximace. Jako zřejmé využití tohoto geometrického prokládání dat se jeví hledání přímky, která dobře reprezentuje vstupní množinu bodů. Na rozdíl od klasické metody nejmenších čtverců má tato metoda přirozenou geometrickou a praktickou interpretaci. Obrázek 3.6 ukazuje příklad ortogonálního prokládání dat, o kterých předpokládáme, že by měly ležet na nějaké přímce, ale vlivem chyb v měření jsou tyto body zašuměné.

Pokud hovoříme o zašumění bodů, máme vždy na mysli zkreslení ideálních dat přičtením vhodné náhodné veličiny. Chceme simulovat reálná nedokonalá fyzikální měření, proto používáme náhodnou veličinu s normálním rozdělením a nulovou střední hodnotou.

V obrázku 3.6 je pro porovnání rovněž uveden výsledný součet druhých mocnin odchylek, viz (3.23), pokud na stejná data použijeme klasickou metodu nejmenších čtverců z oddílu 3.2 a nalezneme tak lineární aproximující funkci.



Obrázek 3.6: Ortogonální prokládání dat přímkou – praktická úloha, aproximace bodů v rovině

Na základě vlastností, které ortogonální prokládání dat přímkou v rovině splňuje a které jsme odvodili v této podkapitole, navrhujeme další možné použití této metody a to k analýze bodové množiny z hlediska osové symetrie bodů.

Intuitivně je pojem osové symetrie jistě zřejmý, zaveďme si jej pro úplnost formálně.

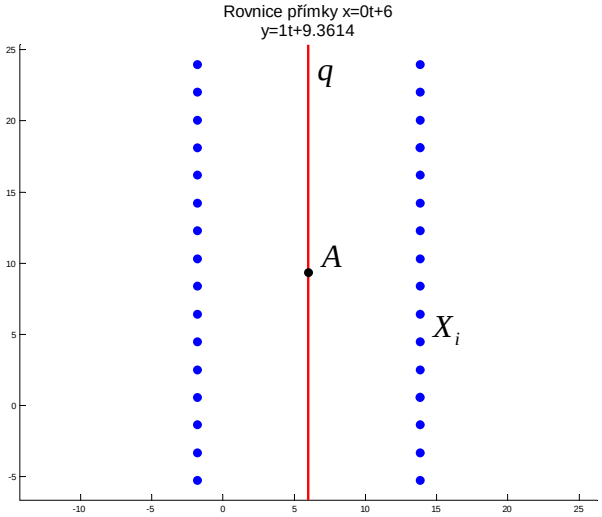
Definice 3.2 (*Osová symetrie s osou o v* E_2). Nechť je dána přímka o v eukleidovské rovině E_2. Potom zobrazení $f_o : E_2 \to E_2$, které každému bodu $A \in E_2$, $A \notin o$ přiřadí bod $f_o(A) \in E_2$ tak, že přímka $Af_o(A)$ je kolmá k přímce o a střed úsečky $Af_o(A)$ leží na přímce o a každému bodu $A \in E_2$, $A \in o$ přiřadí bod $f_o(A) = A$, nazveme *osovou symetrií s osou o*. ■

Definice 3.3 (*Osa symetrie*). Nechť jsou dány přímka o a neprázdná množina $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i]\}_{i=0}^{n}$ v eukleidovské rovině E_2. Řekneme, že množina bodů $\mathcal{X}$ je symetrická podle osy o, jestliže $f_o(\mathcal{X}) = \mathcal{X}$ a přímku o označíme jako *osu symetrie* bodové množiny $\mathcal{X}$. ■

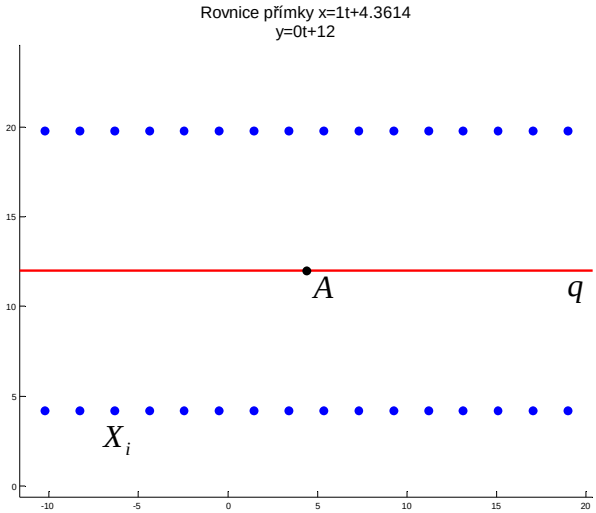
Na základě uvedených definic 3.2 a 3.3 použijme ortogonální prokládání dat přímkou v rovině k hledání os symetrií bodových množin. Všechny bodové množiny v této případové studii jsou počítačově generovány, předem tedy víme, jaké výsledky z hlediska osové symetrie mají vycházet. Snadněji tak lze ověřit správnost výstupů. Dále předpokládáme, že všechny zkoumané množiny jsou bodově symetrické podle jedné nebo dvou os. Chyby v měření při získávání bodových množin budeme později rovněž uvažovat.

Předveďme několik případů ortogonálního prokládání dat přímkou v rovině, začněme jednoduchými bodovými množinami.

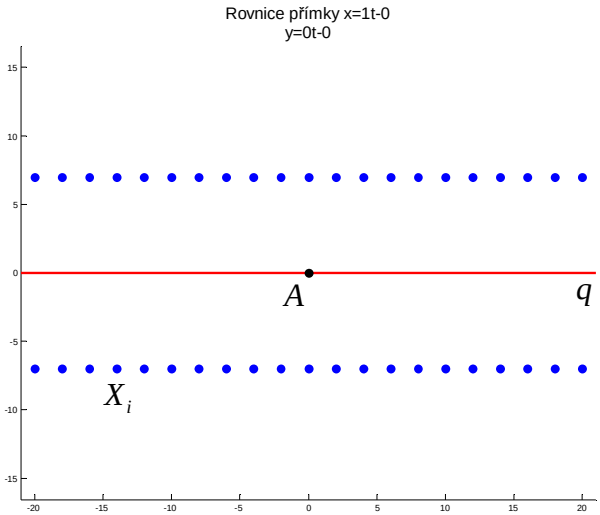
První čtyři případy, tj. obrázky 3.7-3.10, studují osovou symetrii bodových množin získaných z pravidelného navzorkování dvou rovnoběžných úseček. Výsledkem ortogonálního prokládání je vždy jedna z os symetrií těchto množin a to ta, v jejímž směru je množina více protáhlá. Tuto osu budeme označovat za *hlavní*. Je zřejmé, že takovéto výsledky bychom nedostali, kdybychom použili klasickou metodu nejmenších čtverců. První výslednou přímku navíc nelze vyjádřit jako funkční závislost. Metodou ortogonálního prokládání dat tedy lze hledat osy speciálních osově symetrických bodových množin. Je však nutné k analýze přistupovat obezřetně, to si ukážeme dále.



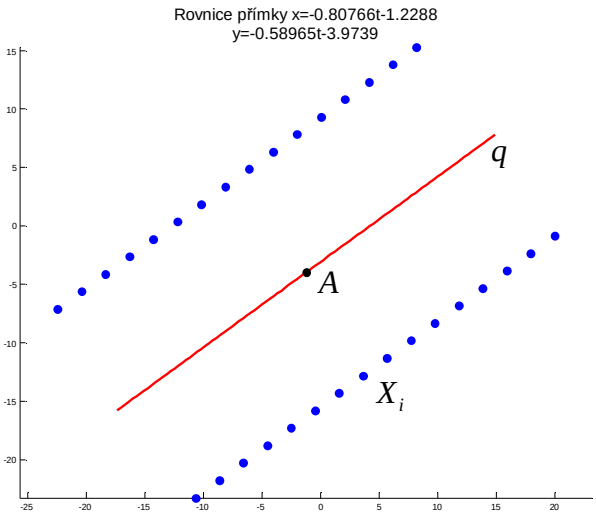
Obrázek 3.7: Ortogonální prokládání dat přímkou v rovině – symetrie bodů podle svislé osy



Obrázek 3.8: Ortogonální prokládání dat přímkou v rovině – symetrie bodů podle vodorovné osy

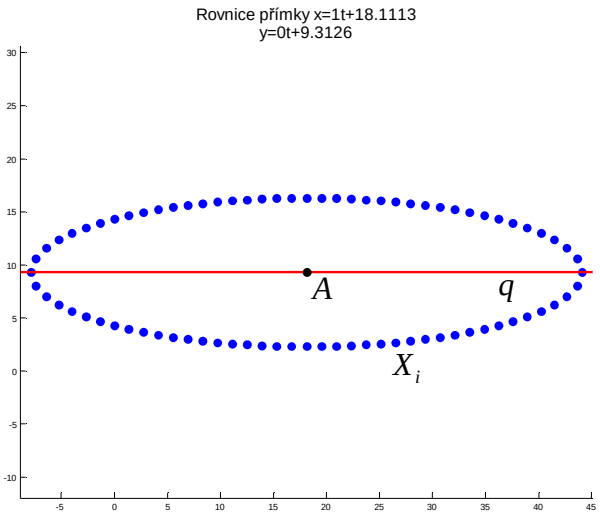


Obrázek 3.9: Ortogonální prokládání dat přímkou v rovině – symetrie bodů podle vodorovné osy (speciální případ, výsledná přímka je osa x a určující bod přímky je počátek soustavy souřadnic)

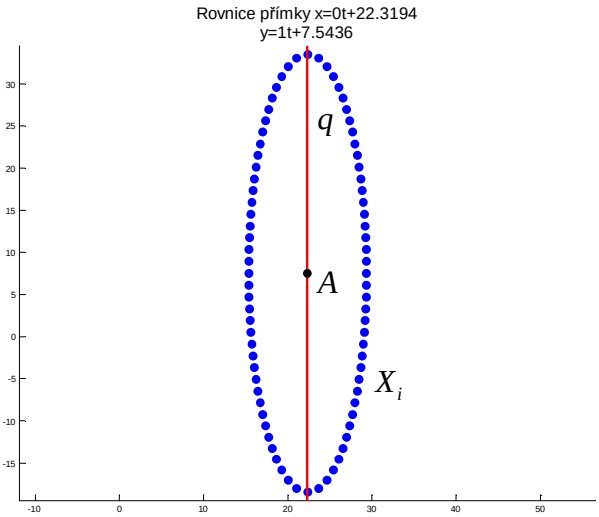


Obrázek 3.10: Ortogonální prokládání dat přímkou v rovině – symetrie bodů podle obecné osy

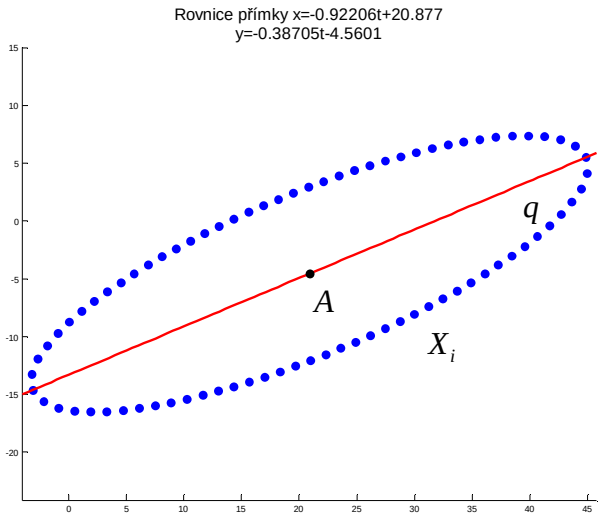
Ortogonální prokládání dat přímkou lze použít například pro hledání osy bodové množiny získané navzorkováním elipsy, jak ilustrují obrázky 3.11-3.13. Výsledná přímka odpovídá hlavní ose této elipsy. Obrázek 3.14 ilustruje ověření správnosti nalezené osy zobrazením bodů jedné poloroviny určené osou v této osové symetrii. Zobrazené body jsou na obrázku oranžově a překrývají se s body vstupní množiny, což je možné ověřit implementačně. Na obrázcích 3.11-3.13 jsou symetrické bodové množiny navzorkovány pravidelně. Na obrázku 3.15 se jedná o případ nepravidelného rozmístění bodů. Opět můžeme sledovat nalezení osy a na obrázku 3.16 ověření správnosti této osy.



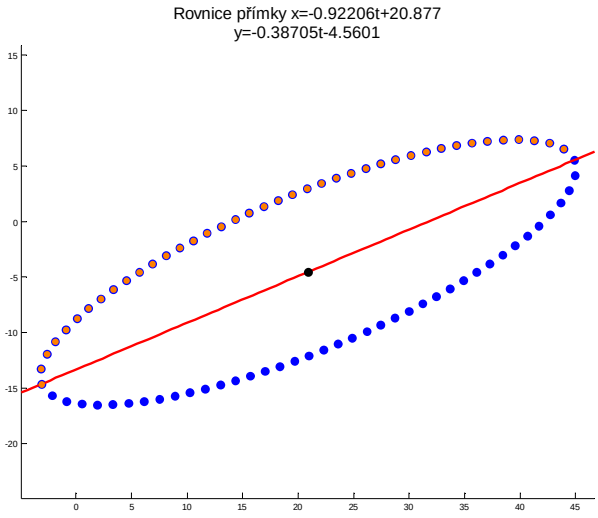
Obrázek 3.11: Ortogonální prokládání dat přímkou v rovině – hlavní osa elipsy rovnoběžná s osou x



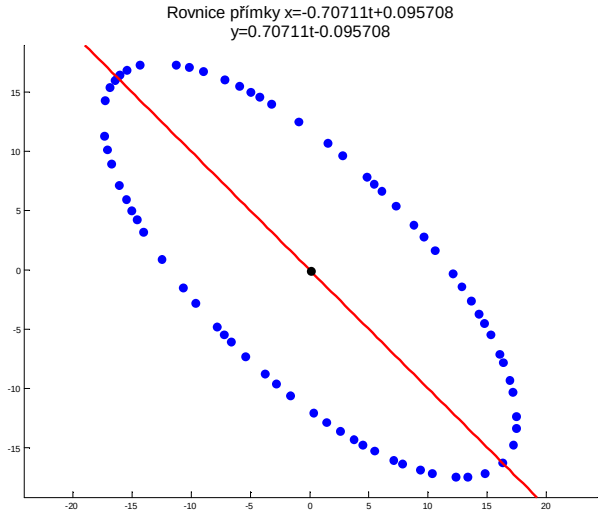
Obrázek 3.12: Ortogonální prokládání dat přímkou v rovině – hlavní osa elipsy rovnoběžná s osou y



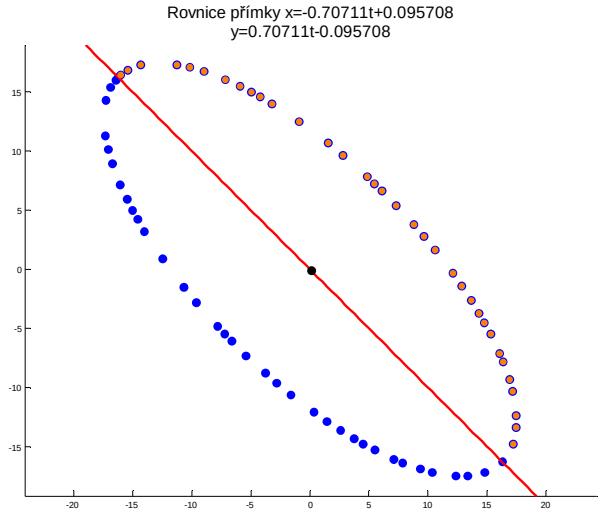
Obrázek 3.13: Ortogonální prokládání dat přímkou v rovině – hlavní osa elipsy (obecný případ)



Obrázek 3.14: Ověření správnosti osy symetrie – zobrazení bodů jedné poloroviny určené osou v nalezené osové symetrii (obrazy bodů oranžově)



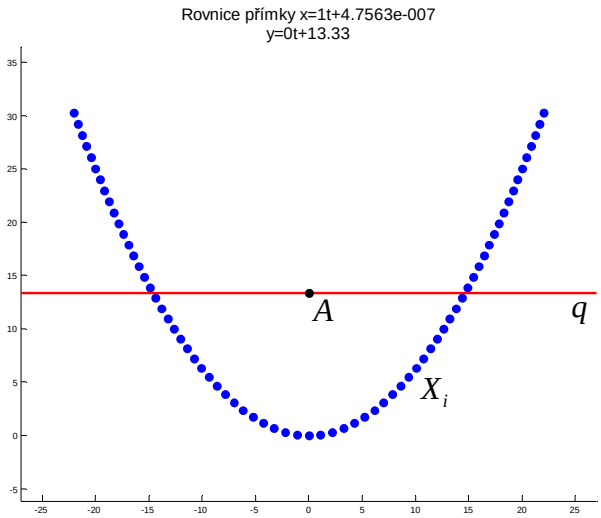
Obrázek 3.15: Ortogonální prokládání dat přímkou v rovině – hlavní osa elipsy (obecný případ), body navzorkovány nepravidelně



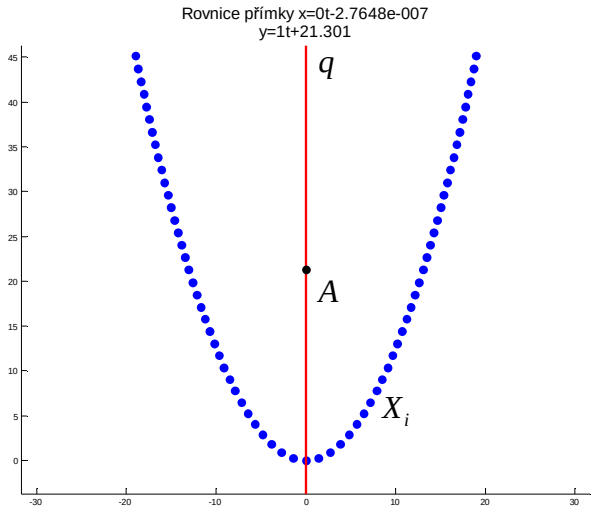
Obrázek 3.16: Ověření správnosti osy symetrie – zobrazení bodů jedné poloroviny určené osou v nalezené osové symetrii (obrazy bodů oranžově)

Pro některé symetrické bodové množiny nedostáváme jako výsledek ortogonálního prokládání osu symetrie, ale přímku, která je na osu kolmá. Takový příklad vidíme na obrázku 3.17. Proč je tomu tak, popíšeme v následujícím oddíle 3.3.3 a dále ověříme zavedením známé statistické metody analýzy hlavních komponent (*Principal Component Analysis*) v oddíle 3.6. Zkoumaná bodová množina na obrázku 3.17 je získána pravidelným navzorkováním oblouku paraboly a je bodově symetrická podle osy y. To, že dostáváme právě takový výsledek, je zřejmé už z tvaru analyzované množiny. Na obrázku 3.18 je bodová množina více protáhlá ve směru osy y, zde je výsledná přímka získaná ortogonálním prokládáním již osa symetrie bodů.

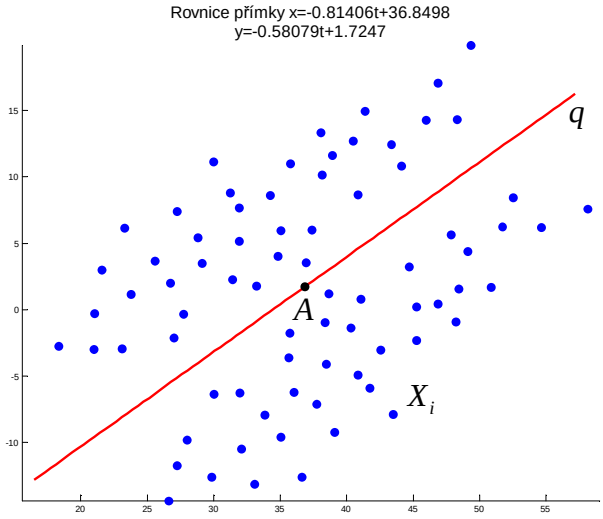
Rozšiřme tyto základní případy dále o zajímavější vstupní bodové množiny. Následující množiny jsou opět bodově symetrické podle jedné nebo dvou os. Obrázky 3.19, 3.21 a 3.22 ukazují příklady nalezení těchto os symetrií. Obrázek 3.20 opět ukazuje správnost výsledku. Na obrázku 3.22 můžeme vidět, že výsledkem ortogonálního prokládání je osa, v jejímž směru je bodová množina opět více protáhlá, dále je tato množina symetrická ještě podle další osy, která je na nalezenou osu kolmá.



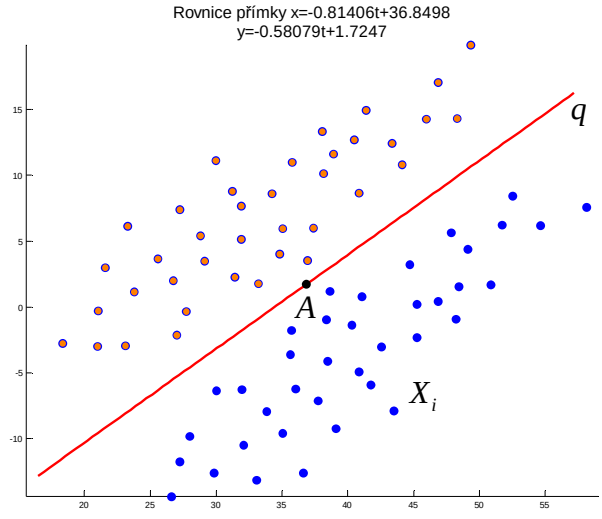
Obrázek 3.17: Ortogonální prokládání dat přímkou v rovině – speciální případ, kdy je výsledná přímka kolmá na osu symetrie



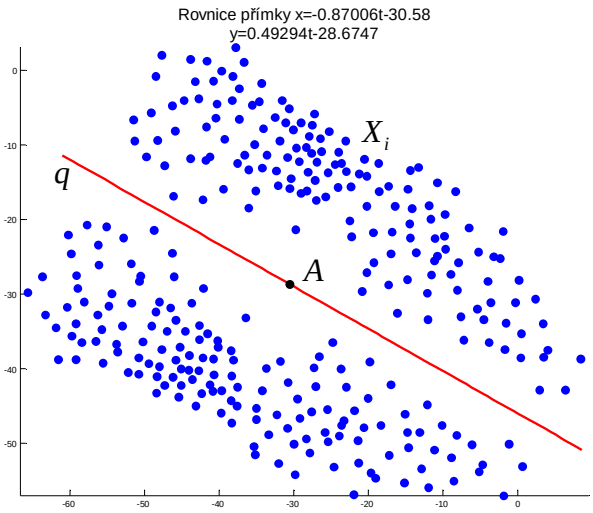
Obrázek 3.18: Ortogonální prokládání dat přímkou v rovině – výsledná přímka je osa symetrie (osa *y*)



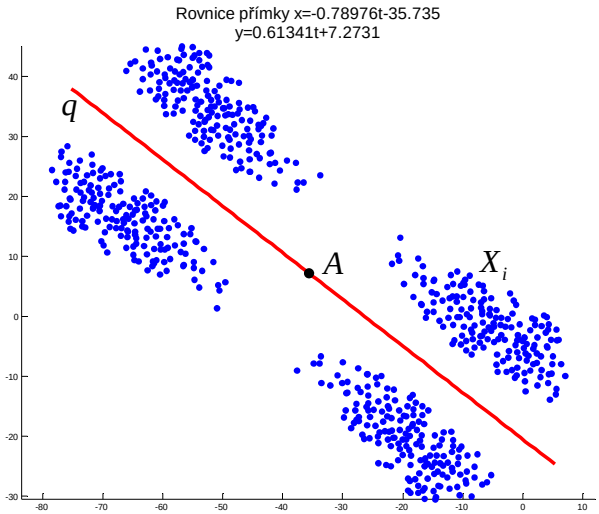
Obrázek 3.19: Ortogonální prokládání dat přímkou v rovině – osa symetrie bodové množiny



Obrázek 3.20: Ověření správnosti osy symetrie – zobrazení bodů jedné poloroviny určené osou v nalezené osové symetrii (obrazy bodů oranžově)

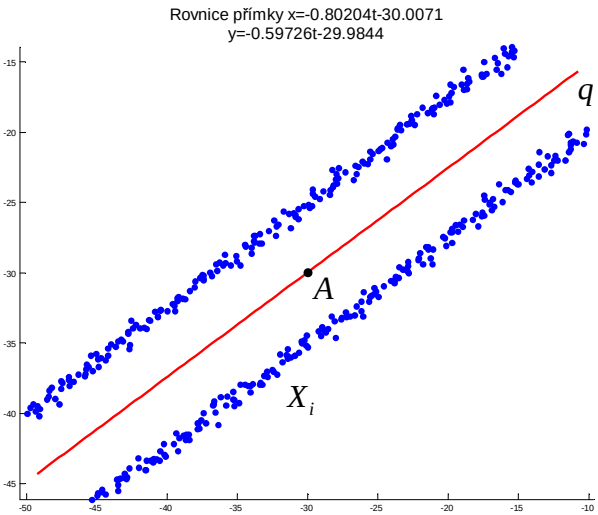


Obrázek 3.21: Ortogonální prokládání dat přímkou v rovině – osa symetrie bodové množiny

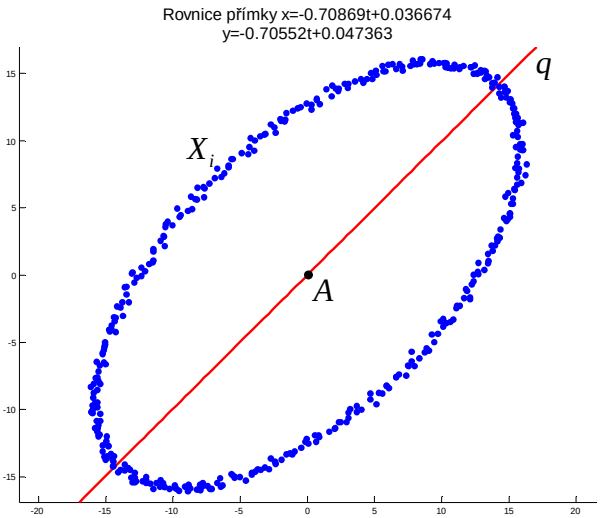


Obrázek 3.22: Ortogonální prokládání dat přímkou v rovině – jedna z os symetrie bodové množiny

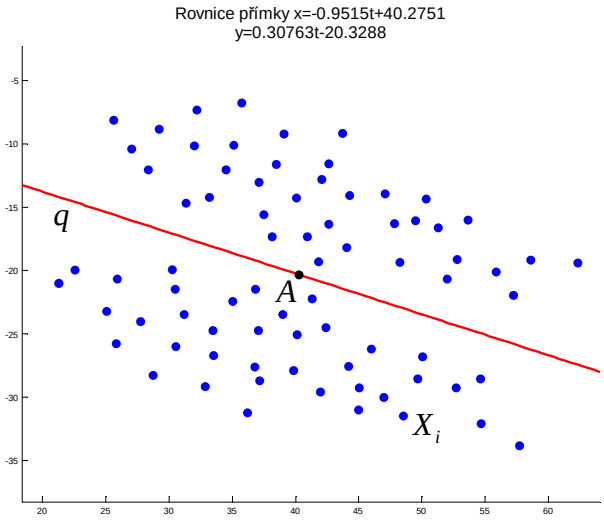
Na závěr této studie uveďme několik příkladů bodových množin, o kterých víme, že mají být bodově symetrické. Simulujeme však reálnou situaci, kdy dochází k nepřesnostem v měření, naše množiny tedy nebudou bodově symetrické přesně. Ortogonálním prokládáním těchto bodů přímkou lze tedy najít pouze přibližnou osu symetrie. Tyto příklady ukazují obrázky 3.23-3.25. Na obrázku 3.23 jsou body vygenerovány na dvou rovnoběžných úsečkách, na obrázku 3.24 na elipse a vždy zašuměny. Na obrázku 3.25 jde o náhodně rozmístěné body v rovině, které jsou přibližně osově symetrické. Na obrázku 3.26 jsou body jedné poloroviny určené osou zobrazeny v této nalezené osové symetrii.



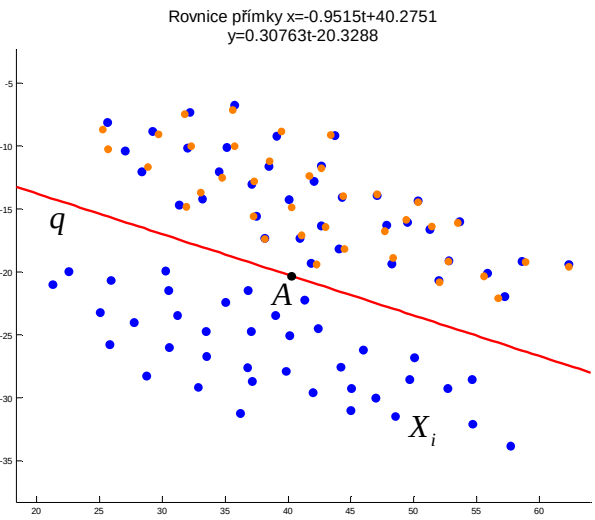
Obrázek 3.23: Ortogonální prokládání dat přímkou v rovině – osa symetrie bodové množiny, která obsahuje zašuměná data



Obrázek 3.24: Ortogonální prokládání dat přímkou v rovině – osa symetrie bodové množiny, která obsahuje zašuměná data



Obrázek 3.25: Ortogonální prokládání dat přímkou v rovině – osa symetrie bodové množiny, která obsahuje náhodně generovaná data



Obrázek 3.26: Ortogonální prokládání dat přímkou v rovině – zobrazení bodů jedné poloroviny určené osou v nalezené osové symetrii

Při rekonstrukci povrchů budeme pracovat s množinou bodů v eukleidovském prostoru E_3, v některých případech lze přesto využít i ortogonálního prokládání dat přímkou v rovině. Množiny, které analyzujeme, totiž splňují některé speciální vlastnosti, například rovinové symetrie, ty můžeme zkoumat v rovinných řezech a využít tak popsané metody.

Kromě toho využijeme i ortogonálního prokládání dat přímkou v eukleidovském prostoru E_3. Uveďme tedy obdobnou případovou studii.

3.3.2 Případová studie v eukleidovském prostoru E_3

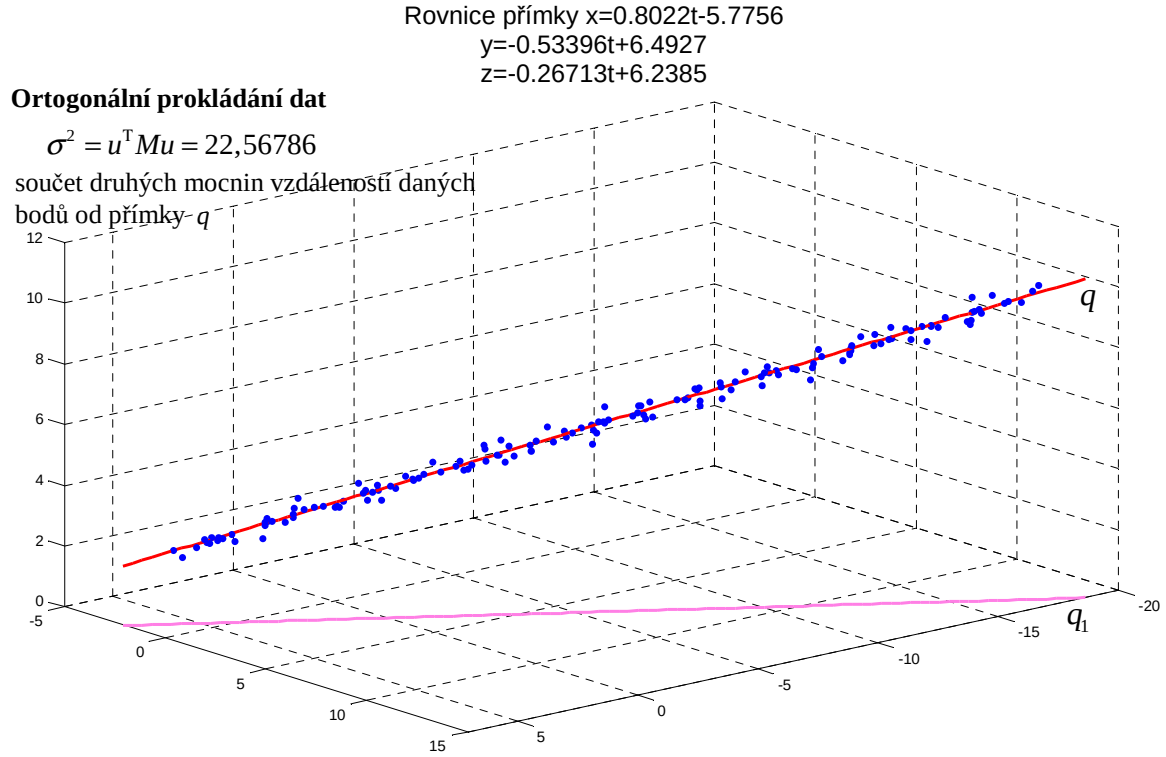
Opět se zaměřme na rozbor několika různých vstupních bodových množin a sledujme výsledky ortogonálního prokládání dat přímkou v prostoru. Cílem je opět prověřit fungování metody a ukázat výsledky pro různé vstupní bodové množiny. Ani tentokrát neuvádíme jednotlivé výpočty. Číselné údaje a výsledky výpočtů lze získat spuštěním programu vytvořeného ve výpočetním prostředí MATLAB s názvem `primka_orto_3d.m`, kde lze zaměnit vstupní bodové množiny. Testovací bodové množiny jsou k dispozici na přiloženém médiu pod názvy `body01.txt`, `body02.txt`, … (cesta: bodove_mnoziny/orto_primka3d).

Stejně tak jako v rovině se ortogonální prokládání dat přímkou používá často k aproximaci. Pro úplnost si tedy ukažme výsledek ortogonálního prokládání dat v prostoru (obrázek 3.27), o kterých předpokládáme, že by měly ležet na nějaké přímce, ale vlivem chyb v měření jsou tyto body zašuměné (opět modelujeme reálnou situaci normálním rozdělením).

Kromě tohoto klasického příkladu aproximace, navrhujeme podobně jako v rovině další možné použití této metody a to k analýze bodové množiny z hlediska osové symetrie bodů. Pojem osové symetrie v eukleidovském prostoru E_3 je jistě zřejmý, pro úplnost ji však zaveďme opět formálně. Není třeba uvádět nové definice, postačí modifikovat definice 3.2

a 3.3 tak, že eukleidovskou rovinu E_2 nahradíme eukleidovským prostorem E_3, jinak zůstávají definice stejné. Poznamenejme dále, že osovou symetrií v eukleidovském prostoru E_3 je v každé rovině obsahující osu této symetrie určena osová symetrie v rovině.

Přejděme k příkladům ortogonálního prokládání dat přímkou. Opět jsou všechny bodové množiny v této případové studii počítačově generovány, předem tedy víme, jaké výsledky z hlediska osové symetrie mají vycházet. Snadněji tak lze ověřit správnost výstupů. Dále předpokládáme, že všechny tyto množiny jsou bodově symetrické podle nějaké osy nebo os, chyby v měření při získávání bodových množin budeme uvažovat později.

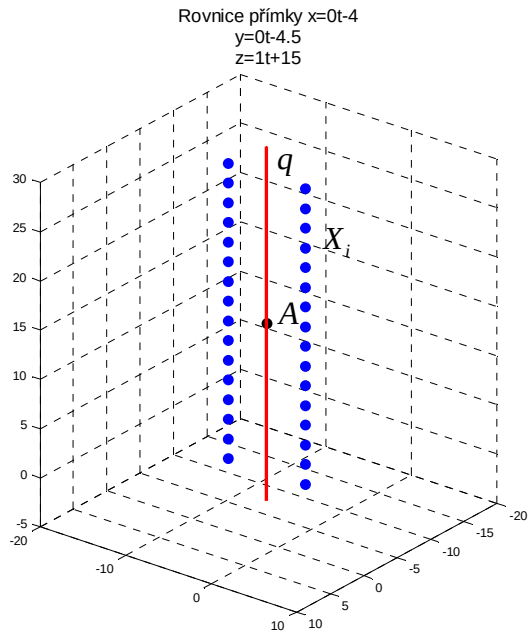


Obrázek 3.27: Ortogonální prokládání dat přímkou – praktická úloha, aproximace bodů v prostoru

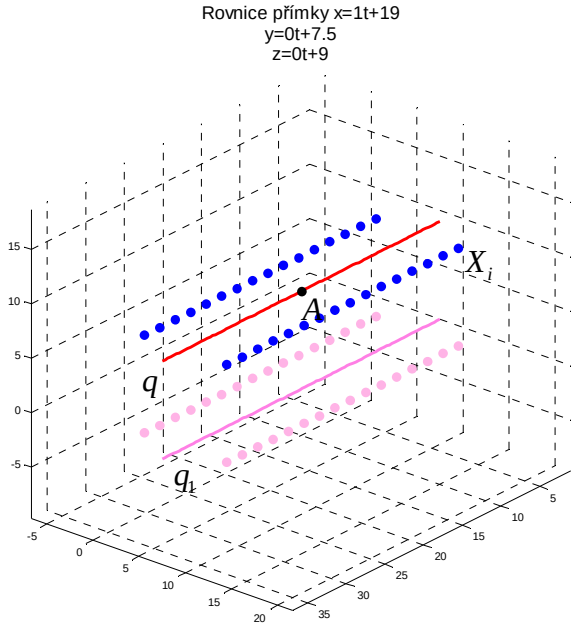
V prostoru analyzujeme obdobné množiny jako v rovině, začínáme opět od jednoduchých příkladů.

První čtyři případy, tj. obrázky 3.28-3.31, zkoumají osovou symetrii bodových množin získaných z pravidelného navzorkování dvou rovnoběžných úseček v prostoru. Výsledkem ortogonálního prokládání je jedna z os symetrií těchto množin. Opět ji označujme za hlavní osu. Pro lepší názornost a orientaci v prostorové situaci jsou do některých obrázků rovněž přidány půdorysy jednotlivých bodů a nalezené osy. Ortogonálním prokládáním dat přímkou lze opět v některých případech nalézt osy symetrických bodových množin.

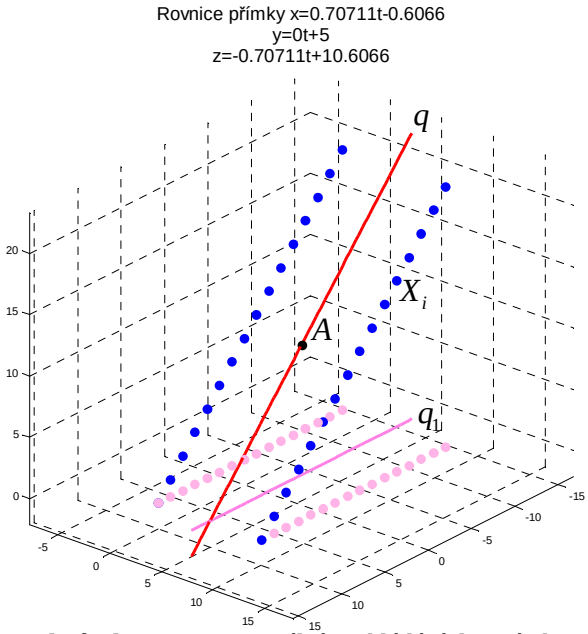
Další příklady množin, můžeme vidět na obrázcích 3.32 a 3.33, které vznikly navzorkováním čtyř rovnoběžných úseček. Platí, že body úhlopříčně protilehlých úseček jsou osově symetrické. Ortogonálním prokládáním dostáváme osu symetrie této množiny.



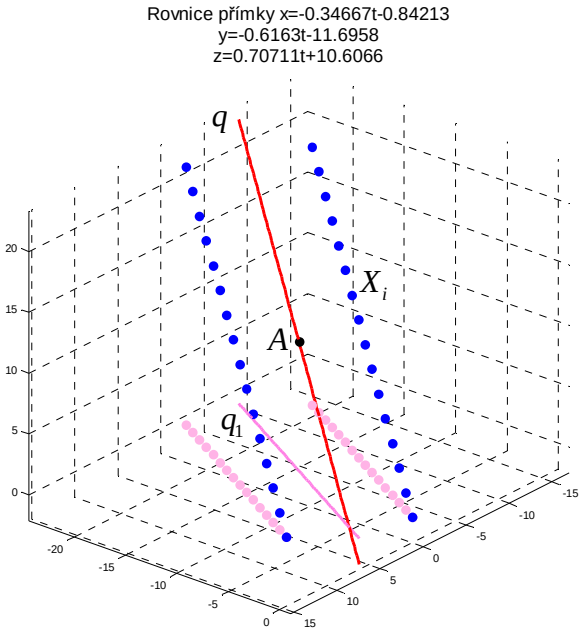
Obrázek 3.28: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy rovnoběžné s osou z



Obrázek 3.29: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy rovnoběžné s osou x



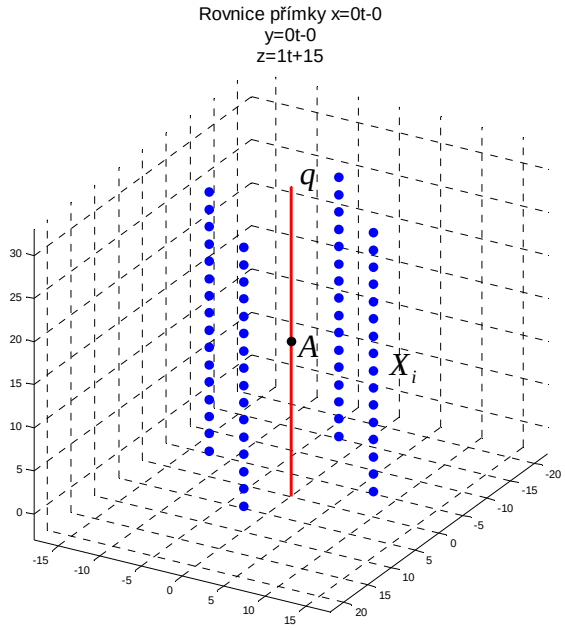
Obrázek 3.30: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy rovnoběžné s rovinou (x, z)



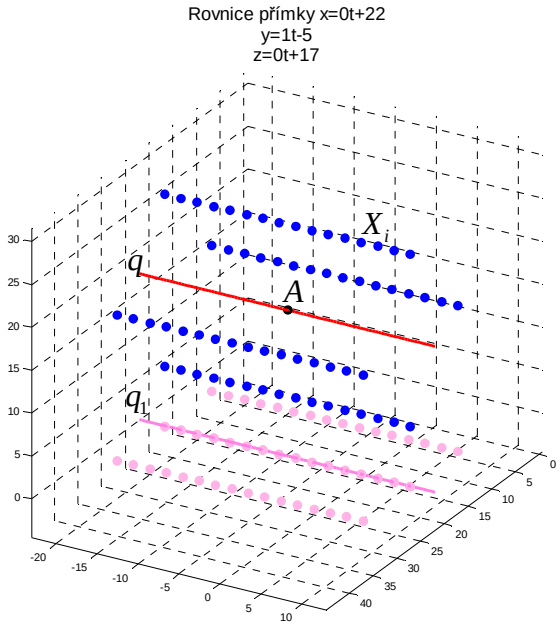
Obrázek 3.31: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle obecné osy

Další množina, kterou analyzujeme, již není pravidelná, ale víme, že je osově symetrická podle svislé osy. Tato množina je zobrazena na obrázcích 3.34 a 3.35. Pro lepší představu je doplněn rovněž pohled ve směru nalezené osy symetrie a jednotlivé spojnice bodů, které si odpovídají v nalezené osové symetrii v prostoru. Početně lze ověřit, že obrazy bodů vstupní

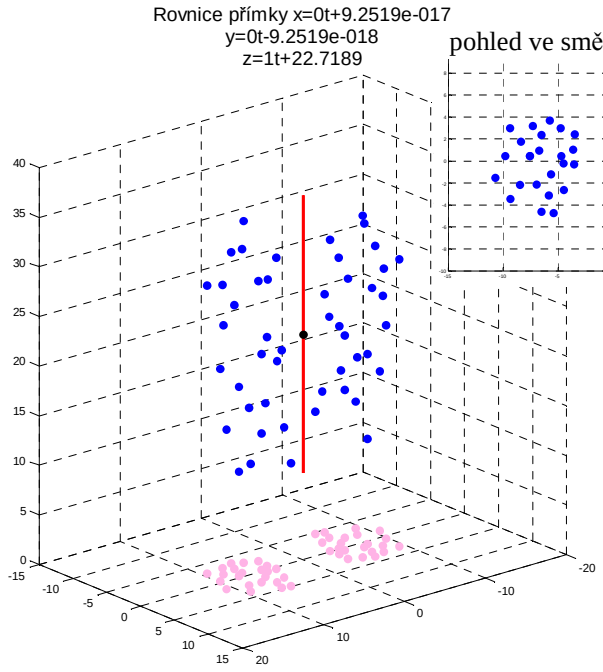
množiny v nalezené osové symetrii se překrývají s body původní množiny. Ilustraci této situace ponecháme ale až na případ, kdy nemáme přesně bodově symetrickou množinu.



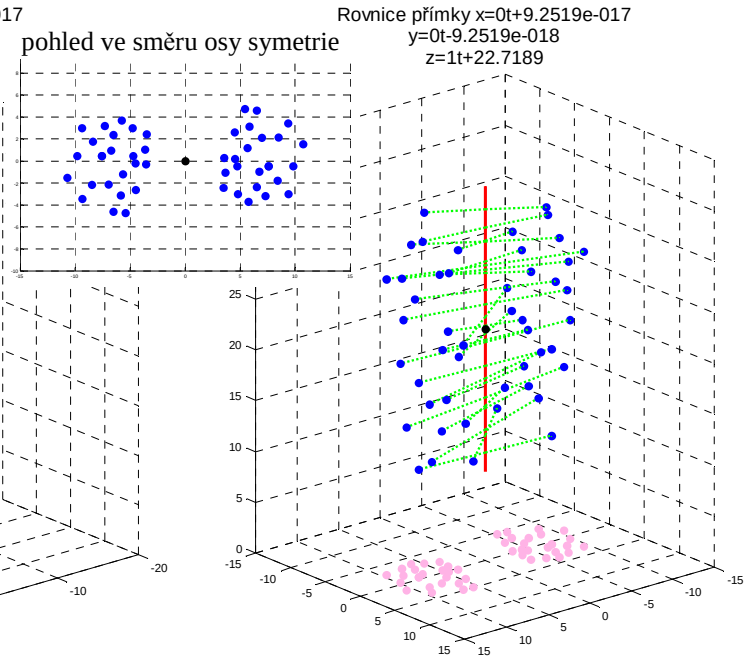
Obrázek 3.32: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy *z*



Obrázek 3.33: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy rovnoběžné s osou *y*



Obrázek 3.34: Ortogonální prokládání dat přímkou v prostoru – symetrie bodů podle osy *z*



Obrázek 3.35: Ortogonální prokládání dat přímkou v prostoru – spojnice odpovídajících si bodů v osové symetrii

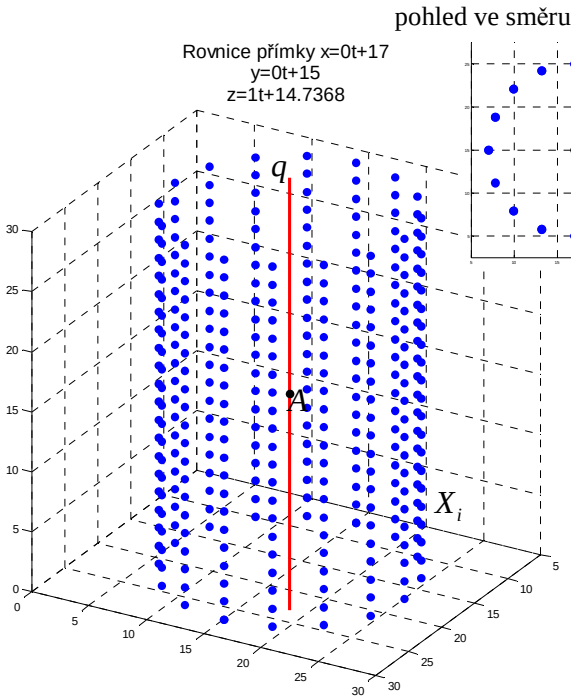
Bodová mračna, která chceme rekonstruovat, často odpovídají elementárním *rotačním plochám* (Surynková, 2008a) nízkého stupně nebo představují různé kombinace těchto ploch. Důležitou vlastností rotačních ploch je, že díky svému vytvoření rotací *tvořicí křivky* kolem osy, jsou podle osy rotace symetrické. Z toho rovněž plyne, že je rotační plocha symetrická podle každé roviny procházející osou rotace. Rovinný řez rotační plochou obsahující osu rotace je křivka, jež je podle této osy symetrická. Křivku nazýváme *meridiánem*. V rovině obsahující osu rotace proto dostáváme osovou symetrii. Je zřejmé, že osa rotace je zároveň osou rotační plochy. Tyto a další vlastnosti rotačních ploch jsou podrobně popsány v mé diplomové práci Plochy stavební praxe, (Surynková, 2008a).

Při rekonstrukci reálných povrchů většinou předem víme, z jakých ploch se tyto objekty skládají. Jedná se o válcové plochy s různými řídicími křivkami nejčastěji právě o válcové plochy rotační nebo o obecné rotační plochy. Pokud pracujeme s bodovými množinami, které vznikly nasnímáním reálných kleneb, jsou rovněž často tvořeny částmi rotační válcové plochy nebo jejich kombinacemi. To lze skutečně doložit analýzou historických pramenů případně studií stavebních plánů zkoumaných objektů, pokud jsou k dispozici. Tyto vlastnosti později uvedeme u každé rekonstruované množiny.

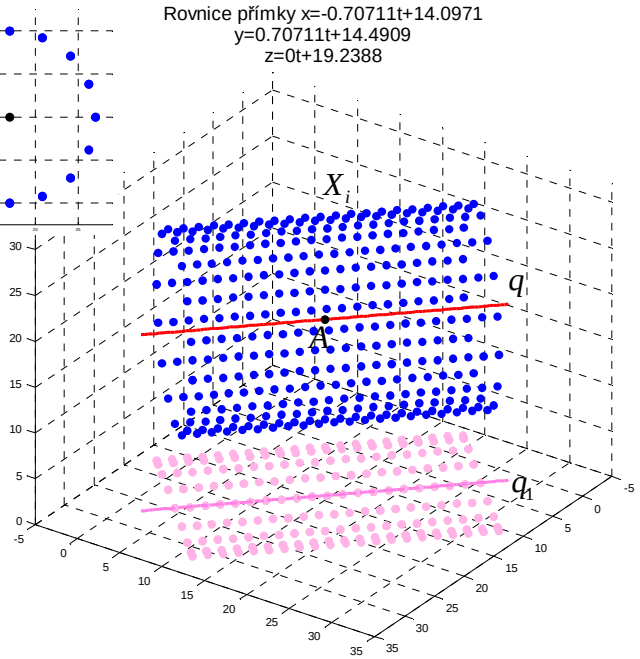
Podívejme se na analýzu počítačově generovaných množin, jejichž body jsme získali pravidelným navzorkováním části povrchu rotační válcové plochy. Obrázky 3.36 a 3.37 ilustrují výsledky hledání osy symetrie těchto množin.

Další množiny jsou získány pravidelným navzorkováním povrchů rotačních ploch. Na obrázcích 3.38 a 3.39 můžeme vidět ukázku bodů ležících na rotačním protáhlém elipsoidu, který vzniká rotací elipsy kolem její hlavní osy. Body jsou rozmístěny rovnoměrně na pravidelné síti *rovnoběžkových kružnic* (Surynková, 2008a). Ortogonálním prokládáním tak získáme osu této rotační plochy a zároveň hlavní osu tvořicí elipsy. Další dva obrázky 3.40 a 3.41 ukazují podobnou situaci. Množiny bodů jsou získány z povrchu jednodílného rotačního hyperboloidu, který vzniká rotací hyperboly kolem její vedlejší osy. Opět jsou body navzorkovány rovnoměrně na pravidelné síti rovnoběžkových kružnic plochy. Ortogonálním prokládáním získáme osu rotační plochy, což je zároveň vedlejší osa tvořicí hyperboly.

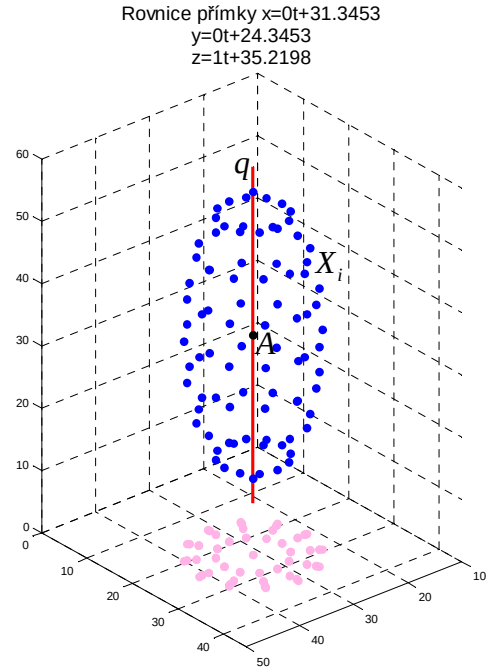
Doposud jsme v prostoru pracovali pouze s množinami bodově symetrickými podle nějaké osy. Na závěr této studie se opět věnujme případům, které se objevují v praktických aplikacích, kdy často dochází k nepřesnostem při snímaní bodů a tvorbě bodového mračna. Následující množiny tedy nebudou přesně bodově symetrické, předpokládáme však, že tyto nepřesnosti vznikly kvůli vnějším faktorům (nepřesnost skenovacího zařízení, lidský faktor). Víme tedy, že povrchy objektů, které bodové množiny popisují, mají osovou symetrii splňovat. Ortogonálním prokládáním těchto bodů přímkou nalezneme tedy přibližnou osu symetrie. Tyto příklady ukazují obrázky 3.42-3.45. Na obrázku 3.42 jsou body vygenerovány na dvou rovnoběžných úsečkách a zašuměny, na obrázku 3.43 náhodně na povrchu protáhlého rotačního elipsoidu a na obrázku 3.44 opět náhodně na povrchu rotační válcové plochy. Na obrázku 3.45 jde o náhodně rozmístěné body v prostoru, které jsou přibližně osově symetrické. Pro lepší názornost je polovina těchto bodů zobrazena v osové symetrii s osou v nalezené přímce.



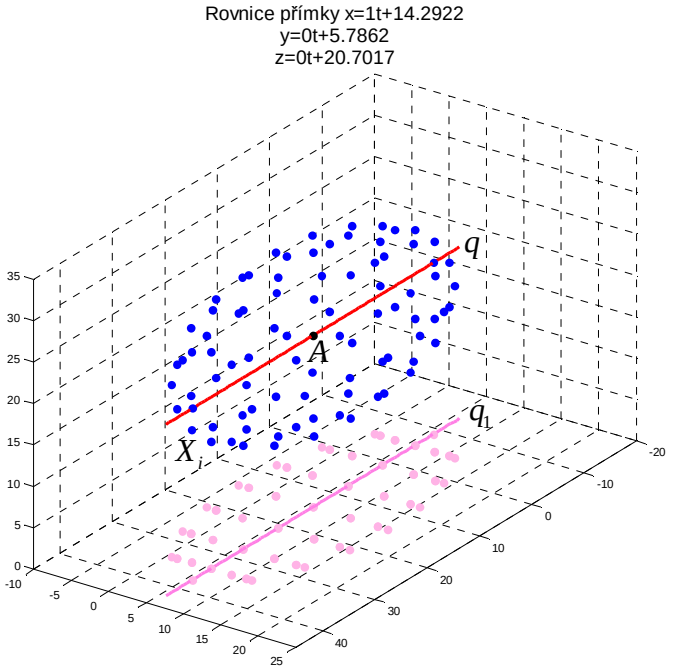
Obrázek 3.36: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu části rotační válcové plochy, osa symetrie rovnoběžná s osou z



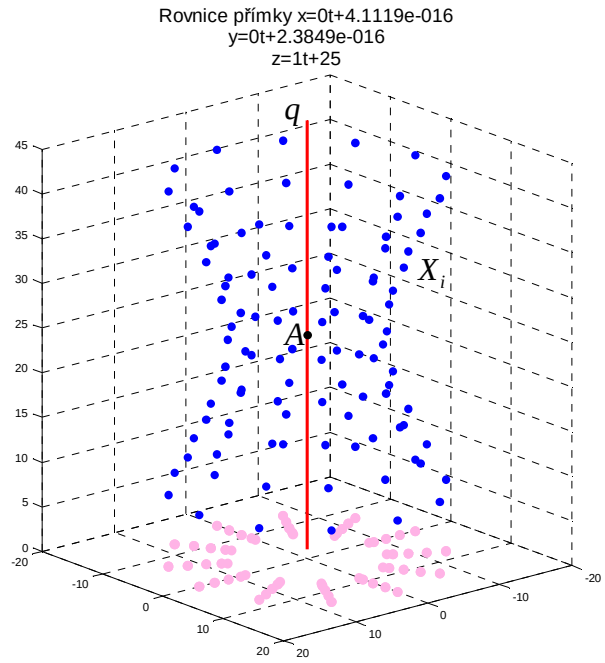
Obrázek 3.37: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu části rotační válcové plochy, osa symetrie rovnoběžná s rovinou (x, y)



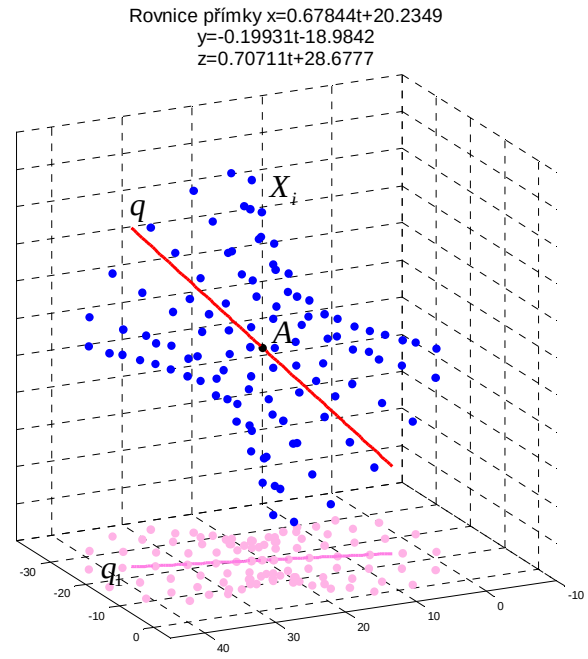
Obrázek 3.38: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu rotačního protáhlého elipsoidu, osa symetrie rovnoběžná s osou z



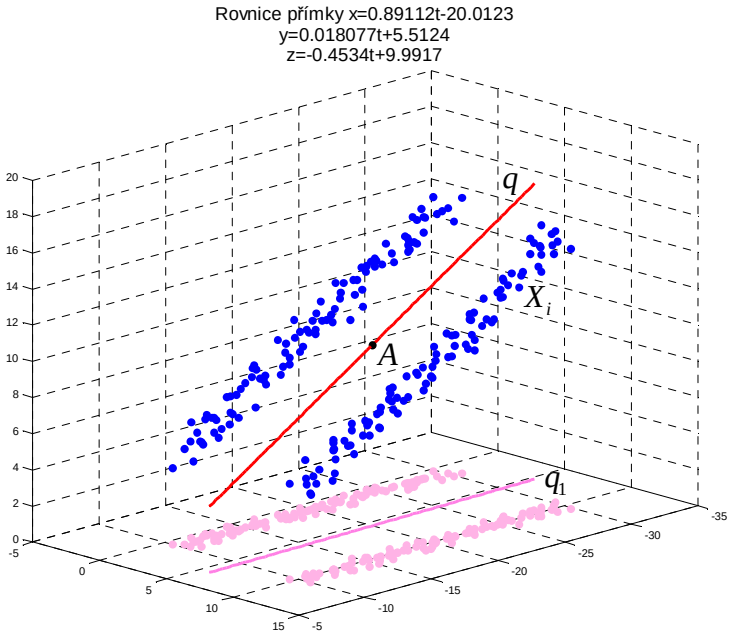
Obrázek 3.39: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu rotačního protáhlého elipsoidu, osa symetrie rovnoběžná s osou x



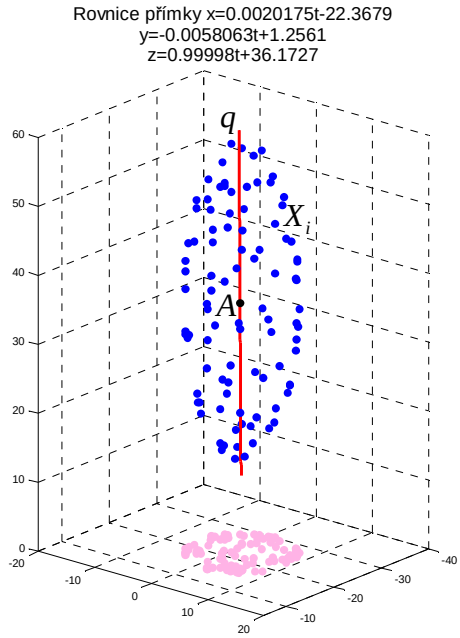
Obrázek 3.40: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu rotačního jednodílného hyperboloidu, osa symetrie je osa z



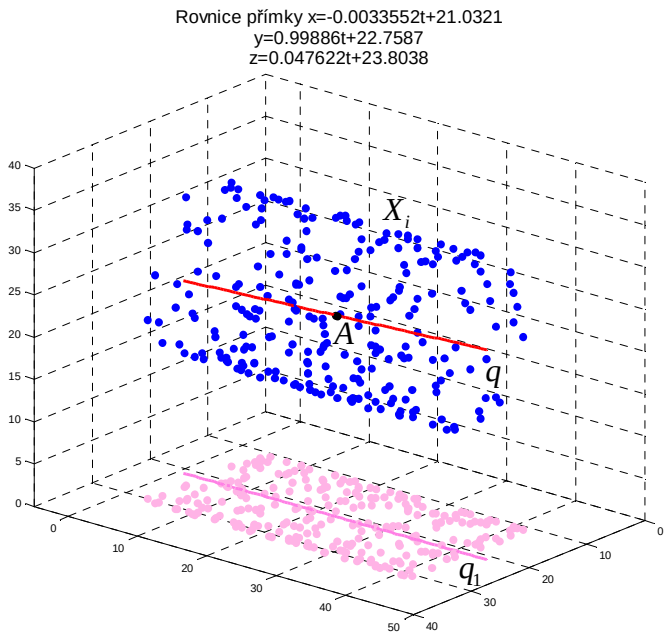
Obrázek 3.41: Ortogonální prokládání dat přímkou v prostoru – body rovnoměrně rozmístěné na povrchu rotačního jednodílného hyperboloidu, osa symetrie je obecná přímka



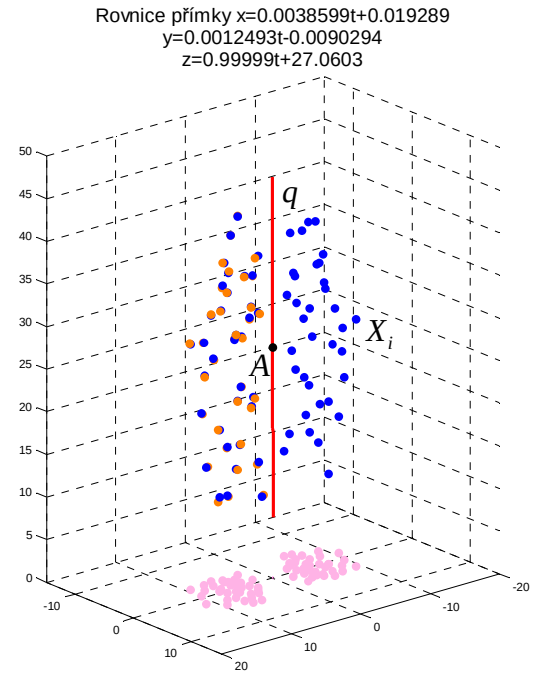
Obrázek 3.42: Ortogonální prokládání dat přímkou v prostoru – osa symetrie bodové množiny, která obsahuje zašuměná data



Obrázek 3.43: Ortogonální prokládání dat přímkou v prostoru – osa symetrie bodové množiny, která obsahuje náhodně generovaná data



Obrázek 3.44: Ortogonální prokládání dat přímkou v prostoru – osa symetrie bodové množiny, která obsahuje náhodně generovaná data



Obrázek 3.45: Ortogonální prokládání dat přímkou v prostoru – zobrazení poloviny bodů v osové symetrii o ose v nalezené přímce

V případové studii v eukleidovské rovině E_2 i v eukleidovském prostoru E_3 jsme popisovali počítačově generované množiny. Záměrně jsme v těchto případových analýzách pracovali s řídkými bodovými množinami, aby bylo jejich zobrazování názorné. V praxi je bodová množina velice hustá, někdy je obtížné tyto množiny zobrazit, aby jednotlivé body nesplývaly. Proto příklady s rozsáhlými bodovými množinami necháváme na později. V kapitole 6 budou odvozené metody použity na reálná bodová mračna získaná měřením skutečných povrchů.

3.3.3 Hledání osových symetrií ortogonálním prokládáním dat přímkou

Při analýze bodových množin a hledání jejich osových symetrií jsme využívali vlastnosti ortogonálního prokládání dat přímkou. Doplňme ještě další teorii, která je nutná k objasnění správnosti našich postupů.

Podle věty 3.3 (*Courant-Fisher*) víme, že minimální hodnota kvadratické formy $f(v) = v^{\mathrm{T}} C v$, kde $C \in \mathbb{R}_{n\times n}$ je symetrická matice, je nejmenší vlastní číslo λ_1 této matice C. Normovaný vlastní vektor $s_1 \in \mathbb{R}_n$ příslušný vlastnímu číslu λ_1 minimalizuje kvadratickou formu, neboť víme, že $f(s_1) = s_1^{\mathrm{T}} C s_1 = \lambda_1$. Obdobně toto platí také pro největší vlastní číslo λ_n matice C. Tedy λ_n odpovídá maximální hodnotě kvadratické formy $f(v) = v^{\mathrm{T}} C v$. Pro jednotkový vlastní vektor $s_n \in \mathbb{R}_n$ příslušný vlastnímu číslu λ_n tedy kvadratická forma nabývá maxima, neboť víme, že $f(s_n) = s_n^{\mathrm{T}} C s_n = \lambda_n$.

Podívejme se, co to bude znamenat pro úlohu ortogonálního prokládání dat přímkou. Předpokládejme, že jsme našli určující bod A hledané přímky q z (3.26) případně z (3.60). Víme, že jednotkový vlastní vektor odpovídající nejmenšímu vlastnímu číslu matice M z (3.53) řeší úlohu ortogonálního prokládání přímkou, tj. jedná se o směrový vektor u přímky q. Ponechme stejný určující bod A přímky, ale změňme směrový vektor u této přímky. Pokud bude směrový vektor u v předpisu přímky q v (3.26) případně (3.60) jednotkový vlastní vektor odpovídající naopak největšímu vlastnímu číslu matice M, dostáváme jako výsledek prokládání přímku, pro kterou platí, že součet druhých mocnin vzdáleností zadaných bodů od této přímky je maximální. Tuto přímku značme r. Ze svazku přímek v eukleidovské rovině E_2 (případně trsu přímek v eukleidovském prostoru E_3), které procházejí určujícím bodem A, tedy vybíráme takovou přímku r, pro kterou kvadratická forma σ^2 z (3.53) nabývá maxima.

Uveďme si následující důležité věty, kterými doplníme zkoumanou problematiku.

Věta 3.5 (*Vlastní vektory příslušné různým vlastním číslům*). Nechť $\lambda_1, \lambda_2, ..., \lambda_k$ jsou navzájem různá vlastní čísla matice $C \in \mathbb{R}_{n\times n}$, potom vlastní vektory $s_1, s_2, ..., s_k$ odpovídající těmto vlastním číslům jsou lineárně nezávislé. ■

Větu ponechme bez důkazu, neboť se jedná o jednoduchou matematickou indukci. Čtenář může znění této věty a její důkaz nalézt například v (Harville, 2008).

Věta 3.6 (*Vlastní čísla symetrických matic*). Všechna vlastní čísla symetrické matice $C \in \mathbb{R}_{n\times n}$ jsou reálná. ■

Odkazujeme se opět na literaturu. Větu 3.6 i s jejím důkazem uvádějí mnozí autoři, jmenujme například (Abadir a Magnus, 2005; Bečvář, 2002; Bronson, 1991; Jeffrey, 2010; Laub, 2005). Věty 3.5 a 3.6 zmiňujeme pro úplnost.

Uveďme dále větu o ortogonalitě vlastních vektorů symetrických matic, která je důležitá pro další postupy a odvozování. Podobné znění a důkaz jsou uvedeny například v (Abadir a Magnus, 2005; Bronson, 1991; Dym, 2007; Harville, 2008; Jeffrey, 2010; Venit *a kol.*, 2008; Laub, 2005).

Věta 3.7 (*Vlastní vektory symetrických matic*). Vlastní vektory odpovídající navzájem různým vlastním číslům symetrické matice $C \in \mathbb{R}_{n\times n}$ jsou vzájemně ortogonální. ■

Důkaz: Nechť $s_i \in \mathbb{R}_n$ a $s_j \in \mathbb{R}_n$ jsou vlastní vektory příslušné navzájem různým vlastním číslům λ_i a λ_j, tedy $\lambda_i \neq \lambda_j$, pro $i \neq j$. Podle definice vlastního vektoru, který odpovídá vlastnímu číslu matice C, platí $Cs_i = \lambda_i s_i$ a $Cs_j = \lambda_j s_j$. Vynásobením těchto rovností zleva postupně s_j^{T} a s_i^{T} dostáváme

$$s_j^{\mathrm{T}} C s_i = s_j^{\mathrm{T}} \lambda_i s_i \quad \text{a} \quad s_i^{\mathrm{T}} C s_j = s_i^{\mathrm{T}} \lambda_j s_j,$$

to lze upravit na

$$s_j^{\mathrm{T}} C s_i = \lambda_i s_j^{\mathrm{T}} s_i \quad \text{a} \quad s_i^{\mathrm{T}} C s_j = \lambda_j s_i^{\mathrm{T}} s_j .$$

Víme, že matice C je symetrická, platí tedy $s_j^{\mathrm{T}} C s_i = s_i^{\mathrm{T}} C s_j$. To lze dokázat jednoduše rozepsáním uvedeného vztahu po jednotlivých prvcích matice a vektorů. S využitím této rovnosti můžeme psát

$$\lambda_i s_j^{\mathrm{T}} s_i = \lambda_j s_i^{\mathrm{T}} s_j .$$

Využijme dále vlastnosti skalárního součinu, tj. víme, že platí $s_j^{\mathrm{T}} s_i = s_i^{\mathrm{T}} s_j$. Potom dostáváme

$$\lambda_i s_j^{\mathrm{T}} s_i - \lambda_j s_i^{\mathrm{T}} s_j = (\lambda_i - \lambda_j) s_i^{\mathrm{T}} s_j = 0 .$$

Z předpokladu $\lambda_i \neq \lambda_j$, pro $i \neq j$, ihned plyne, že $s_i^{\mathrm{T}} s_j = 0$, pro $i \neq j$. Tím jsme ukázali, že vlastní vektory s_i a s_j příslušné navzájem různým vlastním číslům λ_i a λ_j jsou ortogonální. ■

Vraťme se k úloze ortogonálního prokládání dat přímkou v eukleidovské rovině E_2. Podle věty 3.7 platí, že vlastní vektory matice M z (3.53) příslušné navzájem různým vlastním číslům jsou ortogonální. V našem případě pracujeme s normovanými vlastními vektory, platí tedy, že jsou dokonce ortonormální. Tuto vlastnost můžeme využít v našich experimentech ortogonálního prokládání dat přímkou v rovině. V případových studiích jsme zkoumali osovou symetrii bodových množin. Pokud byla bodová množina v rovině symetrická podle dvou navzájem kolmých os, našli jsme ortogonálním prokládáním pouze jednu osu, kterou jsme označili za hlavní. Pokud nalezneme také přímku r, jejíž směrový vektor je jednotkovým vlastním vektorem odpovídajícím většímu vlastnímu číslu matice M, dostáváme tak druhou osu symetrie bodové množiny. Ukažme si to na příkladě bodové množiny, která je přesně symetrická podle dvou navzájem kolmých os, viz obrázek 3.46. Obrázky 3.47 a 3.48 ilustrují ověření správnosti nalezených osových symetrií. Zobrazena je vždy polovina bodů v jedné nebo druhé osové symetrii určené přímkou q nebo r jako osou symetrie. Jelikož se jedná o přesně symetrické množiny, zobrazené body (oranžové nebo žluté) se překrývají se vstupními body množiny, to lze opět ověřit početně.

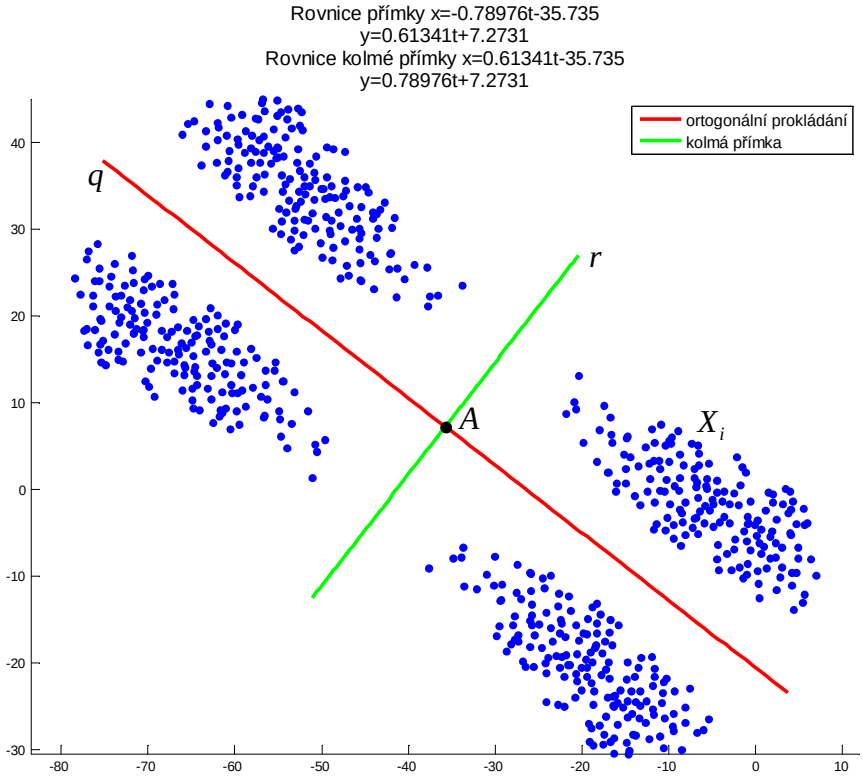
V případě, že bodové množiny nejsou přesně osově symetrické, přímka r se směrovým vektorem, který je jednotkovým vlastním vektorem odpovídajícím většímu vlastnímu číslu matice M, určuje směr, ve kterém je množina bodů nejméně protáhlá. Ukažme to na následujícím příkladě. Body jsou náhodně vygenerovány v oblasti, jejíž hranicí je elipsa. Situaci ilustrují obrázky 3.49 a 3.50.

Přímky q a r tedy představují dvě kolmé přímky se společným bodem A a charakterizují bodovou množinu v rovině, jak bylo popsáno výše. Těmto směrům budeme dále říkat *dva hlavní směry* bodové množiny.

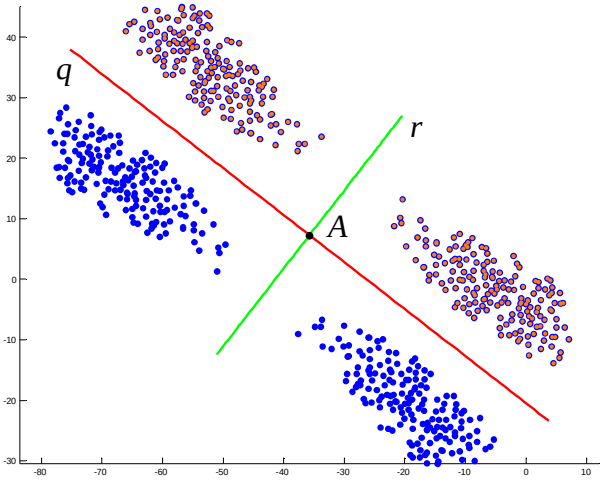
Podívejme se ještě na ortogonální prokládání dat přímkou v eukleidovském prostoru E_3. Opět využijeme větu 3.7, podle které víme, že vlastní vektory matice M z (3.53) příslušné navzájem různým vlastním číslům jsou ortogonální. V našem případě pracujeme s normovanými vlastními vektory, platí tedy, že jsou dokonce ortonormální. Ve výpočtech tedy dostáváme tři vlastní vektory, které určují tři ortogonální směry v prostoru. Jednotkový

vlastní vektor příslušný nejmenšímu vlastnímu číslu řeší úlohu ortogonálního prokládání dat přímkou v prostoru, tedy jedná se o směrový vektor hledané přímky q. Pro přímku r s určujícím bodem A a se směrovým vektorem rovnajícím se jednotkovému vlastnímu vektoru odpovídajícímu naopak největšímu vlastnímu číslu matice M, platí, že součet druhých mocnin vzdáleností zadaných bodů od této přímky je maximální. Přímka s, opět se stejným určujícím bodem A, se směrovým vektorem rovnajícím se jednotkovému vlastnímu vektoru odpovídajícímu prostřednímu vlastnímu číslu, je kolmá na první dvě přímky q a r. Přímky q, r, s tedy představují tři kolmé přímky se společným bodem A a charakterizují bodovou množinu v prostoru, jak bylo popsáno výše. Těmto směrům budeme dále říkat *tři hlavní směry* bodové množiny.

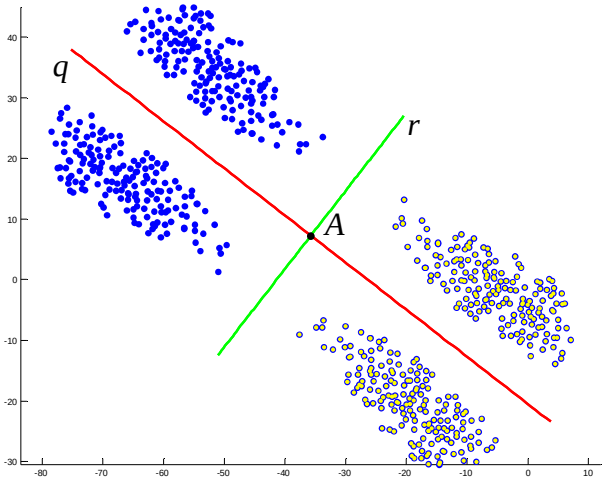
Pravě popsané vlastnosti můžeme využít při hledání osových symetrií v prostoru bodové množiny, která je symetrická podle tří navzájem kolmých os. Ukažme to na příkladě bodové množiny, která vznikla pravidelným navzorkováním povrchu trojosého elipsoidu. Přímky q, r, s určují osy tohoto elipsoidu, viz obrázky 3.51 a 3.52. Na dalších příkladech simulujeme reálnou situaci, nejde tedy o přesně symetrické bodové množiny. Body jsou náhodně vygenerovány v oblasti, jejíž hranicí je povrch trojosého elipsoidu. Nalezené přímky q a r určují směry, ve kterých je množina nejvíce a nejméně protáhlá. Přímka s je na přímky q a r kolmá. V tomto případě jde tedy pouze o přibližné osy symetrie. Situaci můžeme vidět na obrázcích 3.53 a 3.54.



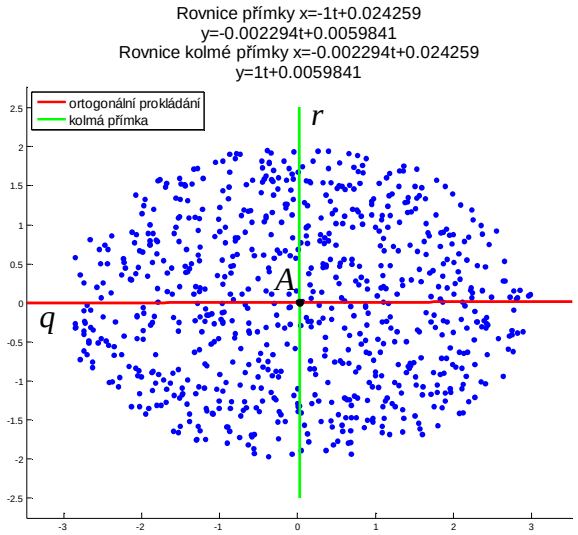
Obrázek 3.46: Ortogonální prokládání dat přímkou v rovině a kolmá přímka – dvě navzájem kolmé osy symetrií bodové množiny, jedna osa (červená) získána metodou ortogonálního prokládání dat přímkou, směrový vektor druhé osy (zelená) odpovídá většímu vlastnímu číslu



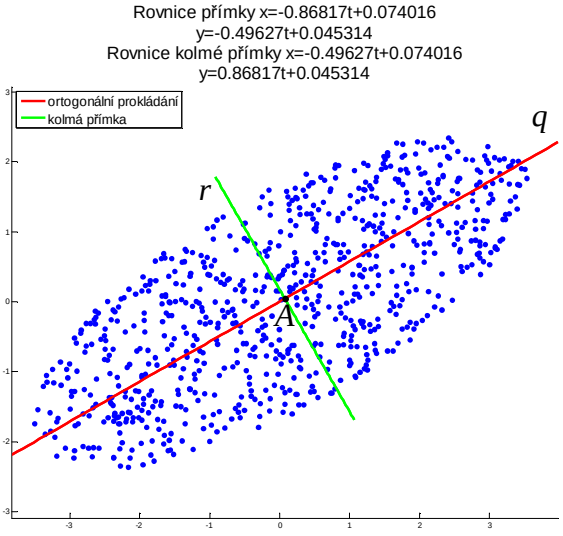
Obrázek 3.47: Ověření správnosti osy symetrie – zobrazení bodů jedné poloroviny určené osou *q* (červenou) v nalezené osové symetrii (obrazy bodů oranžově)



Obrázek 3.48: Ověření správnosti osy symetrie – zobrazení bodů jedné poloroviny určené osou *r* (zelenou) v nalezené osové symetrii (obrazy bodů žlutě)



Obrázek 3.49: Ortogonální prokládání dat přímkou v rovině a kolmá přímka – určují směry, ve kterých je bodová množina nejvíce a nejméně protáhlá

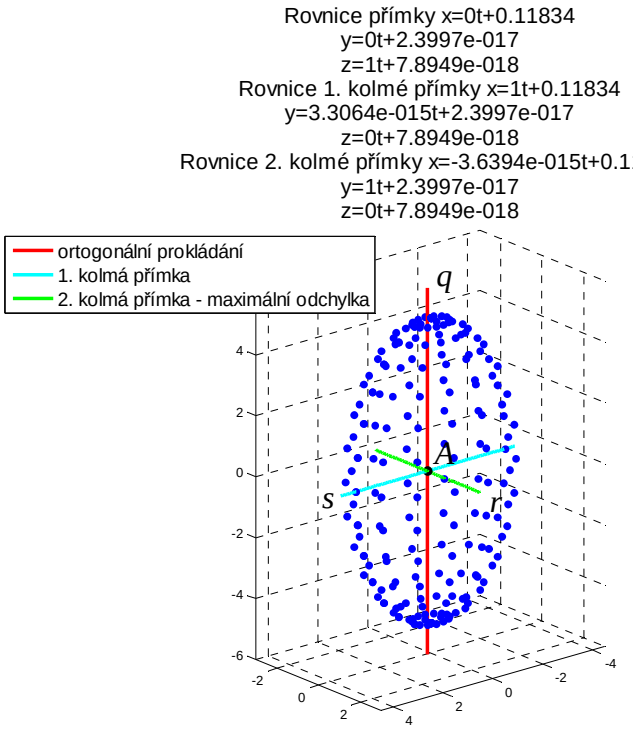


Obrázek 3.50: Ortogonální prokládání dat přímkou v rovině a kolmá přímka – určují směry, ve kterých je bodová množina nejvíce a nejméně protáhlá

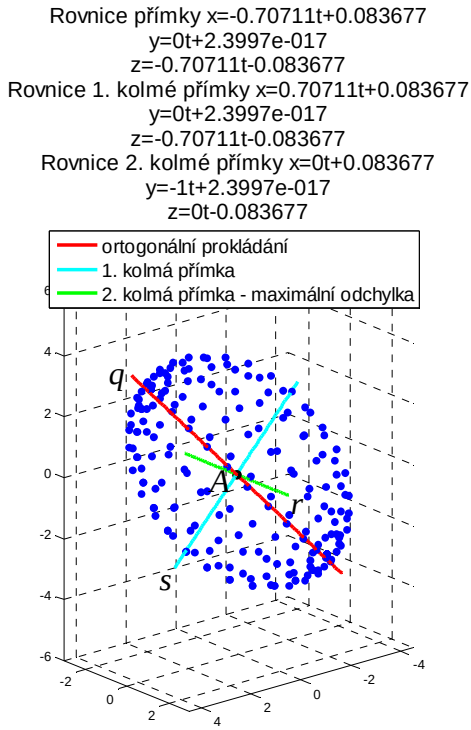
Pojem protáhlosti bodové množiny zatím chápeme spíše intuitivně. K formálnímu zavedení tohoto pojmu nám však poslouží známá statistická metoda analýza hlavních komponent (*Principal Component Analysis*). Ortogonální prokládání dat přímkou s touto metodou úzce souvisí. V oddíle 3.6 tento postup popíšeme a porovnáme jej s odvozenou metodou ortogonálního prokládání dat přímkou v rovině a v prostoru.

Ortogonální prokládání dat přímkou v eukleidovské rovině E_2 a v eukleidovském prostoru E_3 lze využívat k hledání osových symetrií bodových množin. Nutným předpokladem je ovšem znalost, že bodová množina nějakou osovou symetrii splňuje. Hledání os těchto symetrií potřebujeme kvůli zpracování bodové množiny pro další fáze rekonstrukce. Použití

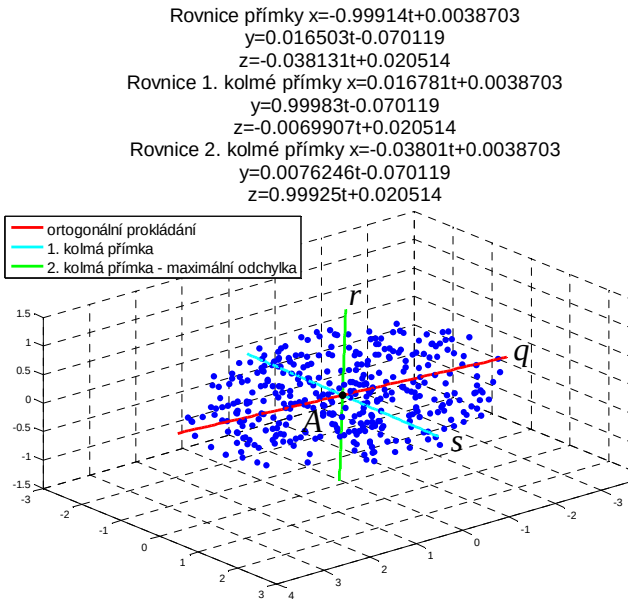
těchto postupů na bodových množinách popisující reálné povrchy bude předloženo v kapitole 6.



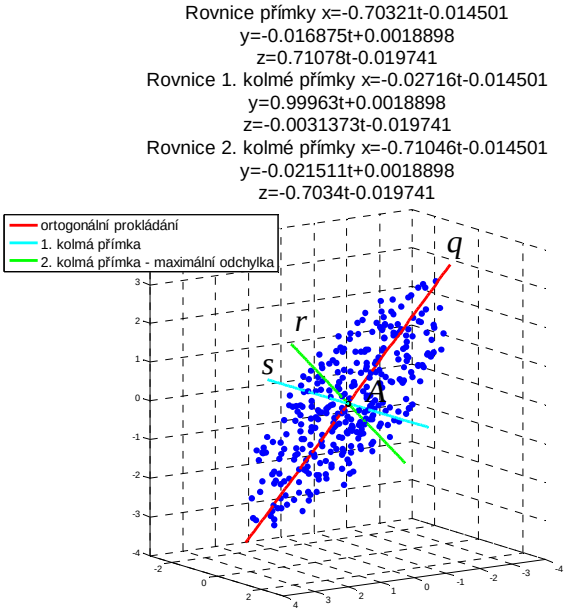
Obrázek 3.51: Ortogonální prokládání dat přímkou v prostoru a další dva kolmé směry – body rovnoměrně rozmístěné na povrchu trojosého elipsoidu ve speciální poloze, přímky určují osy elipsoidu



Obrázek 3.52: Ortogonální prokládání dat přímkou v prostoru a další dva kolmé směry – body rovnoměrně rozmístěné na povrchu trojosého elipsoidu v obecnější poloze, přímky určují osy elipsoidu



Obrázek 3.53: Ortogonální prokládání dat přímkou v prostoru a další dva kolmé směry – určují tři hlavní směry bodové množiny



Obrázek 3.54: Ortogonální prokládání dat přímkou v prostoru a další dva kolmé směry – určují tři hlavní směry bodové množiny

3.4 Aproximace funkce dvou proměnných metodou nejmenších čtverců

Aproximaci funkcí metodou nejmenších čtverců zavedeme obdobně jako v oddíle 3.2 i pro funkce více proměnných. S ohledem na aplikace se omezíme jen na případ funkcí dvou proměnných. Opět tedy půjde o zpracování nepřesných dat, například výsledků nějakého měření, které je zatíženo chybou. Data tedy reprezentují měření funkční závislosti, kde chyba je obsažena pouze v závisle proměnné (v pozorování). Aproximace metodou nejmenších čtverců slouží právě k eliminaci této chyby. Stejně jako při aproximaci funkce jedné proměnné i nyní předpokládáme, že chyby v různých bodech jsou na sobě navzájem nezávislé.

Hledáme tedy aproximující funkci dvou proměnných danými hodnotami získanými nějakým měřením, přičemž předem víme, o jakou se jedná funkční závislost, tj. známe přesnou funkci, kterou nyní pomocí konečného množství hodnot navíc zatížených chybami aproximujeme. Nebudeme aproximaci dat funkcemi dvou proměnných studovat v obecném případě, ale omezíme se na speciální příklad. Předpokládáme, že přesná funkce je polynomická. Při aproximaci používáme opět známé kritérium, čímž je minimalizace součtu druhých mocnin odchylek funkčních hodnot a naměřených hodnot.

Mějme dánu množinu $\mathcal{X}$ $n+1$ bodů $[x_i, y_i, z_i]$, $i = 0,1,...,n$ v eukleidovském prostoru E_3, zapisovat budeme $\mathcal{X} = \{[x_i, y_i, z_i]\}_{i=0}^{n}$ nebo zkráceně $\mathcal{X} = \{X_i\}_{i=0}^{n}$. Dále je dána množina $(j+1)\times(k+1)$ reálných funkcí φ_{pq} dvou reálných proměnných, $p = 0,1,...,j$, $q = 0,1,...,k$, kde $(j+1)\times(k+1) \le (n+1)$, definovaných alespoň ve všech bodech (x_i, y_i). Za třídu aproximujících funkcí vezměme množinu lineárních kombinací

$$\left\{\begin{array}{ccccccccc} & a_{00}\varphi_{00}(x,y) & + & a_{01}\varphi_{01}(x,y) & + & \cdots & + & a_{0k}\varphi_{0k}(x,y) & + \\ + & a_{10}\varphi_{10}(x,y) & + & a_{11}\varphi_{11}(x,y) & + & \cdots & + & a_{1k}\varphi_{1k}(x,y) & + \\ + & \vdots & & \vdots & & & & \vdots & + \\ + & a_{j0}\varphi_{j0}(x,y) & + & a_{j1}\varphi_{j1}(x,y) & + & \cdots & + & a_{jk}\varphi_{jk}(x,y); & \\ & & & a_{pq} \in \mathbb{R}, p = 0,1,...,j, q = 0,1,...,k & & & & & \end{array}\right\}. \tag{3.61}$$

Z této množiny vybereme takovou funkci

$$\varphi(x,y) = \sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}\varphi_{pq}(x,y), \tag{3.62}$$

pro kterou funkce

$$H(a_{00}, a_{01},...,a_{0k}, a_{10}, a_{11},...,a_{1k},...,a_{j0}, a_{j1},...,a_{jk}) = \sum_{i=0}^{n}\left(\varphi(x_i, y_i) - z_i\right)^2, \tag{3.63}$$

nabývá minimální hodnoty. Kdybychom znali přesnou funkci, kterou nyní pomocí konečného množství hodnot navíc zatížených chybami aproximujeme, předpokládáme, že pro nějaké neznámé $j,k \in \mathbb{N}$ lze tuto funkci vyjádřit jako konečnou lineární kombinaci množiny funkcí

φ_{pq}, $p=0,1,...,j$, $q=0,1,...,k$. Hledání minima funkce H z (3.63) závislé na parametrech $a_{00},a_{01},...,a_{0j}$, $a_{10},a_{11},...,a_{1k},...,a_{j0},a_{j1},...,a_{jk}$ budeme opět řešit standardním postupem matematické analýzy. Vypočteme parciální derivace funkce H podle jednotlivých parametrů a ty položíme rovny nule, tj.

$$\frac{\partial H\left(a_{00},a_{01},...,a_{0k},a_{10},a_{11},...,a_{1k},...,a_{j0},a_{j1},...,a_{jk}\right)}{\partial a_{pq}}=2\sum_{i=0}^{n}\left(\varphi(x_i,y_i)-z_i\right)\varphi_{pq}(x_i,y_i)=0, \tag{3.64}$$

kde $p=0,1,...,j$, $q=0,1,...,k$. Hledáme tedy lokální minimum funkce H, nutnou podmínkou pro jeho existenci je právě splnění rovnosti (3.64). Po úpravě výrazu (3.64) získáme soustavu $(j+1)\times(k+1)$ lineárních rovnic pro neznámé $a_{00},a_{01},...,a_{0k},a_{10},a_{11},...,a_{1k},...,a_{j0}$, $a_{j1},...,a_{jk}$, tedy

$$\begin{aligned}
&a_{00}\sum_{i=0}^{n}\varphi_{00}(x_i,y_i)\varphi_{pq}(x_i,y_i)+a_{01}\sum_{i=0}^{n}\varphi_{01}(x_i,y_i)\varphi_{pq}(x_i,y_i)+\ldots+a_{0k}\sum_{i=0}^{n}\varphi_{0k}(x_i,y_i)\varphi_{pq}(x_i,y_i)+\\
&+a_{10}\sum_{i=0}^{n}\varphi_{10}(x_i,y_i)\varphi_{pq}(x_i,y_i)+a_{11}\sum_{i=0}^{n}\varphi_{11}(x_i,y_i)\varphi_{pq}(x_i,y_i)+\ldots+a_{1k}\sum_{i=0}^{n}\varphi_{1k}(x_i,y_i)\varphi_{pq}(x_i,y_i)+\\
&+\quad\cdots\qquad\qquad\cdots\qquad\qquad\cdots\qquad+\\
&+a_{j0}\sum_{i=0}^{n}\varphi_{j0}(x_i,y_i)\varphi_{pq}(x_i,y_i)+a_{j1}\sum_{i=0}^{n}\varphi_{j1}(x_i,y_i)\varphi_{pq}(x_i,y_i)+\ldots+a_{jk}\sum_{i=0}^{n}\varphi_{jk}(x_i,y_i)\varphi_{pq}(x_i,y_i)\;=\\
&=\sum_{i=0}^{n}z_i\varphi_{pq}(x_i,y_i),
\end{aligned} \tag{3.65}$$

kde $p=0,1,...,j$, $q=0,1,...,k$. Soustavu lineárních rovnic (3.65) můžeme vyjádřit pomocí maticového zápisu stejně jako v případě pro jednorozměrné funkce v oddíle 3.2. Kvůli komplikovanosti zápisu zaveďme jednoduchou symboliku. Prvky matice A zapíšeme zjednodušeně, tedy např. prvek $\sum_{i=0}^{n}\varphi_{jk}(x_i,y_i)\varphi_{jk}(x_i,y_i)$ nahradíme symbolem $\varphi_{jk}^{i}\varphi_{jk}^{i}$. Obdobně zjednodušíme také zápis pravých stran jednotlivých lineárních rovnic, tedy např. prvek $\sum_{i=0}^{n}z_i\varphi_{jk}(x_i,y_i)$ nahradíme symbolem $z_i\varphi_{jk}^{i}$. Dostáváme tedy následující tvar

$$A=\begin{pmatrix}
\varphi_{00}^{i}\varphi_{00}^{i} & \varphi_{01}^{i}\varphi_{00}^{i}\cdots & \varphi_{0k}^{i}\varphi_{00}^{i} & \varphi_{10}^{i}\varphi_{00}^{i} & \varphi_{11}^{i}\varphi_{00}^{i}\cdots & \varphi_{1k}^{i}\varphi_{00}^{i}\cdots & \varphi_{j0}^{i}\varphi_{00}^{i} & \varphi_{j1}^{i}\varphi_{00}^{i}\cdots & \varphi_{jk}^{i}\varphi_{00}^{i}\\
\varphi_{00}^{i}\varphi_{01}^{i} & \varphi_{01}^{i}\varphi_{01}^{i}\cdots & \varphi_{0k}^{i}\varphi_{01}^{i} & \varphi_{10}^{i}\varphi_{01}^{i} & \varphi_{11}^{i}\varphi_{01}^{i}\cdots & \varphi_{1k}^{i}\varphi_{01}^{i}\cdots & \varphi_{j0}^{i}\varphi_{01}^{i} & \varphi_{j1}^{i}\varphi_{01}^{i}\cdots & \varphi_{jk}^{i}\varphi_{01}^{i}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\varphi_{00}^{i}\varphi_{0k}^{i} & \varphi_{01}^{i}\varphi_{0k}^{i}\cdots & \varphi_{0k}^{i}\varphi_{0k}^{i} & \varphi_{10}^{i}\varphi_{0k}^{i} & \varphi_{11}^{i}\varphi_{0k}^{i}\cdots & \varphi_{1k}^{i}\varphi_{0k}^{i}\cdots & \varphi_{j0}^{i}\varphi_{0k}^{i} & \varphi_{j1}^{i}\varphi_{0k}^{i}\cdots & \varphi_{jk}^{i}\varphi_{0k}^{i}\\
\varphi_{00}^{i}\varphi_{10}^{i} & \varphi_{01}^{i}\varphi_{10}^{i}\cdots & \varphi_{0k}^{i}\varphi_{10}^{i} & \varphi_{10}^{i}\varphi_{10}^{i} & \varphi_{11}^{i}\varphi_{10}^{i}\cdots & \varphi_{1k}^{i}\varphi_{10}^{i}\cdots & \varphi_{j0}^{i}\varphi_{10}^{i} & \varphi_{j1}^{i}\varphi_{10}^{i}\cdots & \varphi_{jk}^{i}\varphi_{10}^{i}\\
\varphi_{00}^{i}\varphi_{11}^{i} & \varphi_{01}^{i}\varphi_{11}^{i}\cdots & \varphi_{0k}^{i}\varphi_{11}^{i} & \varphi_{10}^{i}\varphi_{11}^{i} & \varphi_{11}^{i}\varphi_{11}^{i}\cdots & \varphi_{1k}^{i}\varphi_{11}^{i}\cdots & \varphi_{j0}^{i}\varphi_{11}^{i} & \varphi_{j1}^{i}\varphi_{11}^{i}\cdots & \varphi_{jk}^{i}\varphi_{11}^{i}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\varphi_{00}^{i}\varphi_{1k}^{i} & \varphi_{01}^{i}\varphi_{1k}^{i}\cdots & \varphi_{0k}^{i}\varphi_{1k}^{i} & \varphi_{10}^{i}\varphi_{1k}^{i} & \varphi_{11}^{i}\varphi_{1k}^{i}\cdots & \varphi_{1k}^{i}\varphi_{1k}^{i}\cdots & \varphi_{j0}^{i}\varphi_{1k}^{i} & \varphi_{j1}^{i}\varphi_{1k}^{i}\cdots & \varphi_{jk}^{i}\varphi_{1k}^{i}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\varphi_{00}^{i}\varphi_{j0}^{i} & \varphi_{01}^{i}\varphi_{j0}^{i}\cdots & \varphi_{0k}^{i}\varphi_{j0}^{i} & \varphi_{10}^{i}\varphi_{j0}^{i} & \varphi_{11}^{i}\varphi_{j0}^{i}\cdots & \varphi_{1k}^{i}\varphi_{j0}^{i}\cdots & \varphi_{j0}^{i}\varphi_{j0}^{i} & \varphi_{j1}^{i}\varphi_{j0}^{i}\cdots & \varphi_{jk}^{i}\varphi_{j0}^{i}\\
\varphi_{00}^{i}\varphi_{j1}^{i} & \varphi_{01}^{i}\varphi_{j1}^{i}\cdots & \varphi_{0k}^{i}\varphi_{j1}^{i} & \varphi_{10}^{i}\varphi_{j1}^{i} & \varphi_{11}^{i}\varphi_{j1}^{i}\cdots & \varphi_{1k}^{i}\varphi_{j1}^{i}\cdots & \varphi_{j0}^{i}\varphi_{j1}^{i} & \varphi_{j1}^{i}\varphi_{j1}^{i}\cdots & \varphi_{jk}^{i}\varphi_{j1}^{i}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\varphi_{00}^{i}\varphi_{jk}^{i} & \varphi_{01}^{i}\varphi_{jk}^{i}\cdots & \varphi_{0k}^{i}\varphi_{jk}^{i} & \varphi_{10}^{i}\varphi_{jk}^{i} & \varphi_{11}^{i}\varphi_{jk}^{i}\cdots & \varphi_{1k}^{i}\varphi_{jk}^{i}\cdots & \varphi_{j0}^{i}\varphi_{jk}^{i} & \varphi_{j1}^{i}\varphi_{jk}^{i}\cdots & \varphi_{jk}^{i}\varphi_{jk}^{i}
\end{pmatrix},\quad b=\begin{pmatrix}
z_i\varphi_{00}^{i}\\ z_i\varphi_{01}^{i}\\ \vdots\\ z_i\varphi_{0k}^{i}\\ z_i\varphi_{10}^{i}\\ z_i\varphi_{11}^{i}\\ \vdots\\ z_i\varphi_{1k}^{i}\\ \vdots\\ z_i\varphi_{j0}^{i}\\ z_i\varphi_{j1}^{i}\\ \vdots\\ z_i\varphi_{jk}^{i}
\end{pmatrix},\quad x=\begin{pmatrix}
a_{00}\\ a_{01}\\ \vdots\\ a_{0k}\\ a_{10}\\ a_{11}\\ \vdots\\ a_{1k}\\ \vdots\\ a_{j0}\\ a_{j1}\\ \vdots\\ a_{jk}
\end{pmatrix}, \tag{3.66}$$

kde $Ax = b$. Je-li matice A regulární, má soustava $Ax = b$ právě jedno řešení.

K soustavě (3.66) lze dojít rovněž alternativní úvahou. Pokud dosadíme do předpisu (3.62) souřadnice jednotlivých bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$, dostaneme přeurčenou soustavu $n+1$ rovnic, tedy soustavu rovnic, která obsahuje více (určujících) rovnic než neznámých. Pokud připouštíme případ $(j+1)\times(k+1) = (n+1)$, dostaneme soustavu $n+1$ rovnic pro $n+1$ neznámých. Soustavu $n+1$ rovnic

$$\begin{aligned}
&\varphi(x_0, y_0) = z_0 = \sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}\varphi_{pq}(x_0, y_0),\\
&\varphi(x_1, y_1) = z_1 = \sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}\varphi_{pq}(x_1, y_1),\\
&\vdots\\
&\varphi(x_n, y_n) = z_n = \sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}\varphi_{pq}(x_n, y_n),
\end{aligned} \tag{3.67}$$

zapišme symbolicky jako $Bx = z$. Opět použijeme zjednodušeného zápisu, tj. např. prvek $\varphi_{jk}(x_i, y_i)$ nahradíme symbolem φ_{jk}^{i}, tedy

$$B = \begin{pmatrix}
\varphi_{00}^{0} & \varphi_{01}^{0} & \cdots & \varphi_{0k}^{0} & \varphi_{10}^{0} & \varphi_{11}^{0} & \cdots & \varphi_{1k}^{0} & \cdots & \varphi_{j0}^{0} & \varphi_{j1}^{0} & \cdots & \varphi_{jk}^{0}\\
\varphi_{00}^{1} & \varphi_{01}^{1} & \cdots & \varphi_{0k}^{1} & \varphi_{10}^{1} & \varphi_{11}^{1} & \cdots & \varphi_{1k}^{1} & \cdots & \varphi_{j0}^{1} & \varphi_{j1}^{1} & \cdots & \varphi_{jk}^{1}\\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots\\
\varphi_{00}^{n} & \varphi_{01}^{n} & \cdots & \varphi_{0k}^{n} & \varphi_{10}^{n} & \varphi_{11}^{n} & \cdots & \varphi_{1k}^{n} & \cdots & \varphi_{j0}^{n} & \varphi_{j1}^{n} & \cdots & \varphi_{jk}^{n}
\end{pmatrix}, z = \begin{pmatrix} z_0\\ z_1\\ \vdots\\ z_n \end{pmatrix},$$

$$x = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0k} & a_{10} & a_{11} & \cdots & a_{1k} & \cdots & a_{j0} & a_{j1} & \cdots & a_{jk} \end{pmatrix}^{\mathrm{T}}. \tag{3.68}$$

Jsou-li sloupce matice B lineárně nezávislé, má soustava (3.68) právě jedno řešení. Soustavu rovnic (3.66) získáme ze soustavy (3.68) vynásobením obou stran rovnice zleva maticí B^{T}, dostáváme tedy $B^{\mathrm{T}}Bx = B^{\mathrm{T}}z$, kde $A = B^{\mathrm{T}}B$ a $b = B^{\mathrm{T}}z$. Rovněž můžeme vektor x neznámých koeficientů vyjádřit jako $x = \left(B^{\mathrm{T}}B\right)^{-1}B^{\mathrm{T}}z$ za předpokladu, že existuje inverzní matice $\left(B^{\mathrm{T}}B\right)^{-1}$.

Existence a jednoznačnost řešení soustavy (3.66), které minimalizuje funkci (3.63) by se dokazovaly obdobně jako v případě jednorozměrných funkcí v oddíle 3.2. Podrobné důkazy uvádět nebudeme, ovšem soustředíme se na implementační ověření správnosti našich postupů pro speciální případ systému $(j+1)\times(k+1)$ reálných funkcí φ_{pq} dvou reálných proměnných, $p = 0,1,...,j$, $q = 0,1,...,k$, jak bude ukázáno dále. Stejně bychom mohli také rozebírat numerickou stabilitu výpočtů při volbě různých rozkladů matice použité soustavy rovnic. Výpočetní prostředí MATLAB, ve kterém veškeré výpočty realizujeme, tyto postupy implementuje, pojednání o nich je mimo rámec této studie.

Podívejme se nyní na případ, kdy je funkce $\varphi(x,y)$ polynomem dvou reálných proměnných, tj. funkce $\varphi(x,y)$ je lineární kombinací polynomů dvou proměnných tvaru $x^p y^q$, $p=0,1,...,j$, $q=0,1,...,k$ s reálnými koeficienty a_{pq}, $p=0,1,...,j$, $q=0,1,...,k$. Předpokládáme tedy, že přesná funkce je polynomem dvou proměnných stupně nejvýše $s \le \min(j,k)$, kvůli chybám v měření však získáme pouze její aproximaci. Přepišme (3.62) následovně

$$\varphi(x,y)=\sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}x^p y^q = \tag{3.69}$$
$$= a_{00}+a_{01}y+\ldots+a_{0k}y+a_{10}x+a_{11}xy+\ldots a_{1k}xy^k+\ldots+a_{j0}x^j+a_{j1}x^j y+\ldots+a_{jk}x^j y^k.$$

Víme, že funkcí (3.69) máme aproximovat polynom dvou proměnných stupně nejvýše s, to znamená, že některé koeficienty jsou v předpisu funkce (3.69) rovny nule. Vystačíme tedy s tím, že funkci $\varphi(x,y)$ budeme uvažovat ve tvaru

$$\varphi(x,y)=\sum_{\substack{p,q\\ p+q\le s}} a_{pq}x^p y^q\,, \tag{3.70}$$

kde $s \le \min(j,k)$. Jestliže v předpisu funkce (3.70) zvolíme $s=1$, dostáváme speciální případ a to polynom dvou proměnných stupně jedna tedy rovinu. Tento případ nás bude nejvíce zajímat.

Pro lepší představu funkci (3.70) ještě rozepišme následovně

$$\varphi(x,y)=\sum_{p=0}^{s}\sum_{q=0}^{s-p} a_{pq}x^p y^q\,, \tag{3.71}$$

kde $s \le \min(j,k)$.

Použijme nyní předpis funkce $\varphi(x,y)$ z (3.69) a dosaďme jej do (3.63)

$$H(a_{00},a_{01},...,a_{0k},a_{10},a_{11},...,a_{1k},...,a_{j0},a_{j1},...,a_{jk})=\sum_{i=0}^{n}\left(\sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}x_i^p y_i^q - z_i\right)^2 = \tag{3.72}$$
$$=\sum_{i=0}^{n}\left(a_{00}+a_{01}y_i+\ldots+a_{0k}y_i+a_{10}x_i+a_{11}x_i y_i+\ldots a_{1k}x_i y_i^k+\ldots+a_{j0}x_i^j+a_{j1}x_i^j y_i+\ldots+a_{jk}x_i^j y_i^k-z_i\right)^2.$$

Hledání minima funkce H z (3.72) závislé na parametrech $a_{00},a_{01},...,a_{0j}$, $a_{10},a_{11},...,a_{1k},...,a_{j0},a_{j1},...,a_{jk}$ budeme řešit stejným postupem jako v obecném případě. Nejprve zapišme parciální derivace funkce H podle jednotlivých parametrů, které pokládáme rovny nule, tj.

$$\frac{\partial H\left(a_{00},a_{01},...,a_{0k},a_{10},a_{11},...,a_{1k},...,a_{j0},a_{j1},...,a_{jk}\right)}{\partial a_{pq}}=2\sum_{i=0}^{n}\left(\sum_{p=0}^{j}\sum_{q=0}^{k} a_{pq}x_i^p y_i^q - z_i\right)x_i^p y_i^q = 0\,, \tag{3.73}$$

kde $p = 0,1,...,j$, $q = 0,1,...,k$. Po úpravě výrazu (3.73) získáme soustavu $(j+1)\times(k+1)$ lineárních rovnic pro neznámé $a_{00}, a_{01}, ..., a_{0k}, a_{10}, a_{11}, ..., a_{1k}, ..., a_{j0}, a_{j1}, ..., a_{jk}$, tj.

$$
\begin{aligned}
& a_{00}\sum_{i=0}^{n} x_i^p y_i^q \;+\; a_{01}\sum_{i=0}^{n} x_i^p y_i^{q+1} \;+\; \dots \;+\; a_{0k}\sum_{i=0}^{n} x_i^p y_i^{q+k} \;+ \\
& +\; a_{10}\sum_{i=0}^{n} x_i^{p+1} y_i^q \;+\; a_{11}\sum_{i=0}^{n} x_i^{p+1} y_i^{q+1} \;+\; \dots \;+\; a_{1k}\sum_{i=0}^{n} x_i^{p+1} y_i^{q+k} \;+ \\
& +\qquad \cdots \qquad\qquad\qquad\qquad\qquad \cdots \qquad + \\
& +\; a_{j0}\sum_{i=0}^{n} x_i^{p+j} y_i^q \;+\; a_{j1}\sum_{i=0}^{n} x_i^{p+j} y_i^{q+1} \;+\; \dots \;+\; a_{jk}\sum_{i=0}^{n} x_i^{p+j} y_i^{q+k} \qquad = \sum_{i=0}^{n} z_i x_i^p y_i^q,
\end{aligned}
\tag{3.74}
$$

kde $p = 0,1,...,j$, $q = 0,1,...,k$. Obdobně bychom mohli soustavu lineárních rovnic rozepsat jako v případě (3.66) pomocí matic.

Zabývejme se nyní speciálním případem funkce $\varphi(x,y)$ nadefinované v (3.70) a (3.71) a zvolme $s = 1$. To znamená, že danou množinu bodů $\mathcal{X} = \{[x_i, y_i, z_i]\}_{i=0}^{n}$ budeme nyní aproximovat polynomem dvou proměnných stupně jedna tedy rovinou. Polynomem dvou proměnných stupně jedna budeme rozumět funkci

$$\varphi(x,y) = a_{00} + a_{01}y + a_{10}x, \tag{3.75}$$

z toho dále plyne, že funkce H z (3.72) vypadá následovně

$$H(a_{00}, a_{01}, a_{10}) = \sum_{i=0}^{n} \left(a_{00} + a_{01}y_i + a_{10}x_i - z_i\right)^2 \tag{3.76}$$

a parciální derivace funkce H podle jednotlivých parametrů a_{00}, a_{01}, a_{10} jsou

$$\frac{\partial H\left(a_{00}, a_{01}, a_{10}\right)}{\partial a_{pq}} = 2\sum_{i=0}^{n} \left(a_{00} + a_{01}y_i + a_{10}x_i - z_i\right) x_i^p y_i^q \,. \tag{3.77}$$

Položíme-li parciální derivace rovny nule, získáme soustavu tentokrát tří lineárních rovnic pro neznámé a_{00}, a_{01}, a_{10}, tj.

$$a_{00}\sum_{i=0}^{n} x_i^p y_i^q + a_{01}\sum_{i=0}^{n} x_i^p y_i^{q+1} + a_{10}\sum_{i=0}^{n} x_i^{p+1} y_i^q = \sum_{i=0}^{n} z_i x_i^p y_i^q \,, \tag{3.78}$$

pro $p = 0, q = 0$; $p = 0$, $q = 1$ a $p = 1$, $q = 0$. Příslušná soustava tří lineárních rovnic zapsaná maticově

(3.79)
$$A=\begin{pmatrix} \sum_{i=0}^{n}1 & \sum_{i=0}^{n}y_i & \sum_{i=0}^{n}x_i \\ \sum_{i=0}^{n}y_i & \sum_{i=0}^{n}y_i^2 & \sum_{i=0}^{n}x_iy_i \\ \sum_{i=0}^{n}x_i & \sum_{i=0}^{n}x_iy_i & \sum_{i=0}^{n}x_i^2 \end{pmatrix},\ b=\begin{pmatrix} \sum_{i=0}^{n}z_i \\ \sum_{i=0}^{n}y_iz_i \\ \sum_{i=0}^{n}x_iz_i \end{pmatrix},\ x=\begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \end{pmatrix},$$

kde $Ax=b$.

Pro lepší představu vše ukažme na příkladech. Vstupní data budeme aproximovat nejdříve polynomem dvou proměnných stupně jedna tedy rovinou. Na menším počtu vstupních bodů si nejdříve znázorníme princip této metody prokládání dat a podrobně rozepíšeme jednotlivé výpočty. Dále předvedeme metodu nejmenších čtverců v praktické aplikaci, proložíme rovinou rozsáhlejší vstupní bodovou množinu. Aproximaci metodou nejmenších čtverců také demonstrujeme na příkladech prokládání dat polynomem dvou proměnných vyšších stupňů.

Příklad 3.3: *PROKLÁDÁNÍ DAT POLYNOMEM DVOU PROMĚNNÝCH METODOU NEJMENŠÍCH ČTVERCŮ.* Popsanou metodu nyní ukážeme na dvou příkladech prokládání dat v prostoru rovinou a na dvou příkladech prokládání dat v prostoru polynomem dvou proměnných druhého stupně. Výpočet rozepíšeme pouze v případě prokládání dat rovinou. Mějme dánu množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i,y_i,z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3 a předpokládejme, že charakter měřené závislosti je možno s dostatečnou přesností vystihnout polynomem dvou proměnných prvního stupně.

Na prvním případě demonstrujme princip metody nejmenších čtverců na menším počtu vstupních bodů. Experimentální data pro aproximaci jsou uvedena v tabulce 3.4, přičemž se nejedná o data, která by odpovídala nějakému skutečnému měření. Dále je naznačen postup výpočtu řešením popsané soustavy rovnic. Rozklad matice soustavy pro zvýšení numerické stability výpočtu by se prováděl stejně jako v případě pro jednorozměrné funkce. (Číselné hodnoty jsou při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Na obrázku 3.55 je znázorněna výsledná aproximující rovina v pohledu ve směru této roviny, na obrázku 3.56 je potom znázorněna situace v prostoru. Obrázek 3.57 již ukazuje praktickou úlohu, kdy body odpovídají nějakému měření zatíženému chybami. Víme, že v tomto případě měla přesná funkce předpis $f(x,y)=2-0{,}5y-0{,}1x$, výsledná aproximující rovina má velmi podobný předpis, jak je popsáno v obrázku. Souřadnice bodů v tomto v případě nevypisujeme, protože se jedná již o rozsáhlejší data.

Na obrázcích 3.58 a) – b) můžeme vidět výsledky prokládání dat v prostoru polynomem dvou proměnných druhého stupně. V prvním případě má přesná funkce předpis $f(x,y)=-3-0{,}04x^2-0{,}1y^2$, grafem této funkce je plocha hyperbolického paraboloidu. Vidíme, že metodou nejmenších čtverců získáváme plochu velice blíz-

kou. Na druhém obrázku má přesná funkce předpis $f(x,y) = -4 + 0{,}1x^2 + 0{,}1y^2$. V tomto případě popisuje rovnice plochu rotačního paraboloidu o ose z.

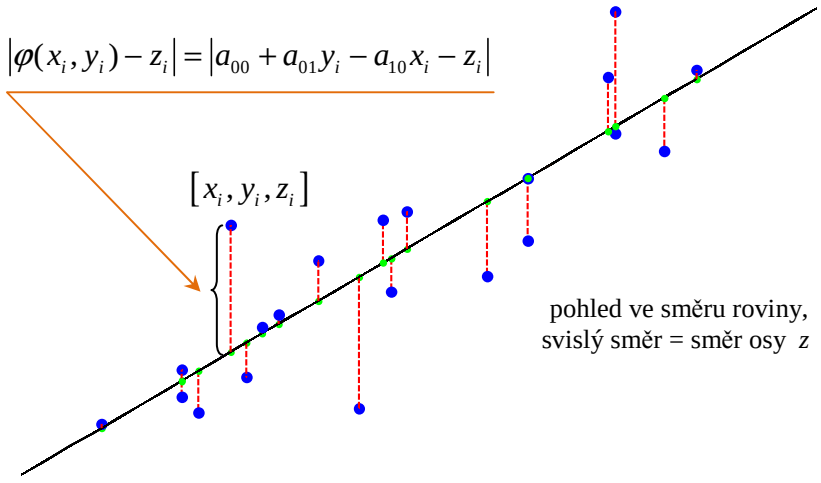
Je velice podstatné k jakému účelu aproximaci funkce potřebujeme. Pokud se jedná o vizualizaci naměřených bodů a zachycení jejich přibližného charakteru, aproximace metodou nejmenších čtverců je plně dostačující. V jiném případě bude potřeba zvolit odlišné kritérium pro hledání aproximující funkce. V příkladech bychom mohli dále rozebírat průběhy výsledných aproximujících funkcí například pomocí rovinných řezů. V dalších experimentech zvolíme jiný typ aproximace. Tyto přístupy uvádíme pro porovnání.

Výstupy jsou získány z výpočetního prostředí MATLAB. V programech řešíme aproximaci dat rovinou a polynomem dvou proměnných libovolného stupně metodou nejmenších čtverců pomocí odvozených vztahů.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu): `mnc_rovina.m` (programy/mnc3d),
`mnc_polynom_3d.m` (programy/mnc3d).

x_i	-20	-17	-16,5	-16,5	-13	-13	-12	-11	-9	-8	
y_i	22	9	4	17	17	20	-9	-4	-10	11,5	
z_i	18	4	8	13	20	11	-6	0	-4	8	
x_i	-7	-6,5	-6	-5	-4	-3	-3	-2	0	3	4
y_i	-6	2,5	16,5	-1,5	11,5	-15	1	-5	-7	3	-10
z_i	-5	5	14	2	3	-9	-8	-3	3	-2	-9

Tabulka 3.4: Experimentální data pro aproximaci rovinou metodou nejmenších čtverců



Obrázek 3.55: Prokládání dat metodou nejmenších čtverců – princip metody

$$B = \begin{pmatrix} 1 & 22 & -20 \\ 1 & 9 & -17 \\ 1 & 4 & -16,5 \\ 1 & 17 & -16,5 \\ 1 & 17 & -13 \\ 1 & 20 & -13 \\ 1 & -9 & -12 \\ 1 & -4 & -11 \\ 1 & -10 & -9 \\ 1 & -11,5 & -8 \\ 1 & -6 & -7 \\ 1 & 2,5 & -6,5 \\ 1 & 16,5 & -6 \\ 1 & -1,5 & -5 \\ 1 & 11,5 & -4 \\ 1 & -15 & -3 \\ 1 & 1 & -3 \\ 1 & -5 & -2 \\ 1 & -7 & 0 \\ 1 & 3 & 3 \\ 1 & -10 & 4 \end{pmatrix} \quad z = \begin{pmatrix} 18 \\ 4 \\ 8 \\ 13 \\ 20 \\ 11 \\ -6 \\ 0 \\ -4 \\ 8 \\ -5 \\ 5 \\ 14 \\ 2 \\ 3 \\ -9 \\ -8 \\ -3 \\ 3 \\ -2 \\ -9 \end{pmatrix}$$

$$B^{\mathrm{T}}B = \begin{pmatrix} 21 & 67,5 & -165,5 \\ 67,5 & 2746,25 & -1361,25 \\ -165,5 & -1361,25 & 2196,75 \end{pmatrix} \qquad B^{\mathrm{T}}z = \begin{pmatrix} 63 \\ 1941 \\ -1222 \end{pmatrix}$$

$$\begin{pmatrix} 21 & 67,5 & -165,5 \\ 67,5 & 2746,25 & -1361,25 \\ -165,5 & -1361,25 & 2196,75 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \end{pmatrix} = \begin{pmatrix} 63 \\ 1941 \\ -1222 \end{pmatrix}$$

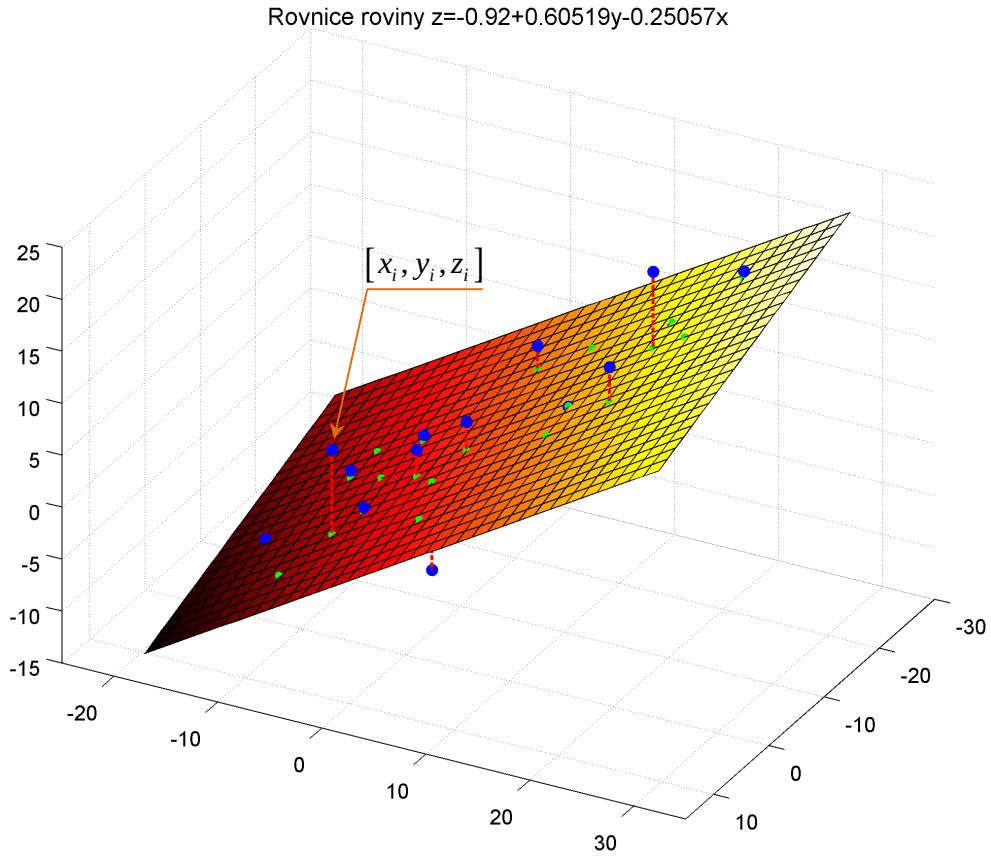
$$B^{\mathrm{T}}Bx = B^{\mathrm{T}}z \quad \text{(řešení soustavy)} \qquad x = \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \end{pmatrix}$$

$$a_{00} = -0,9191 \qquad a_{01} = 0,60519 \qquad a_{10} = -0,25057$$

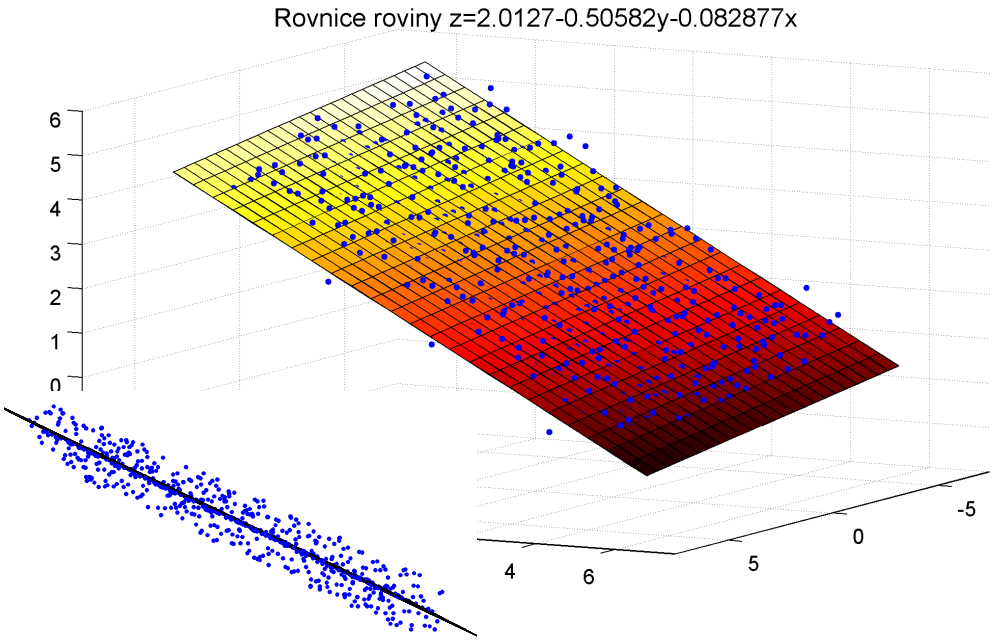
$$H(a_{00}, a_{01}, a_{10}) = 2940,83439$$

$$z = -0,9191 + 0,60519y - 0,25057x$$

rovnice aproximující roviny

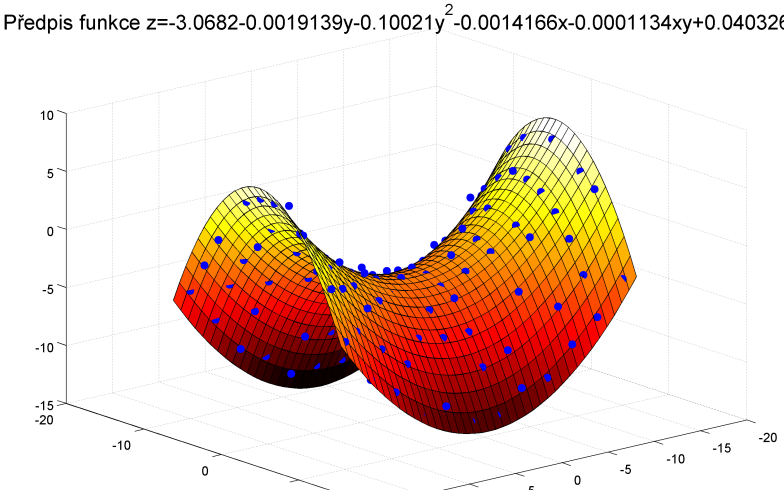


Obrázek 3.56: Prokládání dat metodou nejmenších čtverců – situace v prostoru

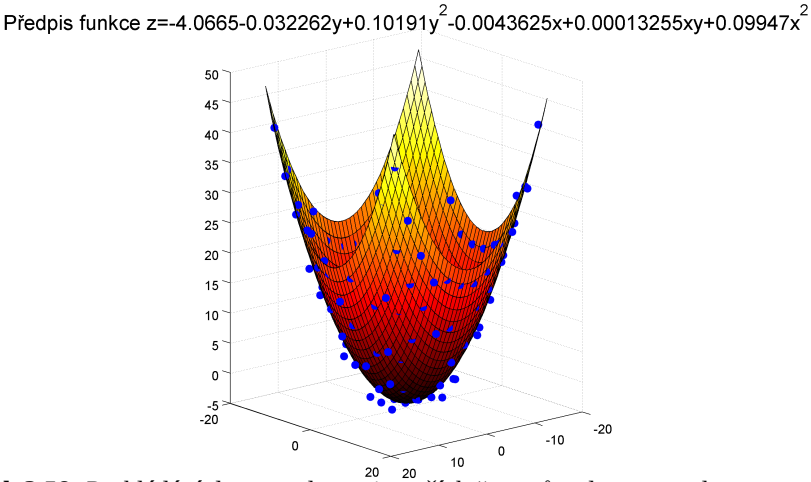


Obrázek 3.57: Prokládání dat metodou nejmenších čtverců – praktická úloha

a)



b)



Obrázek 3.58: Prokládání dat metodou nejmenších čtverců polynomem dvou proměnných druhého stupně

3.5 Ortogonální prokládání dat rovinou

V dalších experimentech a zpracování vstupních bodových množin využijeme metodu prokládání dat rovinou založenou na přirozeném geometrickém kritériu. Opět minimalizujeme součty druhých mocnin vzdáleností, pracujeme tedy na základě metody nejmenších čtverců, ovšem tentokrát tyto odchylky měříme na kolmicích k hledané aproximující rovině. Obdobně jako v případě prokládání dat přímkou, označíme tuto metodu jako ortogonální prokládání dat rovinou (*orthogonal plane fitting*).

Zaveďme tuto metodu precizně. Odvození lze rozšířit do libovolné dimenze, s ohledem na naše aplikace se omezíme pouze na dimenzi tři. Mějme tedy dánu množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3, značme tuto množinu tradičně $\mathcal{X} = \{X_i\}_{i=0}^{n}$. Souřadnice bodů X_i budeme někdy označovat též $\{[X_i^x, X_i^y, X_i^z]\}_{i=0}^{n}$.

Hledáme nyní rovinu, která bude dobře aproximovat zadané body, přičemž volíme následující geometrické kritérium. Minimalizujeme součet druhých mocnin vzdáleností bodů $\{X_i\}_{i=0}^{n}$ od hledané roviny. Hledanou rovinu ρ vyjádříme obecnou rovnicí

$$n_\rho \cdot (X - A) = 0, \tag{3.80}$$

kde A je bod roviny a n_ρ je její jednotkový normálový vektor, tj. $\|n_\rho\| = 1$. Pro libovolný bod X_i platí

$$X_i - A = d_i n_\rho + p_i n_\rho^\perp, \tag{3.81}$$

kde $d_i = n_\rho \cdot (X_i - A) = n_\rho^{\mathrm{T}} (X_i - A)$ je skalární součin vektorů n_ρ a $X_i - A$, p_i je reálný koeficient a $n_\rho^\perp$ je libovolný jednotkový vektor kolmý na vektor n_ρ. Pokud označíme vektor $w_i = X_i - A$, plyne z rovnosti $n_\rho \cdot w_i = \|n_\rho\| \|w_i\| \cos\varphi$, kde $\varphi \in \langle 0, \pi \rangle$ je úhel, který vektory n_ρ a w_i svírají, že $d_i = n_\rho \cdot w_i = \|w_i\| \cos\varphi$, neboť $\|n_\rho\| = 1$. Vektor $X_i - X_i'$, kde bod X_i' je ortogonální průmět bodu X_i na rovinu ρ, lze tedy zapsat

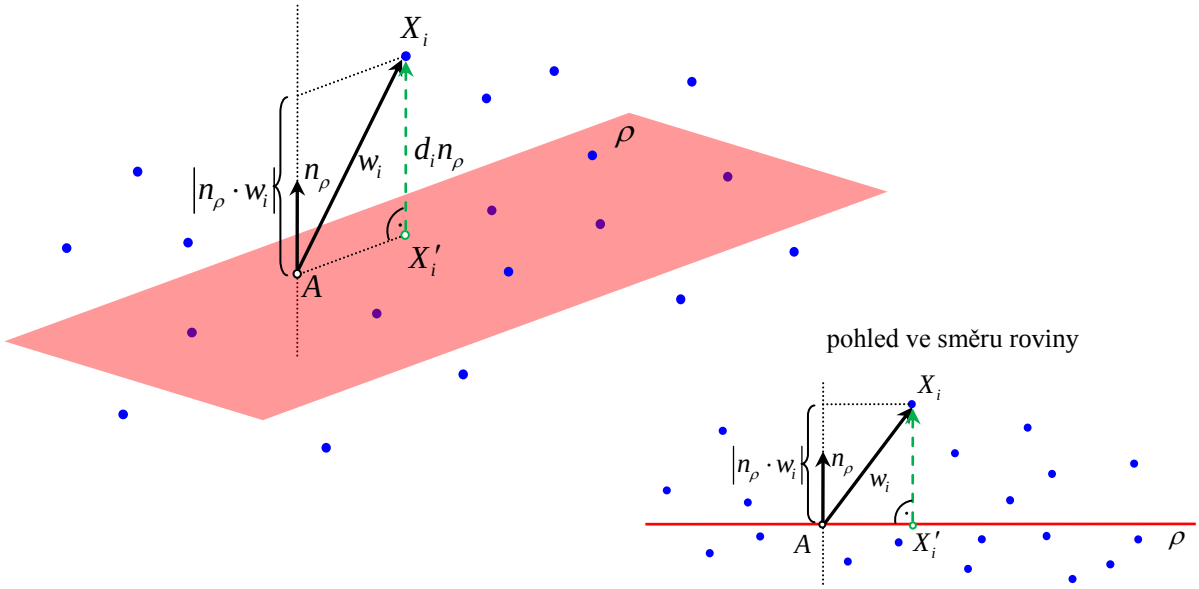
$$X_i - X_i' = d_i n_\rho. \tag{3.82}$$

Velikost vektoru (3.82) je ortogonální vzdálenost bodu X_i od roviny ρ.

Rovinu ρ opět hledáme na základě metody nejmenších čtverců, nyní ale funkce, označme ji σ^2, jejíž minimum hledáme, definujeme jako součet druhých mocnin vzdáleností daných bodů $\{X_i\}_{i=0}^{n}$ od roviny ρ, tj.

$$\sigma^2 = \sum_{i=0}^{n} \|d_i n_\rho\|^2 = \sum_{i=0}^{n} d_i^2 = \sum_{i=0}^{n} (n_\rho \cdot w_i)^2. \tag{3.83}$$

Geometrický význam popsaných vztahů je objasněn na obrázku 3.59.



Obrázek 3.59: Geometrický význam použitých vektorů a vzdáleností při ortogonálním prokládání dat rovinou

Nyní hledáme minimum funkce σ^2 z (3.83) závislé na parametrech A a n_ρ (bod a normálový vektor roviny ρ). Začněme s výpočtem souřadnic bodu A. Stejně jako u ortogonálního prokládání dat rovinou, využijeme standardního postupu hledání minima funkce, tj. spočítáme parciální derivaci funkce σ^2 podle bodu A. Pro zjednodušení výpočtu vyjádříme funkci σ^2 v alternativním tvaru

$$\sigma^2 = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(n_\rho n_\rho^{\mathrm{T}} \right) w_i \right]. \tag{3.84}$$

Výraz (3.83) upravme, abychom ukázali, že (3.84) je skutečně ekvivalentní zápis stejné funkce σ^2, tj.

$$\begin{aligned} \sigma^2 = \sum_{i=0}^{n} \left(n_\rho \cdot w_i \right)^2 = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} n_\rho \right)^2 = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} n_\rho \right) \left(w_i^{\mathrm{T}} n_\rho \right) = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} n_\rho \right) \left(n_\rho^{\mathrm{T}} w_i \right) = \\ = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(n_\rho n_\rho^{\mathrm{T}} \right) w_i \right]. \end{aligned} \tag{3.85}$$

Pro lepší názornost tvar funkce σ^2 z (3.84) rozepišme po souřadnicích, přičemž horní index u vektoru značí *x*-ovou, *y*-ovou nebo *z*-ovou souřadnici, tj. $w_i = (w_i^x, w_i^y, w_i^z)^{\mathrm{T}}$ a $n_\rho = (n_\rho^x, n_\rho^y, n_\rho^z)^{\mathrm{T}}$. Uvádíme také, s jakými typy matic pracujeme.

$$\sigma^2 = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} \left(n_\rho n_\rho^{\mathrm{T}} \right) w_i \right] =$$

$$= \sum_{i=0}^{n} \left[\begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \left(\begin{pmatrix} n_\rho^x \\ n_\rho^y \\ n_\rho^z \end{pmatrix} \begin{pmatrix} n_\rho^x & n_\rho^y & n_\rho^z \end{pmatrix} \right) \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right], \tag{3.86}$$

kde jeden člen sumy představuje součin matic typů $\left[(1\times3)(3\times1)(1\times3)(3\times1)\right]$.

Čtvercovou matici $n_\rho n_\rho^{\mathrm{T}}$ z (3.84) a (3.86) označme jako N a prvky této matice n_{kl}, tedy $N = (n_{kl})$, rozepišme následovně

$$N = \begin{pmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{pmatrix} = \begin{pmatrix} (n_\rho^x)^2 & n_\rho^x n_\rho^y & n_\rho^x n_\rho^z \\ n_\rho^x n_\rho^y & (n_\rho^y)^2 & n_\rho^y n_\rho^z \\ n_\rho^x n_\rho^z & n_\rho^y n_\rho^z & (n_\rho^z)^2 \end{pmatrix}. \tag{3.87}$$

Je zřejmé, že matice N je symetrická, neboť pro každé $k,l = 1,2,3$ platí $n_{kl} = n_{lk}$.

Nyní můžeme přepsat tvar funkce σ^2 z (3.86) a upravit jednotlivé matice uvnitř sumy postupným roznásobením

$$\sigma^2 = \sum_{i=0}^{n} \left[w_i^{\mathrm{T}} N w_i \right] = \sum_{i=0}^{n} \left[\begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \begin{pmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{pmatrix} \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right] =$$

$$\sum_{i=0}^{n} \left[\left(w_i^x n_{11} + w_i^y n_{21} + w_i^z n_{31}, w_i^x n_{12} + w_i^y n_{22} + w_i^z n_{32}, w_i^x n_{13} + w_i^y n_{23} + w_i^z n_{33} \right) \begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \right] =$$

$$\begin{aligned} &= \sum_{i=0}^{n} \Big[(w_i^x)^2 n_{11} + w_i^x w_i^y n_{21} + w_i^x w_i^z n_{31} + \\ &+ w_i^x w_i^y n_{12} + (w_i^y)^2 n_{22} + w_i^y w_i^z n_{32} + \\ &+ w_i^x w_i^z n_{13} + w_i^y w_i^z n_{23} + (w_i^z)^2 n_{33} \Big]. \end{aligned} \tag{3.88}$$

Hodnotu funkce σ^2 jsme obdrželi vyjádřenou v závislosti na souřadnicích jednotlivých vstupních parametrů.

Vypočítejme nyní parciální derivace funkce σ^2 podle bodu A a využijme k tomu tvar funkce σ^2 z (3.84). Z předchozího již víme, že parciální derivaci funkce σ^2 podle bodu A definujeme jako vektor parciálních derivací funkce σ^2 podle jednotlivých souřadnic, tj.

$$\frac{\partial \sigma^2}{\partial A} = \left(\frac{\partial \sigma^2}{\partial A^x}, \frac{\partial \sigma^2}{\partial A^y}, \frac{\partial \sigma^2}{\partial A^z} \right)^{\mathrm{T}}. \tag{3.89}$$

Vektor w_i je definován jako $X_i - A$. Rozepíšeme-li tento vektor po souřadnicích, dostáváme $w_i = (X_i^x - A^x, X_i^y - A^y, X_i^z - A^z)^{\mathrm{T}}$. Funkce σ^2 je tedy složenou funkcí, jak je vidět z (3.88), a pro složky parciální derivace σ^2 podle bodu A platí

$$\frac{\partial \sigma^2}{\partial A^x} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^x} \frac{\partial w_i^x}{\partial A^x}, \; \frac{\partial \sigma^2}{\partial A^y} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^y} \frac{\partial w_i^y}{\partial A^y}, \; \frac{\partial \sigma^2}{\partial A^z} = \sum_{i=0}^{n} \frac{\partial \sigma^2}{\partial w_i^z} \frac{\partial w_i^z}{\partial A^z}. \tag{3.90}$$

Nyní spočítáme parciální derivaci funkce σ^2 podle bodu A a vše rozepíšeme pro jednotlivé souřadnice. K výpočtům parciální derivace použijeme rozepsaný tvar funkce σ^2 po souřadnicích v (3.88). Výsledný tvar parciální derivace zjednodušíme na základě toho, že matice N je symetrická.

Pro *x*-ovou, *y*-ovou a *z*-ovou souřadnici parciální derivace funkce σ^2 podle bodu A dostáváme následující

$$\frac{\partial \sigma^2}{\partial w_i^x} = \sum_{i=0}^{n} \left(2w_i^x n_{11} + w_i^y n_{21} + w_i^z n_{31} + w_i^y n_{12} + w_i^z n_{13} \right), \; \frac{\partial w_i^x}{\partial A^x} = -1$$

$$\Downarrow \tag{3.91}$$

$$\frac{\partial \sigma^2}{\partial A^x} = -\sum_{i=0}^{n} \left(2w_i^x n_{11} + w_i^y n_{21} + w_i^z n_{31} + w_i^y n_{12} + w_i^z n_{13} \right) = -2 \sum_{i=0}^{n} \left(w_i^x n_{11} + w_i^y n_{12} + w_i^z n_{13} \right)$$

a

$$\frac{\partial \sigma^2}{\partial w_i^y} = \sum_{i=0}^{n} \left(w_i^x n_{21} + w_i^x n_{12} + 2w_i^y n_{22} + w_i^z n_{32} + w_i^z n_{23} \right), \; \frac{\partial w_i^y}{\partial A^y} = -1$$

$$\Downarrow \tag{3.92}$$

$$\frac{\partial \sigma^2}{\partial A^y} = -\sum_{i=0}^{n} \left(w_i^x n_{21} + w_i^x n_{12} + 2w_i^y n_{22} + w_i^z n_{32} + w_i^z n_{23} \right) = -2 \sum_{i=0}^{n} \left(w_i^x n_{21} + w_i^y n_{22} + w_i^z n_{23} \right)$$

a

$$\frac{\partial \sigma^2}{\partial w_i^z} = \sum_{i=0}^{n} \left(w_i^x n_{31} + w_i^y n_{32} + w_i^x n_{13} + w_i^y n_{23} + 2w_i^z n_{33} \right), \; \frac{\partial w_i^z}{\partial A^z} = -1$$

$$\Downarrow \tag{3.93}$$

$$\frac{\partial \sigma^2}{\partial A^z} = -\sum_{i=0}^{n} \left(w_i^x n_{31} + w_i^y n_{32} + w_i^x n_{13} + w_i^y n_{23} + 2w_i^z n_{33} \right) = -2 \sum_{i=0}^{n} \left(w_i^x n_{31} + w_i^y n_{32} + w_i^z n_{33} \right).$$

Parciální derivace σ^2 podle bodu A můžeme pro jednotlivé složky z (3.91), (3.92) a (3.93) přepsat jako

$$\frac{\partial \sigma^2}{\partial A^x} = -2\left(n_{11}, n_{12}, n_{13}\right)\sum_{i=0}^{n}\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix},$$

$$\frac{\partial \sigma^2}{\partial A^y} = -2\left(n_{21}, n_{22}, n_{23}\right)\sum_{i=0}^{n}\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix}, \tag{3.94}$$

$$\frac{\partial \sigma^2}{\partial A^z} = -2\left(n_{31}, n_{32}, n_{33}\right)\sum_{i=0}^{n}\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix},$$

tj. skalární součin prvního, druhého nebo třetího řádku matice N a vektoru, který vznikne jako součet všech vektorů w_i pro $i = 0,1,...,n$, vynásobený konstantou -2.

Symbolicky můžeme zapsat parciální derivaci funkce σ^2 podle bodu A následovně

$$\frac{\partial \sigma^2}{\partial A} = -2\left(n_\rho n_\rho^{\mathrm{T}}\right)\sum_{i=0}^{n} w_i\,, \tag{3.95}$$

kde $n_\rho n_\rho^{\mathrm{T}} = N$.

Parciální derivace funkce σ^2 podle bodu A z (3.95) je rovna nulovému vektoru právě tehdy, když

$$\sum_{i=0}^{n} w_i = o\,, \tag{3.96}$$

což lze rozepsat po složkách jako

$$\sum_{i=0}^{n} w_i^x = 0\,,\ \sum_{i=0}^{n} w_i^y = 0\,,\ \sum_{i=0}^{n} w_i^z = 0\,. \tag{3.97}$$

Dosadíme-li do výrazu (3.96) za $w_i = X_i - A$, dostáváme vzorec pro výpočet bodu A

$$A = \frac{1}{n+1}\sum_{i=0}^{n} X_i\,. \tag{3.98}$$

Bod A tedy spočítáme jako aritmetický průměr bodů $\left\{X_i\right\}_{i=0}^{n}$. Výsledek (3.98) vyjádřeme opět v souřadnicích

$$A^x = \frac{1}{n+1}\sum_{i=0}^{n} X_i^{\,x},\ A^y = \frac{1}{n+1}\sum_{i=0}^{n} X_i^{\,y},\ A^z = \frac{1}{n+1}\sum_{i=0}^{n} X_i^{\,z}\,. \tag{3.99}$$

Nyní máme určený bod A, kterým prochází hledaná rovina ρ. Zbývá tedy dopočítat jednotkový normálový vektor n_ρ této roviny. Funkci σ^2 si tentokrát přepišme jako

$$\sigma^2 = n_\rho^{\mathrm{T}} \left(\sum_{i=0}^{n} \left[w_i w_i^{\mathrm{T}} \right] \right) n_\rho = n_\rho^{\mathrm{T}} M n_\rho , \tag{3.100}$$

kde čtvercová matice M značí sumu v závorce.

Výraz (3.100) je skutečně ekvivalentním zápisem funkce σ^2 definované v (3.83), neboť platí

$$\begin{aligned} \sigma^2 &= \sum_{i=0}^{n} \left(n_\rho \cdot w_i \right)^2 = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} n_\rho \right)^2 = \sum_{i=0}^{n} \left(w_i^{\mathrm{T}} n_\rho \right) \left(w_i^{\mathrm{T}} n_\rho \right) = \sum_{i=0}^{n} \left(n_\rho^{\mathrm{T}} w_i \right) \left(w_i^{\mathrm{T}} n_\rho \right) = \\ &= \sum_{i=0}^{n} \left[n_\rho^{\mathrm{T}} \left(w_i w_i^{\mathrm{T}} \right) n_\rho \right] = n_\rho^{\mathrm{T}} \left(\sum_{i=0}^{n} \left[w_i w_i^{\mathrm{T}} \right] \right) n_\rho . \end{aligned} \tag{3.101}$$

Tvar funkce σ^2 z (3.101) si pro lepší názornost rozepišme opět po souřadnicích, tj.

$$\begin{aligned} \sigma^2 &= n_\rho^{\mathrm{T}} \left(\sum_{i=0}^{n} \left[w_i w_i^{\mathrm{T}} \right] \right) n_\rho = \\ &= \begin{pmatrix} n_\rho^x & n_\rho^y & n_\rho^z \end{pmatrix} \left(\sum_{i=0}^{n} \left[\begin{pmatrix} w_i^x \\ w_i^y \\ w_i^z \end{pmatrix} \begin{pmatrix} w_i^x & w_i^y & w_i^z \end{pmatrix} \right] \right) \begin{pmatrix} n_\rho^x \\ n_\rho^y \\ n_\rho^z \end{pmatrix} = n_\rho^{\mathrm{T}} M n_\rho , \end{aligned} \tag{3.102}$$

kde M je čtvercová matice tvaru

$$M = \sum_{i=0}^{n} \begin{pmatrix} (w_i^x)^2 & w_i^x w_i^y & w_i^x w_i^z \\ w_i^x w_i^y & (w_i^y)^2 & w_i^y w_i^z \\ w_i^x w_i^z & w_i^y w_i^z & (w_i^z)^2 \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n} (w_i^x)^2 & \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} w_i^x w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} (w_i^y)^2 & \sum_{i=0}^{n} w_i^y w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^z & \sum_{i=0}^{n} w_i^y w_i^z & \sum_{i=0}^{n} (w_i^z)^2 \end{pmatrix}. \tag{3.103}$$

Označíme-li prvky této matice m_{kl}, tedy $M = (m_{kl})$, je zřejmé, že matice M je symetrická, neboť pro každé $k,l = 1,2,3$ platí $m_{kl} = m_{lk}$.

Obdobně jako u ortogonálního prokládání dat přímkou vidíme, že zobrazení $\sigma^2 = n_\rho^{\mathrm{T}} M n_\rho$ z (3.100) je kvadratická forma. Jednotkový normálový vektor n_ρ hledané roviny ρ tuto formu minimalizuje. Podle věty 3.3 víme, že nejmenší a největší vlastní číslo matice M se rovná minimu a maximu kvadratické formy $\sigma^2 = n_\rho^{\mathrm{T}} M n_\rho$. Pro daný bod A tedy určíme jednotkový normálový vektor n_ρ jako vlastní vektor příslušný nejmenšímu vlastnímu číslu matice M. Předpoklady věty 3.3 jsou splněny, neboť matice M je symetrická a vek-

tor n_ρ je jednotkový. K výpočtu vlastních čísel matice M a jim příslušných vlastních vektorů, užijeme opět standardních postupů, které si ukážeme na konkrétním příkladě.

Nyní jsme tedy nalezli určující bod A a jednotkový normálový vektor n_ρ hledané roviny ρ. Rovinu ρ, jak jsme uvedli na začátku tohoto oddílu, získáme popsanou obecnou rovnicí

$$n_\rho \cdot (X - A) = 0, \tag{3.104}$$

kde A je bod roviny a n_ρ je její jednotkový normálový vektor, tj. $\|n_\rho\| = 1$.

Výsledky, ke kterým jsme dospěli při řešení ortogonálního prokládání dat rovinou, lze nalézt například v (Ahn, 2004) nebo na webových stránkách (Geometric Tools, 2012). Opět jsou zde předloženy pouze závěry. V české literatuře důkladné odvození této metody s názornou geometrickou interpretací rovněž chybí. Tímto jsme doplnili dostupnou literaturu o nutnou teorii, důkazy a výpočetní postupy, které jsou potřebné k pochopení dané problematiky.

Ukažme nyní prokládání dat rovinou na konkrétním příkladě.

Příklad 3.4: *ORTOGONÁLNÍ PROKLÁDÁNÍ DAT ROVINOU*. Odvozený typ aproximace rovinou nyní demonstrujme na příkladě. Mějme dánu množinu $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3. Princip metody předveďme opět na menším počtu vstupních bodů. Experimentální data pro ortogonální prokládání dat rovinou jsou uvedena v tabulce 3.5, přičemž se jedná o stejná data, která byla použita také v ukázce aproximace rovinou metodou nejmenších čtverců. Stejná data jsme volili proto, abychom mohli výsledné aproximace porovnat.

Dále je podrobně rozepsán postup výpočtu vedoucí k nalezení aproximující roviny $n_\rho \cdot (X - A) = 0$. Odvozeny jsou souřadnice bodu A a jednotkového normálového vektoru n_ρ roviny ρ. Naznačen je rovněž postup výpočtu vlastních čísel a příslušných vlastních vektorů matice M. (Ve všech případech jsou číselné hodnoty při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Na obrázcích 3.60 a 3.61 je potom znázorněna výsledná rovina, pro kterou vyžadujeme, aby součet čtverců vyznačených vzdáleností zadaných bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ od roviny byl nejmenší možný. Na obrázku 3.60 je pohled zvolen tak, že se výsledná rovina promítá do přímky.

Výstupy jsou získány z výpočetního prostředí MATLAB. V programu řešíme ortogonální prokládání dat prostoru rovinou pomocí odvozených vztahů, přičemž k výpočtu vlastních vektorů a příslušných vlastních čísel jsou použity funkce MATLABu. Doporučujeme čtenáři prohlédnout si výstup přímo ve výpočetním prostředí MATLAB, kde lze s výsledným obrázkem pohybovat. Prostorová situace je tak názornější. K řešení problému ortogonálního prokládání dat rovinou lze také přímo využít knihovní funkci MATLABu `mldivide`.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu): `rovina_orto.m` (programy/orto_rovina).

---

x_i	-20	-17	-16,5	-16,5	-13	-13	-12	-11	-9	-8	
y_i	22	9	4	17	17	20	-9	-4	-10	11,5	
z_i	18	4	8	13	20	11	-6	0	-4	8	
x_i	-7	-6,5	-6	-5	-4	-3	-3	-2	0	3	4
y_i	-6	2,5	16,5	-1,5	11,5	-15	1	-5	-7	3	-10
z_i	-5	5	14	2	3	-9	-8	-3	3	-2	-9

Tabulka 3.5: Experimentální data v prostoru pro ortogonální prokládání dat rovinou

Obecná rovnice roviny $\rho \quad n_\rho \cdot (X - A) = 0$

Výpočet souřadnic bodu A

$$A = \frac{1}{n+1}\sum_{i=0}^{n} X_i \qquad A = [A^x, A^y, A^z]$$

$$A^x = \frac{1}{n+1}\sum_{i=0}^{n} X_i^x = \frac{1}{21}\cdot(-165,5) = -7,88095$$

$$A^y = \frac{1}{n+1}\sum_{i=0}^{n} X_i^y = \frac{1}{21}\cdot 67,5 = 3,21429$$

$$A^z = \frac{1}{n+1}\sum_{i=0}^{n} X_i^z = \frac{1}{21}\cdot 63 = 3$$

$$A = [-7,88095; 3,21429; 3]$$

$$M = \begin{pmatrix} \sum_{i=0}^{n}(w_i^x)^2 & \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n} w_i^x w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^y & \sum_{i=0}^{n}(w_i^y)^2 & \sum_{i=0}^{n} w_i^y w_i^z \\ \sum_{i=0}^{n} w_i^x w_i^z & \sum_{i=0}^{n} w_i^y w_i^z & \sum_{i=0}^{n}(w_i^z)^2 \end{pmatrix}$$

$$M = \begin{pmatrix} 892,45238 & -829,28571 & -725,5 \\ -829,28571 & 2529,28571 & 1738,5 \\ -725,5 & 1738,5 & 1528 \end{pmatrix}$$

Výpočet vlastních čísel a příslušných vlastních vektorů matice M

$$\lambda E - M = \begin{pmatrix} \lambda - 892,45238 & 829,28571 & 725,5 \\ 829,28571 & \lambda - 2529,28571 & -1738,5 \\ 725,5 & -1738,5 & \lambda - 1528 \end{pmatrix}$$

$$\det(\lambda E - M) = (\lambda - 892,45238)(\lambda - 2529,28571)(\lambda - 1528) - 2091925873,92857 -$$
$$-526350,25(\lambda - 2529,28571) -$$
$$-687714,79592(\lambda - 1528) -$$
$$-3022382,25(\lambda - 1528) =$$
$$= \lambda^3 - 4949,7381\lambda^2 + 3249235,57143\lambda - 461579326,92262$$

charakteristický polynom

kořeny charakteristického polynomu

$$\lambda^3 - 4949,7381\lambda^2 + 3249235,57143\lambda - 461579326,92262 = 0$$

Vlastní čísla

$$\lambda_1 = 201,24409 \qquad \lambda_2 = 545,74499 \qquad \lambda_3 = 4202,74902$$

jednotkový vlastní vektor příslušný nejmenšímu vlastnímu číslu je netriviálním řešením homogenní soustavy s maticí $\lambda_1 E - M$

$$\begin{pmatrix} -691,20829 & 829,28571 & 725,5 \\ 829,28571 & -2328,04162 & -1738,5 \\ 725,5 & -1738,5 & -1326,75591 \end{pmatrix} \begin{pmatrix} n_\rho^x \\ n_\rho^y \\ n_\rho^z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$(\lambda_1 E - M) n_\rho = o$$

řešení soustavy

součet druhých mocnin vzdáleností daných bodů od roviny ρ

$$\sigma^2 = n_\rho^{\mathrm{T}} M n_\rho = 201,24409$$

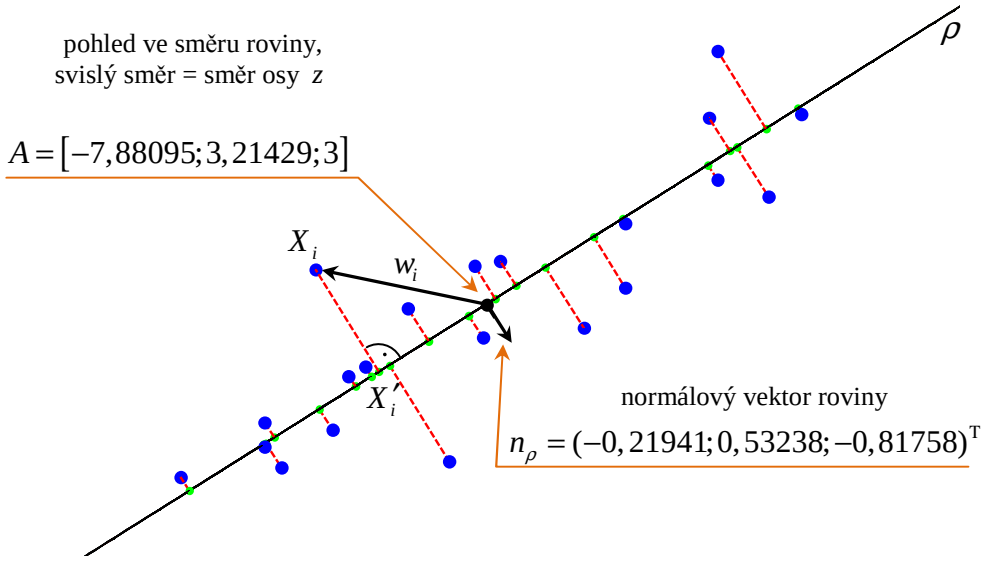
$$n_\rho^x = -0,21941 \qquad n_\rho^y = 0,53238 \qquad n_\rho^z = -0,81758$$

jednotkový normálový vektor roviny ρ

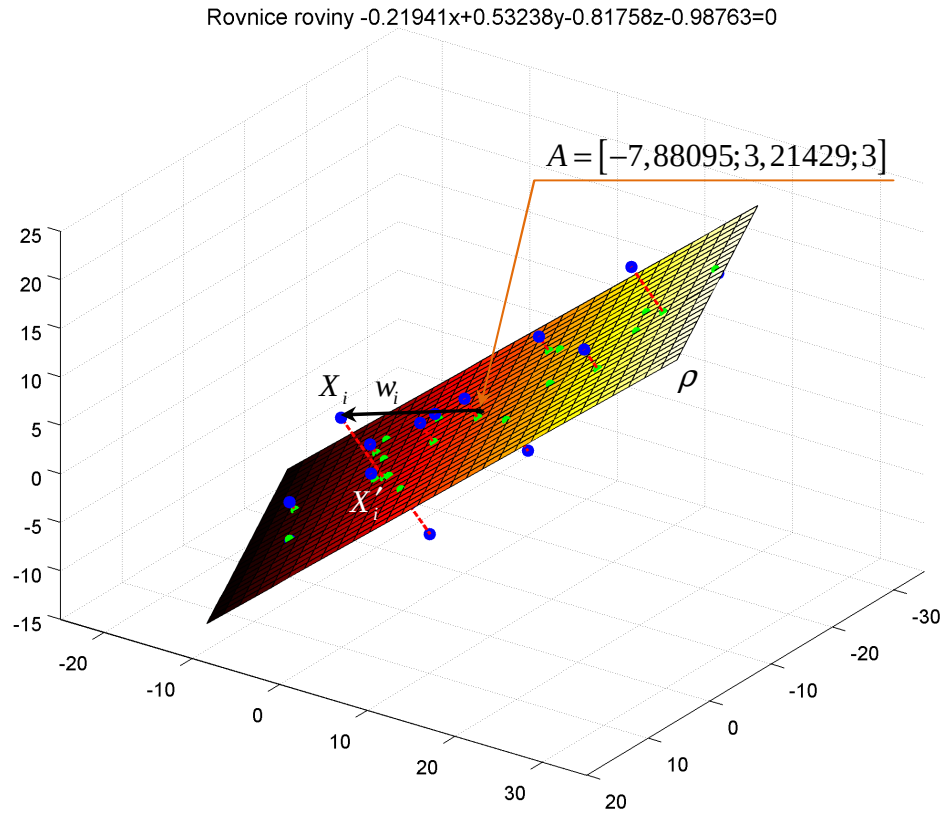
$$n_\rho = (-0,21941; 0,53238; -0,81758)^{\mathrm{T}}$$

obecná rovnice roviny ρ

$$-0,21941x + 0,53238y - 0,81758z - 0,98763 = 0$$



Obrázek 3.60: Ortogonální prokládání dat rovinou – princip metody



Obrázek 3.61: Ortogonální prokládání dat rovinou – situace v prostoru

Ortogonální prokládání dat v prostoru rovinou opět úzce souvisí se známou statistickou metodou analýzou hlavních komponent (*Principal Component Analysis*). V následujícím oddíle si tento postup podrobněji popíšeme a porovnáme jej s odvozenými metodami ortogonálního prokládání dat přímkou v rovině a v prostoru a ortogonálního prokládání dat v prostoru rovinou.

3.6 Analýza hlavních komponent (Principal Component Analysis)

Popišme v tomto oddíle známou vícerozměrnou statistickou metodu hledání hlavních komponent (*Principal Component Analysis, PCA*), která představuje jednu z nejpoužívanějších metod vícerozměrné analýzy, a srovnejme ji s ortogonálním prokládáním dat. Dále budeme o této metodě hovořit jako o PCA metodě. Pojem vícerozměrné náhodné veličiny, úprava vícerozměrných dat a statistická analýza vícerozměrných dat je běžně uváděna v (Anděl, 1985; Jackson, 2003; Jolliffe, 2002; Meloun a Militký, 2004).

Hlavní myšlenkou analýzy hlavních komponent je zjednodušení popisu souboru dat obsahujících velké množství *korelovaných* (vzájemně lineárně závislých) proměnných s tím, že je splněn požadavek co nejmenší ztráty informace o daných datech. Toho se docílí trans-

formací původních proměnných na nové, nekorelované proměnné, které nazýváme *hlavní komponenty* (*principal components*). Jinými slovy, jedná se o lineární transformaci původního souřadnicového systému do souřadnicového systému hlavních komponent, které jsou vzájemně ortogonální (nekorelované) a vybrané tak, aby postihovaly maximální množství informací vyjádřené variabilitou mezi objekty. Nový ortogonální souřadnicový systém je natočen do směrů, které postihují *maximální variabilitu minimalizující vzdálenosti objektů od hlavních komponent*. Tuto základní myšlenku metody PCA uvádějí například (Meloun a Militký, 2004). Každá hlavní komponenta je lineární kombinací původních proměnných a charakterizována je tedy svým rozptylem. Hlavní komponenty jsou seřazeny podle důležitosti, tj. od komponenty s největším rozptylem po komponentu s rozptylem nejmenším. Většina informace o variabilitě původních dat je přitom obsažena v první hlavní komponentě, nejméně informace potom v hlavní komponentě poslední. První hlavní komponenta popisuje největší část proměnlivosti tedy rozptylu původních dat, druhá hlavní komponenta popisuje největší část rozptylu neobsaženého v první komponentě a obdobně dále. Každý objekt má tedy přiřazeny nové souřadnice, projekce do hlavních komponent. Výsledky metody PCA se často prezentují v grafické formě a slouží ke snížení rozměrnosti problému náhradou původních m znaků menším počtem hlavních komponent, které jsou tvořeny lineární kombinací původních znaků. Předpokládá se totiž, že nevyužité hlavní komponenty obsahují malé množství informace, protože jejich rozptyl je příliš malý. První dvě nebo první tři hlavní komponenty slouží jako techniky zobrazení vícerozměrných dat v projekci do roviny, respektive do prostoru. Obdobnou stručnou charakteristiku metody PCA předkládají Meloun a Militký v (2004).

Popišme statistickou analýzu vícerozměrných dat podrobněji. Vedle jednorozměrných analytických informací, obsažených v náhodném skaláru, se v technické praxi vyskytují i vícerozměrné analytické informace, obsažené v náhodném vektoru ξ s m složkami $\xi_1, \xi_2, ..., \xi_m$, přičemž se předpokládá, že složky tohoto vektoru jsou náhodnými veličinami. Na základě nějaké analýzy potom máme k dispozici tzv. *zdrojovou matici* dat $X \in \mathbb{R}_{n\times m}$, která představuje náhodný výběr velikosti n. Výběr je tvořen *n*-ticí vektorů $x_i = (x_{i1}, x_{i2}, ..., x_{im})^{\mathrm{T}}$, $i = 1, 2, ..., n$, které budeme chápat (v souvislosti s našimi aplikacemi) jako souřadnice n bodů v *m*-rozměrném prostoru. Zdrojová matice vypadá tedy následovně (složky jednoho vektoru jsou zapsány v řádcích)

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}. \tag{3.105}$$

Budeme vždy předpokládat, že počet bodů n, tedy velikost výběru, je větší než počet složek m náhodného vektoru. V našem případě, kdy pracujeme v eukleidovské rovině E_2, případně v eukleidovském prostoru E_3, máme výběr, který je tvořen *n*-ticí vektorů $x_i = (x_i^x, x_i^y)^{\mathrm{T}}$, případně *n*-ticí vektorů $x_i = (x_i^x, x_i^y, x_i^z)^{\mathrm{T}}$, kde $i = 1, 2, ..., n$.

Vektor ξ je náhodný vektor charakterizovaný svou tzv. *sdruženou distribuční funkcí* (Meloun a Militký, 2004). Máme n měření náhodného vektoru ξ, tedy zdrojovou matici X. Abychom mohli přejít k zavedení metody PCA, je nutné nadefinovat některé pojmy z popisné statistiky vícerozměrných dat. Čerpáme přitom ze zmíněných monografií (Jackson, 2003; Jolliffe, 2002; Meloun a Militký, 2004). K charakteristice jednotlivých složek náhodného vektoru ξ využijeme obdobně jako u jednorozměrných náhodných veličin momenty. K popisu polohy *j*-té složky ξ_j vektoru ξ se používá *střední hodnota* $E(\xi_j)=\mu_j$ a k popisu rozptýlení *rozptyl* $D(\xi_j)=\sigma_j^2$. Lze tedy psát $\mu = E(\xi) = \left(E(\xi_1), E(\xi_2), ..., E(\xi_m)\right)^{\mathrm{T}}$ a $D(\xi) = \left(D(\xi_1), D(\xi_2), ..., D(\xi_m)\right)^{\mathrm{T}}$. Navíc definujme *míru intenzity* vztahu mezi složkami ξ_i a ξ_j, $i \neq j$ vektoru ξ. K tomu nám vhodně poslouží *kovariance* $\mathrm{cov}(\xi_i, \xi_j)$, pro kterou platí $\mathrm{cov}(\xi_i, \xi_j) = E(\xi_i \xi_j) - E(\xi_i)E(\xi_j)$. Vlastnosti kovariance jsou zřejmé, lze je také nalézt ve zmiňované literatuře. Při výpočtech je místo kovariance někdy výhodnější používat *koeficient korelace* $\rho(\xi_i, \xi_j)$ definovaný vztahem $\rho(\xi_i, \xi_j) = \mathrm{cov}(\xi_i, \xi_j) / \sigma_i \sigma_j$, kde σ_i a σ_j jsou *směrodatné odchylky* složek ξ_i a ξ_j, tedy $\sigma_i = \sqrt{D(\xi_i)}$. Koeficient korelace $\rho(\xi_i, \xi_j)$ budeme dále značit ρ_{ij}. Statistické chování náhodného vektoru ξ se standardně charakterizuje pomocí vektoru středních hodnot $\mu = \left(E(\xi_1), E(\xi_2), ..., E(\xi_m)\right)^{\mathrm{T}}$ a pomocí symetrické *kovariančí matice* řádu m, která má následující tvar

$$C = \begin{pmatrix} D(\xi_1) & \mathrm{cov}(\xi_1, \xi_2) & \cdots & \mathrm{cov}(\xi_1, \xi_m) \\ \mathrm{cov}(\xi_1, \xi_2) & D(\xi_2) & \cdots & \mathrm{cov}(\xi_2, \xi_m) \\ \vdots & \vdots & & \vdots \\ \mathrm{cov}(\xi_1, \xi_m) & \mathrm{cov}(\xi_2, \xi_m) & \cdots & D(\xi_m) \end{pmatrix}. \tag{3.106}$$

Místo kovarianční matice se někdy používá také její normovaná verze a to korelační matice tvaru

$$R = \begin{pmatrix} 1 & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{12} & 1 & \cdots & \rho_{2m} \\ \vdots & \vdots & & \vdots \\ \rho_{1m} & \rho_{2m} & \cdots & 1 \end{pmatrix}, \tag{3.107}$$

která je symetrická. Na diagonále má tato matice samé jedničky, mimo diagonálu jednotlivé koeficienty korelace.

Podívejme se dále na odhady parametrů polohy a variability. Z vícerozměrného náhodného výběru velikosti n, který je popsán zdrojovou maticí $X \in \mathbb{R}_{n \times m}$, kde *i*-tý řádek matice popisuje *m*-rozměrný vektor $x_i = (x_{i1}, x_{i2}, ..., x_{im})^{\mathrm{T}}$, určíme *vektor odhadů středních hodnot (vektor průměrů, výběrový vektor středních hodnot)* vztahem

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \,, \tag{3.108}$$

rozepsáno po souřadnicích

(3.109) $$\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij},$$

kde $j = 1, 2, ..., m$.

Vektor průměrů z (3.108) můžeme zapsat také pomocí následujícího vztahu

(3.110) $$\overline{x} = \frac{1}{n} X^{\mathrm{T}} 1,$$

kde 1 je sloupcový vektor o *n* složkách, které se všechny rovnají jedničce.

Odhad kovarianční matice (výběrová kovarianční matice) je definován vztahem

(3.111) $$S^0 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^{\mathrm{T}},$$

což můžeme opět zapsat maticově

(3.112) $$S^0 = \frac{1}{n} X^{\mathrm{T}} U X,$$

pro matici U definovanou jako

(3.113) $$U = E - \frac{1}{n} 1 1^{\mathrm{T}}.$$

Matice E v předpisu (3.113) je jednotková matice řádu n.

Lze ukázat, že pro vektor průměrů definovaného v (3.108) platí

(3.114) $$E(\overline{x}) = \mu \text{ a } D(\overline{x}) = \frac{1}{n} C.$$

Pro odhad kovarianční matice je možné odvodit

(3.115) $$E(S^0) = \frac{n-1}{n} C.$$

Ze vztahů (3.114) a (3.115) je tedy patrné, že vektor odhadů $\overline{x}$ je nevychýlený, kdežto odhad kovarianční matice je vychýlený. Proto se podobně jako u jednorozměrných dat používá tzv. *výběrová korigovaná kovarianční matice*

(3.116) $$S = \frac{n}{n-1} S^0 ,$$

která je již nevychýleným odhadem kovarianční matice C.

Popsané pojmy a definice potřebné k zavedení vícerozměrné statistické metody jsme převzali ze zmíněné literatury (Jackson, 2003; Jolliffe, 2002; Meloun a Militký, 2004), přičemž jsme se drželi značení použitého v (Meloun a Militký, 2004). Pro podrobnější informace o této problematice odkazujeme čtenáře rovněž na tyto zdroje.

Přejděme nyní k popisu metody PCA. Jak již víme z úvodu tohoto oddílu, základním cílem PCA je transformovat původní soubor dat $x_i = (x_{i1}, x_{i2}, ..., x_{im})^{\mathrm{T}}$, $i = 1, 2, ..., n$, do menšího počtu nových proměnných $y_1, y_2, ..., y_r$, které nazýváme hlavní komponenty.

Popišme si nejdříve situaci pro náhodný vektor $\xi = (\xi_1, \xi_2, ..., \xi_m)^{\mathrm{T}}$, jehož složky jsou náhodné veličiny, a pro případ kdy známe střední hodnoty $E(\xi_j) = \mu_j$, $j = 1, 2, ..., m$ a kovarianční matici C.

Nyní chceme z náhodných veličin $\xi_1, \xi_2, ..., \xi_m$ vytvořit menší počet náhodných veličin, které budou co nejlépe nahrazovat původní systém. Nové náhodné veličiny budeme konstruovat jako lineární kombinace náhodných veličin $\xi_1, \xi_2, ..., \xi_m$, přičemž požadujeme, aby byly tyto lineární kombinace navzájem nekorelované a postupně vyčerpávaly maximálním možným způsobem variabilitu původního systému. To znamená, že první hlavní komponenta y_1 je takovou lineární kombinací vstupních znaků, která má největší rozptyl mezi všemi ostatními lineárními kombinacemi. Hlavní komponenta y_1 má tvar

(3.117) $$y_1 = \sum_{j=1}^{m} v_{1j} \xi_j = v_1^{\mathrm{T}} \xi ,$$

kde $v_1 \in \mathbb{R}_m$ je jednotkový vektor koeficientů.

Víme, že kovarianční matice $C \in \mathbb{R}_{m \times m}$ definovaná v (3.106) je symetrická. Lze dokázat, že je také pozitivně semidefinitní, tedy pro libovolný nenulový vektor $y \in \mathbb{R}_m$ platí $y^{\mathrm{T}} C y \geq 0$, což je ekvivalentní s tím, že všechna vlastní čísla matice C jsou nezáporná. Podle věty 3.4 víme, že pro matici C existuje ortonormální matice $Q \in \mathbb{R}_{m \times m}$ ($Q^{\mathrm{T}} Q = E$) a diagonální matice $\Lambda \in \mathbb{R}_{m \times m}$ tak, že $C = Q \Lambda Q^{\mathrm{T}}$. Dále víme, že vlastní čísla matice Λ jsou právě prvky její diagonály λ_j, $j = 1, 2, ..., m$ a jsou to rovněž všechna vlastní čísla matice C. Sloupce ortonormální matice Q jsou ortonormální vlastní vektory kovarianční matice C příslušné vlastním číslům λ_j, $j = 1, 2, ..., m$.

Podle definice můžeme pro vlastní vektory $v_j, j = 1, 2, ..., m$ matice C příslušné vlastním číslům λ_j, $j = 1, 2, ..., m$ psát $C v_j = \lambda_j v_j$. Uspořádáme-li vlastní čísla kovarianční matice C, tj. $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m \geq 0$, z věty 3.3 plyne, že jednotkový vlastní vektor v_1 příslušný největšímu vlastnímu číslu λ_1 maximalizuje kvadratickou formu $v^{\mathrm{T}} C v$. Maximum této kvadratické formy je tedy právě největší vlastní číslo λ_1. Pro každý vektor $v \in \mathbb{R}_m$ splňující podmínku $\|v\| = 1$ tedy platí nerovnost $v^{\mathrm{T}} C v \leq \lambda_1$.

Hlavní komponenta y_1 popsaná v (3.117) má maximální rozptyl, neboť platí, že $D(y_1) = D(v_1^{\mathrm{T}}\xi) = v_1^{\mathrm{T}} C v_1$, viz (Anděl, 1985) nebo (Meloun a Militký, 2004). Z předchozích vztahů rovněž plyne, že maximální hodnota $D(y_1)$ je λ_1.

Druhá hlavní komponenta má tvar

$$y_2 = \sum_{j=1}^{m} v_{2j}\xi_j = v_2^{\mathrm{T}}\xi, \tag{3.118}$$

kde $v_2 \in \mathbb{R}_m$ je jednotkový vektor koeficientů, je nekorelovaná s náhodnou veličinou y_1 a její rozptyl $D(y_2)$ je maximální. Podmínka nekorelovanosti říká, že

$$\mathrm{cov}(y_2, y_1) = \mathrm{cov}(v_2^{\mathrm{T}}\xi, v_1^{\mathrm{T}}\xi) = v_2^{\mathrm{T}} C v_1 = 0. \tag{3.119}$$

Pro jednotkový vlastní vektor v_1 matice C a jemu příslušné vlastní číslo λ_1 je podle definice $Cv_1 = \lambda_1 v_1$. Můžeme tedy psát

$$v_2^{\mathrm{T}} C v_1 = v_2^{\mathrm{T}} \lambda_1 v_1 = \lambda_1 v_2^{\mathrm{T}} v_1. \tag{3.120}$$

Předpokládáme, že kovarianční matice C má právě r kladných vlastních čísel a že jsou tato čísla navzájem různá. (V praxi je většinou počet r nenulových vlastních čísel matice C roven m.) Máme největší vlastní číslo $\lambda_1 > 0$ a veličiny y_1 a y_2 jsou tedy nekorelované právě tehdy, když $v_2^{\mathrm{T}} v_1 = 0$.

Hlavní komponenta y_2 má rozptyl $D(y_2) = D(v_2^{\mathrm{T}}\xi) = v_2^{\mathrm{T}} C v_2$ a z předchozích vztahů opět plyne, že maximální hodnota $D(y_2)$ je λ_2.

Obdobně budeme postupovat dále. S ohledem na naše aplikace, kdy pracujeme buď v eukleidovské rovině E_2 nebo v eukleidovském prostoru E_3, tedy $m = 2$ nebo $m = 3$, uveďme ještě odvození třetí hlavní komponenty y_3, tj.

$$y_3 = \sum_{j=1}^{m} v_{3j}\xi_j = v_3^{\mathrm{T}}\xi, \tag{3.121}$$

kde $v_3 \in \mathbb{R}_m$ je jednotkový vektor koeficientů. Hlavní komponenta y_3 je nekorelovaná s náhodnými veličinami y_1 a y_2 a její rozptyl $D(y_3)$ je maximální. To znamená, že $v_3^{\mathrm{T}} v_1 = 0$ a $v_3^{\mathrm{T}} v_2 = 0$ a maximální hodnota $D(y_3)$ je λ_3.

Obecně získáme hlavní komponenty $y_1, y_2, ..., y_r$, pro které

$$y_j = v_j^{\mathrm{T}}\xi,\ D(y_j) = \lambda_j,\ \mathrm{cov}(y_j, y_i) = 0 \tag{3.122}$$

pro všechna $i < j$ a $j = 1, 2, ..., r$. Jak jsme již uvedli, v praxi bývá počet r hlavních komponent roven m. V obecné úloze nemusí být vlastní čísla navzájem různá, v takovém případě další hlavní komponentu vybíráme libovolně.

Hlavní komponenty mají zajímavé vlastnosti, odkazujeme čtenáře na literaturu např. (Anděl, 1985).

Na metodu PCA lze nahlížet jako na problém maximalizace rozptylů hlavních komponent. Tento klasický přístup jsme právě popsali. Lze však rovněž dokázat, že jde o minimalizaci eukleidovských vzdáleností původních znaků a jejich ortogonálních projekcí na hlavní komponenty, důkaz viz (Anděl, 1985; Meloun a Militký, 2004).

Při praktických aplikacích máme k dispozici zdrojovou matici dat $X \in \mathbb{R}_{n\times m}$, která představuje náhodný výběr velikosti n. Místo kovarianční matice C, kterou nyní nemáme k dispozici, pracujeme s výběrovou kovarianční maticí S^0 s vlastními vektory $c_j, j = 1,2,...,m$ příslušnými vlastním číslům $\bar{\lambda}_j$, $j = 1,2,...,m$. Abychom mohli výsledky interpretovat vizuálně, zavedeme vektor $Y_j \in \mathbb{R}_n$, pro který platí $Y_j = \left(c_j^{\mathrm{T}}(X - \bar{X})^{\mathrm{T}}\right)^{\mathrm{T}} = (X - \bar{X})c_j$, $j = 1,2,...,m$, kde matice $\bar{X} \in \mathbb{R}_{n\times m}$ má na každém řádku vektor průměrů definovaný v (3.108). Souřadnice vektoru Y_j lze interpretovat jako pozorování *j*-té hlavní komponenty y_j.

Vlastní vektory $c_j, j = 1,2,...,m$ matice S^0 příslušné vlastním číslům $\bar{\lambda}_j$, $j = 1,2,...,m$ můžeme chápat jako směry nového souřadnicového systému, kterému budeme říkat *souřadnicový systém hlavních komponent*. Původní objekty mají v tomto novém systému za souřadnice složky vektorů Y_j. Počátek tohoto souřadnicového systému odpovídá průměrnému objektu, v našem případě aritmetickému průměru vstupních bodů. Fungování metody ukážeme na příkladě.

Příklad 3.5: *ANALÝZA HLAVNÍCH KOMPONENT.* Popsanou metodu nyní aplikujme na hledání osových symetrií bodových množin a to jak v eukleidovské rovině E_2, tak v eukleidovském prostoru E_3.

Mějme dánu zdrojovou matici X, která je tvořena *n*-ticí vektorů $x_i = (x_i^x, x_i^y)^{\mathrm{T}}$, kde $i = 1,2,...,n$. Jednotlivé řádky matice X chápeme jako souřadnice n bodů v eukleidovské rovině E_2. Jedná se o stejné body, které byly použity v příkladě 3.2 při ortogonálním prokládání dat. Souřadnice jednotlivých bodů jsou tedy dány tabulkou 3.2. Volba tohoto vstupu je zřejmá, naším cílem je obě metody porovnat a ukázat, že dostáváme stejné výsledky.

Předložen je dále podrobný postup výpočtu vedoucí k nalezení nového souřadnicového systému hlavních komponent. Výsledný souřadnicový systém hlavních komponent je zobrazen na obrázku 3.62 v původní soustavě souřadnic, nové osy značíme o_1 a o_2. Na obrázku 3.63 jsou vstupní body zobrazeny v souřadnicovém systému hlavních komponent. Body mají v tomto novém systému za souřadnice složky vektorů Y_1 a Y_2. Pokud bychom vyjádřili souřadnice počátku (tj. průměru bodů) v novém souřadnicovém systému, získáme bod o souřadnicích $[0,0]$. (Ve všech případech jsou číselné hodnoty při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Dále jsou zpracovány obecnější bodové množiny. Souřadnice jednotlivých bodů nevypisujeme, neboť se jedná již o rozsáhlejší data. Pro případ v eukleidovské rovině E_2 jsou použity stejné body jako v příkladě ortogonálního prokládání dat znázorněném na obrázku 3.50. Pro případ v eukleidovském prostoru E_3 se potom jedná

o body použité na obrázku 3.54. Výsledné souřadnicové systémy hlavních komponent jsou znázorněny na obrázcích 3.64 a 3.65.

Výstupy jsou získány z výpočetního prostředí MATLAB. V programech řešíme hledání nového souřadnicového systému metodou PCA na základě odvozených vztahů, přičemž k výpočtu vlastních vektorů a příslušných vlastních čísel jsou použity funkce MATLABu. Kvůli názornosti doporučujeme čtenáři prohlédnout si výstup metody v prostoru přímo ve výpočetním prostředí MATLAB, kde lze s výsledným obrázkem pohybat.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu): `PCA_2d.m` (programy/pca),
`PCA_3d.m` (programy/pca).

Výpočet souřadnic počátku nového souřadnicového systému

$$O = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad O = \left[O^x, O^y\right]$$

$$O^x = \frac{1}{n}\sum_{i=1}^{n} x_i^x = \frac{1}{21}\cdot 113{,}5 = 5{,}40476$$

$$O^y = \frac{1}{n}\sum_{i=1}^{n} x_i^y = \frac{1}{21}\cdot 22 = 1{,}04762$$

$$O = [5{,}40476; 1{,}04762]$$

Zdrojová matice X

$$X = \begin{pmatrix} -2 & -3 \\ 0 & -4 \\ 1 & -2 \\ 2 & 1 \\ 2{,}5 & -2 \\ 3 & -3 \\ 4 & 1 \\ 4{,}5 & -2 \\ 5 & 1{,}5 \\ 5{,}5 & 3 \\ 5{,}5 & -1{,}5 \\ 6 & 1 \\ 6 & 2 \\ 7 & 4 \\ 8 & 1{,}5 \\ 8{,}5 & 4 \\ 9 & 1 \\ 9 & 4{,}5 \\ 9 & 5 \\ 10 & 4 \\ 10 & 6 \end{pmatrix} \qquad X - \overline{X} = \begin{pmatrix} -7{,}40476 & -4{,}04762 \\ -5{,}40476 & -5{,}04762 \\ -4{,}40476 & -3{,}04762 \\ -3{,}40476 & -0{,}04762 \\ -2{,}90476 & -3{,}04762 \\ -2{,}40476 & -4{,}04762 \\ -1{,}40476 & -0{,}04762 \\ -0{,}90476 & -3{,}04762 \\ -0{,}40476 & 0{,}45238 \\ 0{,}09524 & 1{,}95238 \\ 0{,}09524 & -2{,}54762 \\ 0{,}59524 & -0{,}04762 \\ 0{,}59524 & 0{,}95238 \\ 1{,}59524 & 2{,}95238 \\ 2{,}59524 & 0{,}45238 \\ 3{,}09524 & 2{,}95238 \\ 3{,}59524 & -0{,}04762 \\ 3{,}59524 & 3{,}45238 \\ 3{,}59524 & 3{,}95238 \\ 4{,}59524 & 2{,}95238 \\ 4{,}59524 & 4{,}95238 \end{pmatrix}$$

Odhad kovarianční matice S^0

$$S^0 = \begin{pmatrix} 11{,}08617 & 8{,}11168 \\ 8{,}11168 & 8{,}37868 \end{pmatrix}$$

Výpočet vlastních čísel a příslušných vlastních vektorů matice S^0

$$\lambda E - S^0 = \begin{pmatrix} \lambda - 11{,}08617 & -8{,}11168 \\ -8{,}11168 & \lambda - 8{,}37868 \end{pmatrix}$$

$$\det(\lambda E - S^0) = (\lambda - 11,08617)(\lambda - 8,37868) - 65,79932 =$$
$$= \lambda^2 - 19,46485\lambda + 27,08819$$

charakteristický polynom

kořeny charakteristického polynomu
$\lambda^2 - 19,46485\lambda + 27,08819 = 0$

Vlastní čísla

$\bar{\lambda}_1 = 17,95629 \qquad \bar{\lambda}_2 = 1,50856$

jednotkové vlastní vektory příslušné vlastním číslům jsou netriviálním řešením homogenních soustav s maticemi $\bar{\lambda}_1 E - S^0$ a $\bar{\lambda}_2 E - S^0$

$$\begin{pmatrix} 6,87012 & -8,11168 \\ -8,11168 & 9,57761 \end{pmatrix}\begin{pmatrix} c_1^x \\ c_1^y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$(\bar{\lambda}_1 E - S^0)c_1 = o$$

$$\begin{pmatrix} -9,57761 & -8,11168 \\ -8,11168 & -6,87012 \end{pmatrix}\begin{pmatrix} c_2^x \\ c_2^y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$(\bar{\lambda}_2 E - S^0)c_2 = o$$

řešení soustav

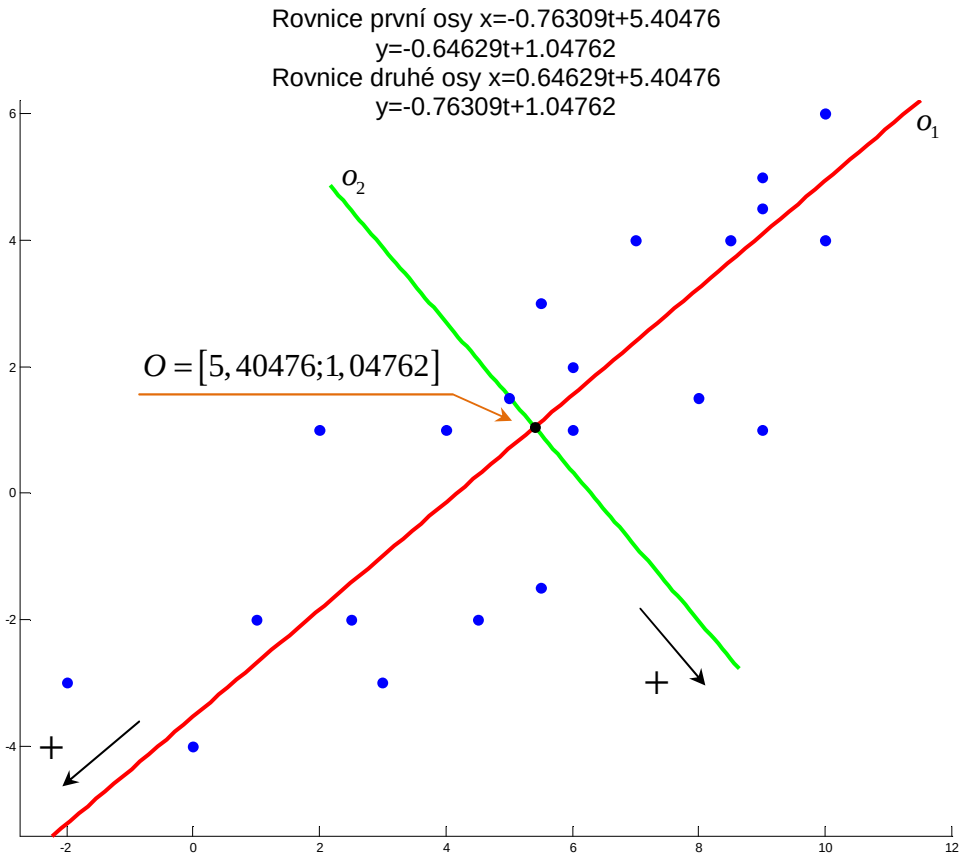
jednotkové vlastní vektory c_1 a c_2

$c_1 = (-0,76309; -0,64629)^T$
$c_2 = (0,64629; -0,76309)^T$

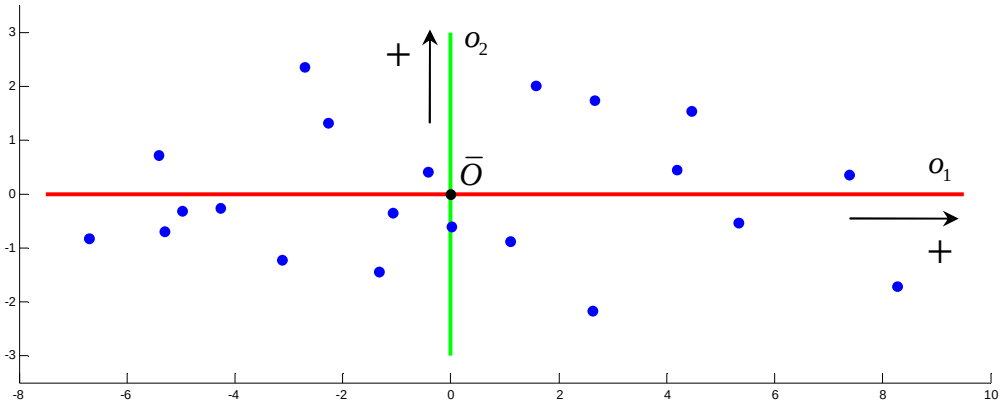
Souřadnice bodů v novém souřadnicovém systému

$$Y_1 = (X - \bar{X})c_1 = \begin{pmatrix} 8,26644 \\ 7,38656 \\ 5,33088 \\ 2,62891 \\ 4,18625 \\ 4,451 \\ 1,10273 \\ 2,66007 \\ 0,0165 \\ -1,33448 \\ 1,57383 \\ -0,42344 \\ -1,06974 \\ -3,12541 \\ -2,27277 \\ -4,27005 \\ -2,71271 \\ -4,97474 \\ -5,29788 \\ -5,41468 \\ -6,70727 \end{pmatrix} \qquad Y_2 = (X - \bar{X})c_2 = \begin{pmatrix} -1,69695 \\ 0,35873 \\ -0,52116 \\ -2,16414 \\ 0,44828 \\ 1,53452 \\ -0,87155 \\ 1,74087 \\ -0,6068 \\ -1,42829 \\ 2,00561 \\ 0,42104 \\ -0,34205 \\ -1,22194 \\ 1,33208 \\ -0,2525 \\ 2,35991 \\ -0,3109 \\ -0,69244 \\ 0,71694 \\ -0,80924 \end{pmatrix}$$

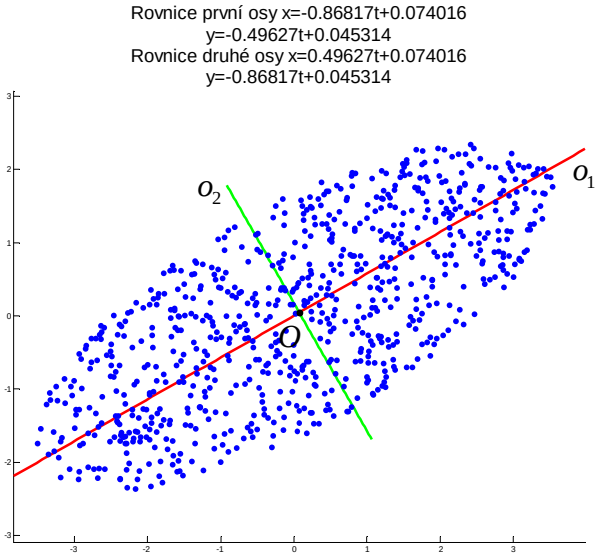
Po dosazení průměru bodů ověříme, že souřadnice počátku v novém souřadnicovém systému je $\bar{O} = [0,0]$



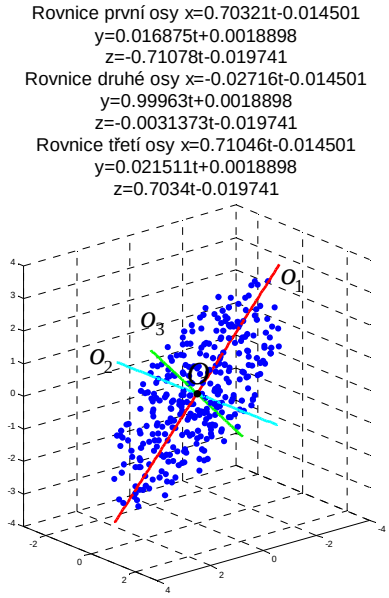
Obrázek 3.62: Princip metody PCA – nový souřadnicový systém hlavních komponent



Obrázek 3.63: Princip metody PCA – nový souřadnicový systém hlavních komponent



Obrázek 3.64: Metoda PCA – nový souřadnicový systém hlavních komponent v rovině



Obrázek 3.65: Metoda PCA – nový souřadnicový systém hlavních komponent v prostoru

Z výpočtů je zřejmé, že první osa nového souřadnicového systému hlavních komponent, jejíž směr odpovídá prvnímu vlastnímu vektoru c_1, je totožná s přímkou, kterou jsme získali metodou ortogonálního prokládání dat. Druhá přímka, kterou jsme určovali jako přímku kolmou k přímce získanou ortogonálním prokládáním v oddílu 3.3.3, odpovídá druhé ose nového souřadnicového systému, tj. přímce, jejíž směr je dán druhým vlastním vektorem c_2. Tyto dvě přímky se společným bodem získaným jako průměrem všech bodů vstupní množiny jsme označovali za dva hlavní směry bodové množiny v případech v eukleidovské rovině E_2, nyní víme, že odpovídají osám nového souřadnicového systému hlavních komponent, viz obrázek 3.64. Pokud jsme pracovali s eukleidovským prostorem E_3, určovali jsme ještě třetí přímku, která byla kolmá na první dvě a procházela jejich průsečíkem. Tři kolmé přímky v prostoru se společným bodem jsme označili jako tři hlavní směry bodové množiny. Tyto tři přímky jsou totožné s osami nového souřadnicového systému hlavních komponent, viz obrázek 3.65.

V případě ortogonálního prokládání dat v prostoru rovinou nalezneme rovněž analogii s metodou PCA. Opět se jedná o maximalizaci rozptylů hlavních komponent, přičemž nás zajímají pouze první dvě hlavní komponenty. V souřadnicovém systému hlavních komponent tedy první dvě osy o_1 a o_2 určují rovinu, která odpovídá rovině získané ortogonálním prokládáním dat. Výslednou rovinu lze chápat jako souřadnicovou rovinu nového systému hlavních komponent. V případě vstupních bodových množin, které jsou symetrické podle tří navzájem kolmých rovin, lze tyto roviny nalézt právě pomocí metody PCA jako souřadnicové roviny systému hlavních komponent.

V dalším textu budeme odvozené metody používat k řešení dílčích problémů v rekonstrukci povrchů již z reálných vstupních bodových množin. Obecně se ve zpracování dat rozhodujeme častěji pro metodu ortogonálního prokládání dat přímkou nebo rovinou než pro metodu hlavních komponent a to především kvůli její geometrické názornosti. Z předchozího ale víme, že se jedná o ten samý postup, rozdíl je pouze v metodě výpočtu.

Kapitola 4

Analýza bodové množiny – iterační metody

V této kapitole jsou předloženy nově navržené algoritmy pro zpracování vstupních bodových množin. Funkčnost a vlastnosti navržených algoritmů jsou ověřovány na počítačově generovaných množinách bodů. Analýze bodové množiny je záměrně věnována velká pozornost, neboť na jejím úspěšném provedení závisí kvalita předzpracování dat, které je pak použito jako vstup pro další navazující techniky v polygonální a tvarové fázi rekonstrukce povrchů. Bez předložené analýzy bodových množin se proto nelze v dalších částech rekonstrukce povrchů obejít.

Vlastní postupy řešení jsou rozvinuty pro hledání os rotačních válcových ploch (oddíl *Hledání osy rotační válcové plochy*) a zobecněny jsou na libovolné rotační plochy (oddíl *Hledání osy obecné rotační plochy*). Bodové množiny jsou analyzovány rovněž z hlediska rovinových symetrií (oddíl *Hledání rovinových symetrií*) podle jedné roviny nebo dvou a tří vzájemně kolmých rovin.

Všechny prezentované techniky jsou založeny na diferenciálních numerických metodách vícerozměrné minimalizace. Abychom se mohli na existující metody vícerozměrné minimalizace dále odkazovat, uvádíme nezbytnou teorii (oddíl *Diferenciální numerické metody*). Vlastní algoritmy, které jsou v této kapitole představeny, hledají řešení jmenovaných problémů iteračním způsobem. Tento postup jsme volili pro jeho názornost a možnost přímého využití geometrického významu zkoumaných ploch v návrhu metod a algoritmů. Zkoumané problémy rekonstrukce jsou v navržených metodách řešeny vždy s využitím geometrických vlastností rekonstruovaných povrchů. Při hledání řešení si totiž klademe za cíl co nejvíce zdůraznit důležitost znalosti prostorové geometrie v aplikačních oblastech, jakou je právě digitální rekonstrukce povrchů. Algoritmy navrhujeme tak, aby využívaly elementárních geometrických vlastností popisovaného objektu. Takový přístup usnadňuje další využití navržených algoritmů a stejně tak jejich implementaci, která pak speciálních geometrických vlastností může využít ke zvýšení efektivity. Názorný řešící postup založený na určitém jednoduchém geometrickém vztahu bývá často velice obtížné odhalit, o to je jeho nalezení cennější.

Všechny nově navržené algoritmy implementujeme ve výpočetním prostředí MATLAB. Z MATLABu rovněž získáváme grafické výstupy, které doprovázejí veškeré geometrické algoritmy. Kvůli přehlednosti a lepšímu porozumění algoritmům programy zapisujeme také pomocí symbolického kódu splňujícím tradičně používané normy zápisu programů v procedurálních jazycích (typu Pascal, C, resp. procedurální podmnožiny jazyků Java, C++, atd.). Dále předkládáme celou řadu příkladů s názornými ilustracemi demonstrujících funkčnost odvozených postupů, přičemž v rámci této kapitoly testujeme programy výhradně na počítačově generovaných bodových množinách. Syntetická data zde používáme záměrně,

neboť u počítačově generovaných bodových množin můžeme snadno regulovat jejich velikost, což následně umožní přehlednější a rychlejší zpracování, a dále předem víme, jakou geometrickou vlastnost tyto bodové množiny splňují, lze proto ověřit správnost řešení. Testovací bodové množiny vytváříme v modelovacím softwaru Rhinoceros (regulární množiny) nebo ve výpočetním prostředí MATLAB (regulární množiny, náhodně generované body na povrchu ploch nebo zašuměné body). Naimplementované programy a konkrétní počítačově generované vstupní bodové množiny jsou k dispozici rovněž na přiloženém výměnném médiu.

4.1 Diferenciální numerické metody

V tomto oddíle popíšeme známou minimalizační metodu založenou na výpočtu gradientu funkce. Na základě této numerické metody navrhujeme v oddílech 4.2, 4.3 a 4.4 nové řešící postupy pro dílčí problémy rekonstrukce povrchu zkoumaného objektu. Pracujeme stále s bodovou množinou a diferenciální numerické metody používáme ke zjištění osových a rovinových symetrií vstupních bodových množin.

4.1.1 Gradientní metody hledání extrému

V úlohách budeme často potřebovat minimalizovat reálné funkce více proměnných. V následujícím oddíle proto popíšeme často užívanou gradientní *metodu největšího spádu*, (Yang, 2008; Došlý, 2005). K zavedení této optimalizační techniky je třeba nadefinovat základní pojmy diferenciálního počtu funkcí více proměnných.

Definice 4.1 (*Totální diferenciál*). Nechť reálná funkce *d* reálných proměnných $f:\mathbb{R}_d \to \mathbb{R}$ je definovaná na nějakém okolí $U(a) \subset \mathbb{R}_d$, bodu $a=(a_1,a_2,...,a_d)\in\mathbb{R}_d$. *Totální diferenciál funkce f v bodě a* je lineární zobrazení $df(a):\mathbb{R}_d \to \mathbb{R}$ tvaru

$$df(a)(h)=\sum_{i=1}^{d}\alpha_i h_i,$$

kde $\alpha_i \in \mathbb{R}$, $i=1,2,...,d$ a $h=(h_1,h_2,...,h_d)^{\mathrm{T}}\in\mathbb{R}_d$ takové, že platí

$$\lim_{h\to o}\frac{\eta(h)}{h}=0, \text{ kde } \eta(h)=f(a+h)-f(a)-df(a)(h). \blacksquare$$

Definice 4.2 (*Derivace funkce ve směru*). Nechť jsou dány reálná funkce *d* reálných proměnných $f:\mathbb{R}_d \to \mathbb{R}$ definovaná na nějakém okolí $U(a) \subset \mathbb{R}_d$, bodu $a=(a_1,a_2,...,a_d)\in\mathbb{R}_d$,

a jednotkový vektor $v = (v_1, v_2, ..., v_d)^T \in \mathbb{R}_d$. Potom *derivací funkce f v bodě a ve směru v* nazýváme vlastní limitu (pokud existuje)

$$\lim_{t \to 0} \frac{f(a+tv) - f(a)}{t},$$

kterou označujeme $\frac{\partial f}{\partial v}(a)$. ■

Definice 4.3 (*Gradient funkce*). Gradientem funkce f v bodě a nazýváme vektor

$$\nabla f(a) = \left(\frac{\partial f}{\partial x_1}(a), \frac{\partial f}{\partial x_2}(a), ..., \frac{\partial f}{\partial x_d}(a) \right)^T,$$

kde $\frac{\partial f}{\partial x_i}(a)$ značí parciální derivaci funkce f podle proměnné x_i v bodě a. ■

Gradient má pro reálnou funkci f více proměnných důležitý význam. Pokud má funkce f v bodě a totální diferenciál a je-li $\nabla f(a) \neq 0$, potom pro každý jednotkový vektor $v \in \mathbb{R}_d$ platí

$$\left| \frac{\partial f}{\partial v}(a) \right| \leq \left\| \nabla f(a) \right\|, \tag{4.1}$$

což plyne z Cauchyovy-Schwarzovy nerovnosti (Bečvář, 2002) a z rovnosti (Kopáček, 2003)

$$\frac{\partial f}{\partial v}(a) = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i}(a) v_i \,. \tag{4.2}$$

Rovnost v (4.1) nastává pro případ $v = \nabla f(a) / \left\| \nabla f(a) \right\|$. Gradient tedy udává směr, ve kterém je absolutní hodnota derivace ve směru největší (tj. největší ze všech derivací ve směru), tedy směr, v němž maximálně roste funkční hodnota.

Připomeňme dále definici lokálního minima reálné funkce více reálných proměnných.

Definice 4.4 (*Lokální minimum reálné funkce více proměnných*). Nechť funkce $f : \mathbb{R}_d \to \mathbb{R}$ je definovaná na množině $M \subset \mathbb{R}_d$ a $a = (a_1, a_2, ..., a_d) \in M$. Řekneme, že funkce f má v bodě a *lokální minimum vzhledem k M*, jestliže existuje takové okolí $U(a)$, že $f(a) \leq f(x)$ pro $x \in U(a) \cap M$. ■

Definice 4.1, 4.2, 4.3 a 4.4 jsou převzaty z (Kopáček, 2003).

Přejděme nyní k úloze minimalizace reálné funkce f více reálných proměnných definované na množině $M \subset \mathbb{R}_d$. Volíme gradientní metodu největšího spádu, která patří mezi základní iterační minimalizační metody diferencovatelných funkcí více proměnných. Zaveďme tuto metodu nejdříve intuitivně. Zvolíme počáteční bod, který je obecně různý od hleda-

ného bodu, kde funkce nabývá lokálního minima, a určíme v tomto bodě směr, v němž funkce nejvíce klesá. Z počátečního bodu se posuneme do dalšího bodu nacházejícího se v tomto směru. Tento postup opakujeme, dokud není splněna nějaká předem daná podmínka, která iterační proces ukončí. Postupnou iterací získáme tzv. *minimalizující posloupnost bodů*. Funkční hodnota v posledním bodě této posloupnosti je odhadem lokálního minima zadané reálné funkce více proměnných. Obtíže mohou nastat v případě, nabývá-li funkce lokálního minima ve více bodech. Lze dokázat, že pokud se dostaneme do určitého okolí lokálního minima, bude tato metoda vždy k tomuto minimu konvergovat, (Yang, 2008).

Popišme metodu největšího spádu formálně. Budeme pracovat s předpokladem, že minimalizovaná funkce má v každém bodě parciální derivace a totální diferenciál. Poznamenejme, že tento předpoklad je použit pouze k odvození metody a v reálné úloze nemusí platit. Počáteční bod označme jako x_0 a předpokládejme, že je různý od bodu x^*, ve kterém funkce nabývá lokálního minima. Sestrojíme minimalizující posloupnost bodů, kde v $(k+1)$. kroku získáme nový bod posloupnosti jako

$$x_{k+1} = x_k - \lambda_k \nabla f(x_k). \tag{4.3}$$

V gradientní metodě největšího spádu tedy volíme směr posunutí do dalšího bodu tak, abychom získali co největší pokles. Pro posloupnost bodů bude požadováno, aby platilo

$$x_{k+1} = \arg\min_{\lambda>0} f(x_k - \lambda \nabla f(x_k)). \tag{4.4}$$

Délka kroku λ_k se nejčastěji volí jednorozměrnou minimalizací, neboť můžeme psát

$$\min_{\lambda>0} f(x_k - \lambda \nabla f(x_k)) = \min_{\lambda>0} \varphi_k(\lambda) = \varphi_k(\lambda_k), \tag{4.5}$$

kde λ_k je bod, ve kterém jednorozměrná funkce φ_k nabývá minima.

Lze dokázat, že směry $\nabla f(x_k)$ a $\nabla f(x_{k+1})$ jsou navzájem ortogonální. Tuto skutečnost ukážeme na konkrétním příkladě. Postup určení dalšího bodu minimalizující posloupnosti budeme opakovat, dokud nebude splněna podmínka ukončení iteračního procesu. Volíme proto vhodné malé $\varepsilon \in (0,1)$ a iteraci ukončíme v *k*-tém kroku, je-li splněno $\|\nabla f(x_k)\| < \varepsilon$. Funkční hodnota v bodě x_k je potom odhadem lokálního minima zadané reálné funkce více proměnných.

Jednorozměrnou minimalizaci funkce φ_k realizujeme pomocí *Newtonovy metody tečen* (Došlý, 2005). Jedná se o iterační metodu, kdy předem neznáme počet kroků. Opřeme se o tvrzení, které budeme dále používat jako předpoklad. Nabývá-li jednorozměrná reálná funkce φ_k v nějakém bodě $\lambda^* \in \mathbb{R}$ svého lokálního minima, pak $\varphi_k'(\lambda^*) = 0$. Předpokládejme, že je funkce φ_k dvakrát diferencovatelná. Počáteční bod λ_0 pro minimalizaci zvolíme a další body posloupnosti, která konverguje k hledanému bodu s minimální funkční hodnotou, spočítáme pomocí vztahu

(4.6) $$\lambda_{j+1} = \lambda_j - \frac{\varphi_k'(\lambda_j)}{\varphi_k''(\lambda_j)}.$$

Vztah (4.6) lze jednoduše vysvětlit na základě geometrické představy. V každém kroku iterace určujeme v bodě λ_j tečnu ke grafu funkce $\varphi_k'(\lambda)$. Průsečík této tečny s osou x je další bod minimalizující posloupnosti. To znamená, že rovnici tečny ke grafu funkce $\varphi_k'(\lambda)$ pokládáme rovnou nule a po úpravě dostáváme vztah (4.6).

Iterační proces ukončíme, jestliže pro vhodně zvolené malé $\varepsilon \in (0,1)$ platí $\left|\lambda_{j+1} - \lambda_j\right| < \varepsilon$. Potom funkční hodnotu v bodě λ_{j+1} označíme jako odhad lokálního minima zadané reálné funkce jedné proměnné.

V našich experimentech jsme začínali s jednodušší verzí vícerozměrné minimalizace a to *modifikovanou metodou největšího spádu*, kde jsme délku kroku λ_k volili konstantní. Tato volba se pro dané aplikační oblasti ukázala v mnohých případech jako plně dostačující. Pokud to bylo nutné, konstantní délku kroku jsme v minimalizačním procesu nahradili délkou určenou jednorozměrnou minimalizací a přešli tak k obecné metodě největšího spádu. Obecná metoda největšího spádu představuje přesnější minimalizační techniku.

Popsané metody nyní porovnejme na příkladech minimalizace reálné funkce dvou reálných proměnných.

Příklad 4.1: *MINIMALIZACE REÁLNÉ FUNKCE DVOU REÁLNÝCH PROMĚNNÝCH*. Popsané metody nyní použijme k minimalizaci reálných funkcí dvou reálných proměnných, nejdříve pracujme s jednoduchou funkcí s předpisem $f(x,y) = 0{,}1x^2 + 0{,}09y^2$. Funkce má lokální minimum v bodě $(0,0)$, hodnota tohoto minima je 0.

Začněme s obecnou metodou největšího spádu. Počáteční bod pro iterační proces je zvolen $x_0 = (8,6)$. Podmínka pro ukončení iterace je určena hodnotou $\varepsilon = 0{,}0001$ (v praktických aplikacích je tato hodnota menší). Pro lepší představu je podrobně rozepsán první krok minimalizace. Na obrázku 4.1 je znázorněn graf této funkce, tedy plocha rotačního paraboloidu o ose z, a minimalizující posloupnost bodů. Odhad lokálního minima pro tuto volbu parametru ε vychází hodnota 0 (zaokrouhleno) v bodě $(-0{,}00001; -0{,}00001)$. Tento výsledek získáme již po 4 iteracích. Je patrné, že metoda konverguje k lokálnímu minimu velice rychle v porovnání s další metodou. V pohledu shora je zvýrazněna ortogonalita směrů v bodech minimalizující posloupnosti. Body blízko minima se téměř překrývají, proto je nepopisujeme.

V modifikované metodě největšího spádu vycházíme ze stejného bodu, tedy $x_0 = (8,6)$. Délku kroku λ_k volíme konstantní a to $\lambda_k = 0{,}5$. Podmínka pro ukončení iterace je rovněž $\varepsilon = 0{,}0001$. Odhad lokálního minima pro tuto volbu parametrů λ_k a ε vychází hodnota 0 (zaokrouhleno) v bodě $(-0{,}00479; -0{,}00452)$. Na obrázku 4.2 je znázorněn opět graf této funkce s minimalizující posloupností bodů a připojen je rovněž pohled shora. V tomto případě již výpočty nevypisujeme, neboť se jedná o jednodušší úlohu, ve které oproti předchozí metodě chybí výpočet jednorozměrné minimalizace. Výsledky výpočtů lze získat spuštěním programu vytvořeného ve výpočet-

ním prostředí MATLAB. Kvůli konstantní délce kroku jsme hledané minimum nepatrně přeskočili. Pokud bychom vypsali hodnoty minimalizující posloupnosti, zjistili bychom, že v seznamu jsou již dříve body více se blížící lokálnímu minimu. Tento problém lze částečně vyřešit vhodnou volbou parametrů minimalizace. Je rovněž patrné, že se v metodě přibližujeme k lokálnímu minimu mnohem pomaleji než v metodě předchozí. (Číselné hodnoty jsou při vypisování v textu vždy zaokrouhlovány na pět desetinných míst.)

Na závěr demonstrujme obě metody minimalizace u složitější funkce, která má více lokálních extrémů. Zkoumejme reálnou funkci dvou proměnných s předpisem $f(x,y) = 0{,}1x^4 - 0{,}5x^2 - 0{,}3x + 0{,}05y^4 + 0{,}2y^2 - 0{,}2y - 0{,}4xy + 5$. Výsledky výpočtů lze získat spuštěním programu vytvořeného ve výpočetním prostředí MATLAB.

V obecné metodě největšího spádu je zvolen počáteční bod iterace $x_0 = (0{,}5; -2{,}5)$. Pro tento bod nalezneme jedno z možných lokálních minim, protože jak víme, minimalizace je na této prvotní volbě závislá. Podmínka pro ukončení iteračního procesu je volena $\varepsilon = 0{,}0001$. Na obrázku 4.3 je znázorněn graf této funkce a minimalizující posloupnost bodů. Odhad lokálního minima pro tuto volbu parametru ε vychází hodnota $3{,}16093$ v bodě $(1{,}89262; 1{,}29822)$. Tento výsledek získáme již po 9 iteracích. V pohledu shora je zvýrazněna ortogonalita směrů v bodech minimalizující posloupnosti a jsou vypsány souřadnice prvních dvou směrů. Body blízko minima se téměř překrývají, proto je nepopisujeme a nezobrazujeme ani další směry.

V modifikované metodě největšího spádu vycházíme ze stejného bodu, tedy $x_0 = (0{,}5; -2{,}5)$. Délku kroku λ_k volíme konstantní a to $\lambda_k = 0{,}2$. Podmínka pro ukončení iterace je rovněž $\varepsilon = 0{,}0001$. Odhad lokálního minima pro tuto volbu parametrů λ_k a ε vychází hodnota $3{,}161$ v bodě $(1{,}88686; 1{,}29164)$. Na obrázku 4.4 je znázorněn opět graf této funkce s minimalizující posloupností bodů a připojen je rovněž pohled shora.

Výstupy jsou získány z výpočetního prostředí MATLAB. Programy řeší modifikovanou a obecnou metodu největšího spádu, přičemž výpočty derivací, parciálních derivací a gradientů v minimalizačních procesech provádíme numericky.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

`nejvetsi_spad_modifik.m` (programy/minimalizace),
`nejvetsi_spad.m` (programy/minimalizace).

Předpis reálné funkce dvou proměnných a počáteční bod iterace

$$f(x,y) = 0{,}1x^2 + 0{,}09y^2$$
$$x_0 = (8,6),\ f(x_0) = 9{,}64$$

$$\Rightarrow \quad \nabla f = (0{,}2x; 0{,}18y)^{\mathrm{T}}$$
$$-\nabla f(x_0) = (-1{,}6; -1{,}08)^{\mathrm{T}}$$

Jednorozměrná minimalizační úloha

$$f(x_0 - \lambda \nabla f(x_0)) = 0{,}1(8 - 1{,}6\lambda)^2 + 0{,}09(6 - 1{,}08\lambda)^2 =$$
$$= 0{,}36098\lambda^2 - 3{,}7264\lambda + 9{,}64 \longrightarrow \min_{\lambda > 0}$$

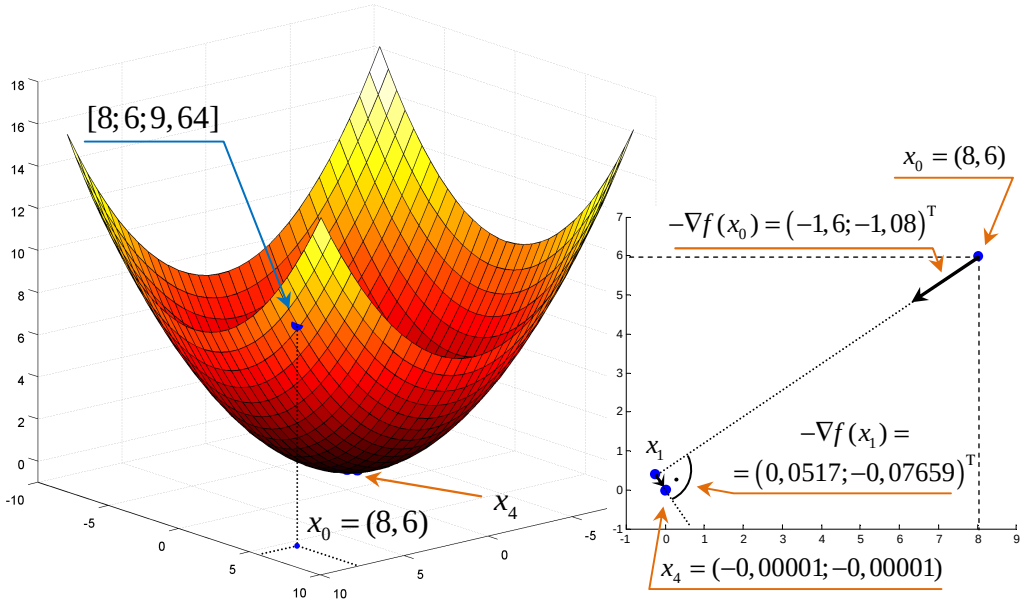
$\lambda_0 = 5{,}16156$ $\Rightarrow$ $x_1 = x_0 - \lambda_0 \nabla f(x_0) = (-0{,}2585; 0{,}42551)$

další bod minimalizující posloupnosti

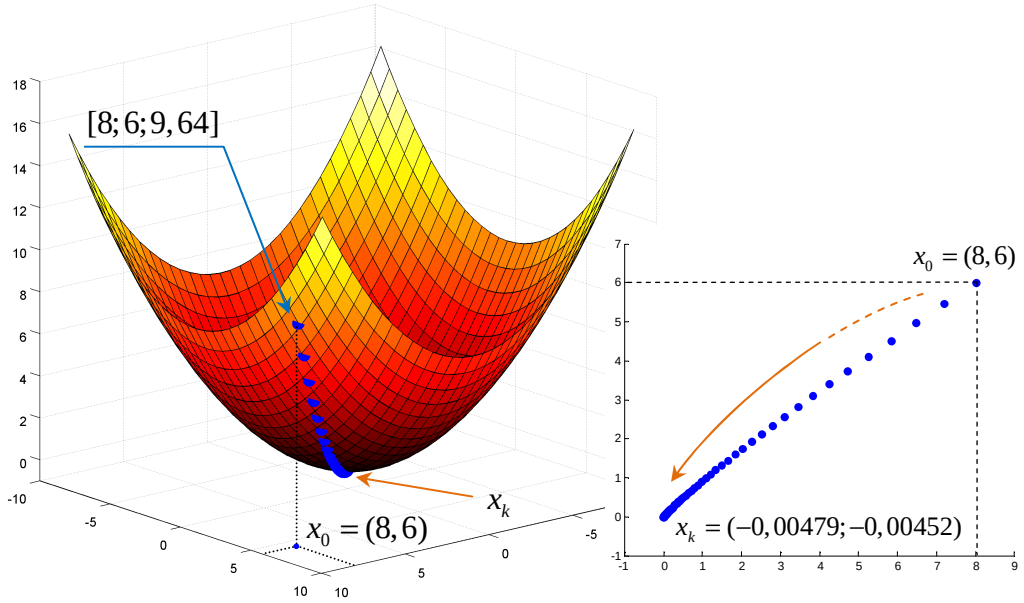
$-\nabla f(x_1) = (0{,}0517; -0{,}07659)^{\mathrm{T}}$ další směr

Ověření ortogonality směrů $(\nabla f(x_0)) \cdot (\nabla f(x_1)) = 0$

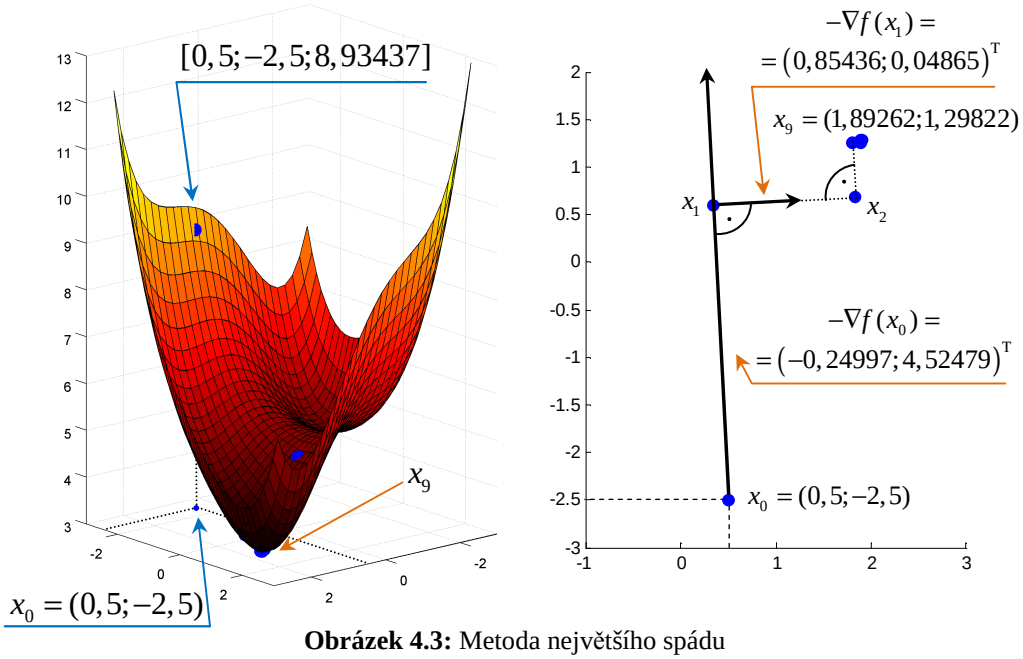
$x_4 = (-0{,}00001; -0{,}00001)$, $f(x_4) = 0$ odhad lokálního minima



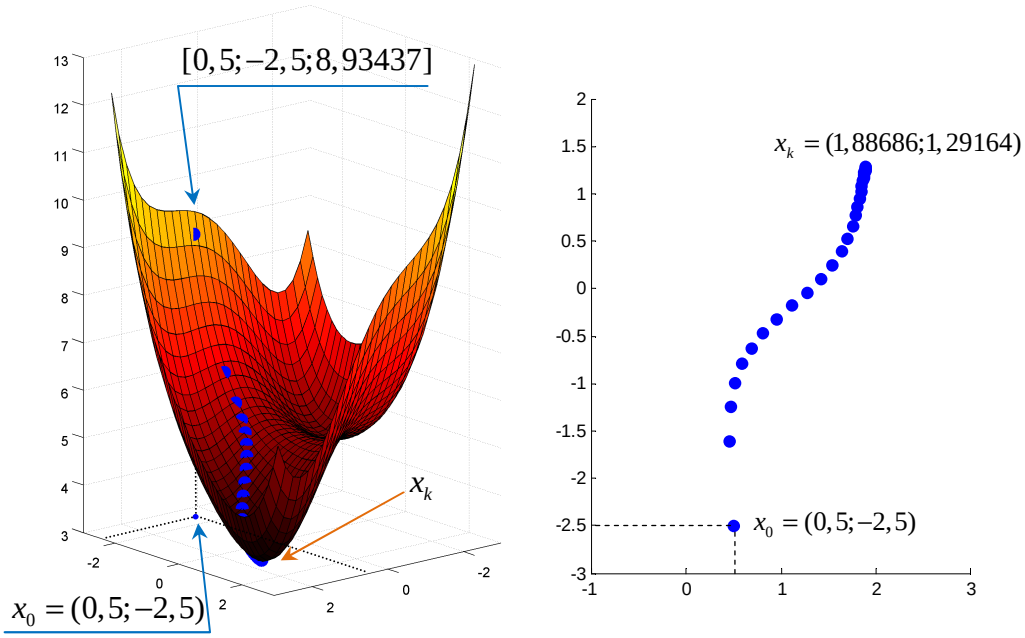
Obrázek 4.1: Metoda největšího spádu



Obrázek 4.2: Modifikovaná metoda největšího spádu s konstantní délkou kroku



Obrázek 4.3: Metoda největšího spádu



Obrázek 4.4: Modifikovaná metoda největšího spádu s konstantní délkou kroku

---

V praktických aplikacích nemáme k dispozici explicitní předpis reálné funkce f více proměnných. Parciální derivace a gradient funkce f proto určujeme na základě numerických metod. Ukážeme si to až dále při použití minimalizačních metod v nově navržených postupech u konkrétních příkladů.

4.2 Hledání osy rotační válcové plochy

V tomto oddíle představme nově navrženou metodu hledání os symetrií speciálních bodových množin v rovině a v prostoru. V rovině se bude jednat o body, které leží na rovnoběžných přímkách, v prostoru potom o body ležící na povrchu rotační válcové plochy. V obou případech připouštíme, že body nejsou navzorkovány pravidelně, dokonce mohou některé části bodových množin chybět.

K řešení těchto speciálních případů nás přivedl problém, že často pracujeme s neúplnými bodovými množinami nebo s množinami, které reprezentují pouze část nějaké elementární plochy. Bodové množiny, které odpovídají reálným objektům nebo jejich zmenšeným modelům např. klenbám, jsou většinou jen části nějakých elementárních ploch. Například křížové a valené klenby, které analyzujeme, jsou velice často složeny z částí rotačních válcových ploch. Hledáme-li tedy osy těchto částí ploch, nelze použít metodu ortogonálního prokládání přímkou nebo metodu PCA, neboť nemůžeme využít toho, že množina se všemi body obsahuje také body, které jsou s nimi souměrné nebo alespoň přibližně souměrné podle osy symetrie. Proto jsme vyvinuli metodu, která řeší nalezení osy symetrie bodů na rovnoběžných přímkách (zajímá nás pouze jedna osa symetrie, jak bude vidět z příkladů) a bodů na rotačních válcových plochách, přičemž se může jednat o neúplné nebo nepravidelné bodové množiny. Pro jednoduchost začínáme právě s analýzou těchto speciálních bodových množin, později naše metody zobecníme, abychom byli schopni zpracovat též bodové množiny reprezentující obecné rotační plochy.

Princip nově navržené metody vysvětlíme pro případ bodové množiny reprezentující rotační válcovou plochu. Pro body ležící na rovnoběžných přímkách bude postup zcela analogický a ukážeme jej na příkladě.

Předpokládejme, že máme k dispozici neprázdnou množinu $\mathcal{X}$ n bodů v eukleidovském prostoru E_3 získané navzorkováním rotační válcové plochy, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$ (indexujeme nyní od jedné). Jak již bylo řečeno, body mohou být navzorkovány pravidelně, nepravidelně nebo mohou některé části bodové množiny dokonce chybět. Z definice osy rotační válcové plochy víme, že se jedná o přímku, která má od každého bodu plochy stejnou vzdálenost rovnající se poloměru této rotační válcové plochy. Hledání osy rotační válcové plochy řešíme iteračně s použitím odvozených minimalizačních postupů z předchozího oddílu. V prvním kroku algoritmu zvolíme libovolnou počáteční polohu přímky, kterou v dalších krocích optimalizujeme, dokud se přímka dostatečně nepřiblíží ose rotační válcové plochy. Definujeme proto *náhodnou veličinu*, která představuje vzdálenost bodu rotační válcové plochy od její osy, tedy poloměr této plochy. K dispozici ovšem máme pouze pozorování náhodné veličiny v každém kroku iteračního algoritmu, tedy vektor, jehož složky jsou vzdálenosti všech bodů rotační válcové plochy od aktuální přímky (v prvním kroku od zvolené přímky) a představují náhodný výběr velikosti n. Jedná se tedy o vektor reálných čísel. Dále zavádíme tzv. *chybovou funkci*, kterou definujeme jako rozptyl náhodné veličiny. Hledáme minimum chybové funkce, neboť víme, že rozptyl náhodné veli-

činy musí být v ideálním případě roven nule, tj. v takovém případě, kdy je aktuální přímka rovna ose rotační válcové plochy. Minimalizačním procesem tak získáme odhad této osy. Minimalizaci chybové funkce provádíme pomocí odvozené diferenciální numerické metody největšího spádu a to jak její obecnou, tak i modifikovanou verzí s konstantní délkou kroku.

Chybová funkce má celkem šest parametrů (pracujeme v eukleidovském prostoru E_3) – souřadnice určujícího bodu aktuální přímky a souřadnice jejího směrového vektoru, které v každém kroku algoritmu aktualizujeme. Jedná se tedy o reálnou funkci šesti proměnných.

Pojmy náhodné veličiny a rozptylu jsme již definovaly v oddíle o PCA, připomeňme pouze definice odhadu parametrů polohy a variability. Z náhodného výběru velikosti n, který je popsán vektorem $x = (x_1, x_2, ..., x_n)^{\mathrm{T}}$, určíme *odhad střední hodnoty (výběrový průměr)* vztahem

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.7}$$

a *odhad rozptylu (výběrový rozptyl)* vztahem

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x})^2 . \tag{4.8}$$

Definice těchto pojmů a další vlastnosti je možné nalézt například v (Meloun a Militký, 2004; Zvára a Štěpán, 2001).

Výpočet chybové funkce je předložen v následujícím symbolickém kódu jako algoritmus 4.1. Chybová funkce je v pseudokódu označena jako *error_function* a její hodnota pro konkrétních šest parametrů jako *error*.

Algoritmus 4.1: *VÝPOČET CHYBOVÉ FUNKCE ERROR_FUNCTION PRO ROTAČNÍ VÁLCOVOU PLOCHU*. Vstupními parametry chybové funkce jsou určující bod A a směrový vektor u aktuální přímky, výstupem je reálná hodnota *error*.

Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3 (indexujeme nyní od jedné), které leží na rotační válcové ploše. Symbolický kód využívá několik proměnných. Složky vektoru *distance* představují náhodný výběr velikosti *n*, tj. vzdálenosti bodů vstupní množiny od aktuální přímky. Vzdálenost *i*-tého bodu od aktuální přímky počítáme jako velikost vektoru p_i, tj.

$$\|p_i\| = \left\| -(X_i - A) + \frac{(X_i - A)\cdot u}{\|u\|^2} u \right\| .$$

Proměnná *average* je odhadem střední hodnoty, kterou určujeme podle vzorce (4.7). Do proměnné *error* ukládáme výstupní hodnotu chybové funkce *error_function* pro vstupní parametry A a u a to odhad rozptylu podle vzorce (4.8).

---

function *error_function*(A,u): **real**

1: $distance \leftarrow \emptyset$
2: **for** $i = 1, 2, ..., n$ **do**
3: $\quad distance[i] \leftarrow \|p_i\|$
4: $average \leftarrow \frac{1}{n}\sum_{i=1}^{n} distance[i]$
5: **for** $i = 1, 2, ..., n$ **do**
6: $\quad d[i] \leftarrow (distance[i] - average)^2$
7: $error \leftarrow \frac{1}{n-1}\sum_{i=1}^{n} d[i]$
8: **return** *error*

Zapišme dále pomocí pseudokódu proces hledání osy rotační válcové plochy minimalizací chybové funkce. V algoritmu 4.2 nejdříve začněme s jednodušší variantou minimalizace a to s modifikovanou metodou největšího spádu s konstantní délkou kroku. V algoritmu 4.4 je popsána obecnější metoda největšího spádu s určováním délky kroku pomocí jednorozměrné minimalizace. Algoritmy se tedy liší pouze v určování délky kroku. Podrobně je rozepisujeme kvůli přehlednosti a ukázce jednorozměrné minimalizace pomocí Newtonovy metody tečen v algoritmu 4.3.

Algoritmus 4.2: *HLEDÁNÍ OSY ROTAČNÍ VÁLCOVÉ PLOCHY – VARIANTA A.* První varianta hledání osy rotační válcové plochy je založena na modifikované metodě největšího spádu a je symbolicky popsána jako funkce. Jejími vstupními parametry jsou určující bod A a směrový vektor u přímky, kterou volíme libovolně. Výstupy funkce jsou dvě minimalizující posloupnosti bodů a směrových vektorů – *sequence_points* a *sequence_vectors*. Poslední bod a poslední vektor těchto dvou posloupností jsou určující bod a směrový vektor přímky, která tvoří odhad hledané osy rotační válcové plochy.

Předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které leží na rotační válcové ploše. Matice vstupních bodů typu $(n \times 3)$, kde jeden řádek představuje jeden vstupní bod, je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus 4.2 využívá externí funkci *error_function* popsanou v předešlém pseudokódu jako algoritmus 4.1. Popišme další proměnné, které jsou použity v následujícím symbolickém kódu. Proměnné ΔA ($\Delta A\_x,y,z$), Δu ($\Delta u\_x,y,z$) slouží k výpočtu numerické parciální derivace, A_{STEP} a u_{STEP} jsou aktuální bod a směrový vektor v jednom kroku iterace. Proměnné ∂A_{STEP} a ∂u_{STEP} jsou numericky vypočítané parciální derivace funkce *error_function* podle bodu A a podle směrového vektoru u (myšleno po souřadnicích) v A_{STEP} a u_{STEP}, ∇ je gradient funkce *error_function* pro hodnoty A_{STEP} a u_{STEP}. Proměnné *new_A* a *new_u* jsou nový bod a vektor minimalizujících posloupností získané modifikovanou metodou největšího spádu v jednom kroku minimalizace. Iterační proces je ukončen, pokud je splněna jedna z podmínek.

V pseudokódu používáme ještě globální proměnné δ, ε, λ a *max_iteration*, které určují nastavení metody minimalizace. Vhodná je například volba

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,1$
max_iteration $= 100$.

---

function *search_axis - VariantA*(A,u): **pair of sequences**

1: *plot*$(line, A, u)$
%{vykreslení počáteční polohy přímky s určujícím bodem A a směrovým vektorem u }%
2: $\Delta A \leftarrow A$
3: $A_{STEP} \leftarrow A$
4: $\Delta u \leftarrow u$
5: $u_{STEP} \leftarrow u$
6: *iteration* $\leftarrow 0$
7: *sequence_points*[1] $\leftarrow A$
8: *sequence_vectors*[1] $\leftarrow u$
9: **while** *True* **do** %{ukončení cyklu řízeno uvnitř}
10: | *iteration* $\leftarrow$ *iteration* $+1$
11: | $\Delta A\_x \leftarrow \Delta A + (\delta, 0, 0)$
12: | $\Delta A\_y \leftarrow \Delta A + (0, \delta, 0)$
13: | $\Delta A\_z \leftarrow \Delta A + (0, 0, \delta)$
14: | $\Delta u\_x \leftarrow \Delta u + (\delta, 0, 0)$
15: | $\Delta u\_y \leftarrow \Delta u + (0, \delta, 0)$
16: | $\Delta u\_z \leftarrow \Delta u + (0, 0, \delta)$
17: | $\partial A_{STEP} \leftarrow \left(error\_function(\Delta A, u_{STEP}) - error\_function(A_{STEP}, u_{STEP})\right) / \delta$
| %{parciální derivace funkce *error_function* podle bodu A v A_{STEP} }%
18: | $\partial u_{STEP} \leftarrow \left(error\_function(A_{STEP}, \Delta u) - error\_function(A_{STEP}, u_{STEP})\right) / \delta$
| %{parciální derivace funkce *error_function* podle vektoru u v u_{STEP} }%
19: | $\nabla \leftarrow (\partial A_{STEP}, \partial u_{STEP})$
| %{gradient funkce *error_function* }%
20: | *accuracy* $\leftarrow$ *norm*(∇)
21: | **if** *accuracy* $\leq \varepsilon$ **or** *iteration* $\geq$ *max_iteration* **then**
| | %{podmínka ukončení iterace}%
22: | | **break**
23: | *new_A* $\leftarrow A_{STEP} - \lambda \cdot \nabla$[1:3]
24: | *new_u* $\leftarrow u_{STEP} - \lambda \cdot \nabla$[4:end]
25: | *sequence_points*[*iteration*] $\leftarrow$ *new_A*
26: | *sequence_vectors*[*iteration*] $\leftarrow$ *new_u*
27: | $\Delta A \leftarrow$ *new_A*
28: | $A_{STEP} \leftarrow$ *new_A*
29: | $\Delta u \leftarrow$ *new_u*
30: | $u_{STEP} \leftarrow$ *new_u*
31: | *plot*$(line, A_{STEP}, u_{STEP})$

| %{vykreslení aktuální polohy přímky s určujícím bodem A_{STEP} a směrovým vektorem u_{STEP}, animace minimalizační metody}%

32: $plot(line, A_{STEP}, u_{STEP})$
%{vykreslení odhadu polohy hledané osy s určujícím bodem A_{STEP} a směrovým vektorem u_{STEP}}%
33: $plot(X)$
%{vykreslení bodů vstupní množiny}%
34: **return** (*sequence_points, sequence_vectors*)

V algoritmu 4.3 je symbolicky zapsána jednorozměrná minimalizace pomocí Newtonovy metody tečen. Klasický postup výpočtu je zde kvůli praktické aplikaci obohacen o další parametry, které řídí kvalitní průběh metody. Jedná se o ukončení iterace v případě nepřesného výpočtu druhé derivace nebo v případě, kdy se bod minimalizující posloupnosti dostává mimo určený interval (řádek 13). Vhodné hodnoty těchto parametrů bylo nutné experimentálně vysledovat.

Algoritmus 4.3: *NEWTONOVA METODA TEČEN*. Jednorozměrná minimalizace reálné funkce je založena na Newtonově metodě tečen a implementována je jako funkce. Vstupními parametry funkce jsou počáteční bod minimalizace λ_0, který volíme, určující bod A_{STEP} a směrový vektor u_{STEP} aktuální přímky a gradient pro tyto hodnoty. Výstupem funkce je reálná hodnota *minimum*, která je odhadem bodu, v němž funkce nabývá lokálního minima.

Předpokládejme, že je opět dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které leží na rotační válcové ploše. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Funkce *Newton_minimization* využívá externí funkci $\lambda\_error\_function$, která slouží k výpočtu hodnoty chybové funkce *error_function* v bodě λ. V následujícím symbolickém kódu jsou použity proměnné $\Delta\lambda$, $\Delta\Delta\lambda$ a $\lambda\_error\_function_{1,2,3}$, které slouží k výpočtu numerické derivace. Proměnné $\partial\lambda_{STEP}$ a $\partial\Delta\lambda$ jsou numericky vypočítané derivace funkce $\lambda\_error\_function$ v λ_{STEP} a $\Delta\lambda$, ∇ je gradient funkce *error_function* pro hodnoty A_{STEP} a u_{STEP}. Proměnná $\partial^2\lambda_{STEP}$ je druhá derivace funkce $\lambda\_error\_function$ v λ_{STEP}. Proměnná *new_*λ představuje nový bod minimalizující posloupnosti získané Newtonovou metodou tečen. Iterační proces je ukončen, pokud je splněna jedna z podmínek.

V pseudokódu používáme ještě globální proměnné δ, ε, λ^{LB}, λ^{UB} a *max_iteration*, které určují nastavení metody minimalizace. Vhodná je například volba

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -200$
$\lambda^{UB} = 200$
$max\_iteration = 500$.

---

function *Newton_minimization*$(\lambda_0, A_{STEP}, u_{STEP}, \nabla)$: **real**

1: $\lambda_{STEP} \leftarrow \lambda_0$
2: $iteration \leftarrow 0$
3: **while** *True* **do** %{ukončení cyklu řízeno uvnitř}
4: $iteration \leftarrow iteration + 1$
5: $\Delta\lambda \leftarrow \lambda_{STEP} + \delta$
6: $\Delta\Delta\lambda \leftarrow \Delta\lambda + \delta$
7: $\lambda\_error\_function_1 = \lambda\_error\_function(\lambda_{STEP}, A_{STEP}, u_{STEP}, \nabla)$
8: $\lambda\_error\_function_2 = \lambda\_error\_function(\Delta\lambda, A_{STEP}, u_{STEP}, \nabla)$
9: $\lambda\_error\_function_3 = \lambda\_error\_function(\Delta\Delta\lambda, A_{STEP}, u_{STEP}, \nabla)$
10: $\partial\lambda_{STEP} = (\lambda\_error\_function_2 - \lambda\_error\_function_1) / \delta$
%{derivace funkce $\lambda\_error\_function$ v λ_{STEP} }%
11: $\partial\Delta\lambda = (\lambda\_error\_function_3 - \lambda\_error\_function_2) / \delta$
%{derivace funkce $\lambda\_error\_function$ v $\Delta\lambda$ }%
12: $\partial^2\lambda_{STEP} = (\partial\Delta\lambda - \partial\lambda_{STEP}) / \delta$
%{druhá derivace funkce $\lambda\_error\_function$ v λ_{STEP} }%
13: **if** $\partial^2\lambda_{STEP} \le \varepsilon$ **then**
%{podmínka ukončení iterace, nepřesný výpočet druhé derivace}%
14: **return** λ_{STEP}
15: $new\_\lambda \leftarrow \lambda_{STEP} - (\partial\lambda_{STEP} / \partial^2\lambda_{STEP})$
16: **if** $new\_\lambda \le \lambda^{LB}$ **or** $new\_\lambda \ge \lambda^{UB}$ **then**
%{únik hodnoty new_λ z vymezeného intervalu}%
17: **return** λ_{STEP}
18: $accuracy \leftarrow abs(new\_\lambda - \lambda_{STEP})$
19: **if** $accuracy \le \varepsilon$ **or** $iteration \ge max\_iteration$ **then**
%{podmínka ukončení iterace}%
20: **break**
21: $\lambda_{STEP} \leftarrow new\_\lambda$
22: $minimum \leftarrow new\_\lambda$
23: **return** *minimum*

function λ*_error_function*$(\lambda, A_{STEP}, u_{STEP}, \nabla)$: **real**

1: $distance \leftarrow \emptyset$
2: $A \leftarrow A_{STEP} - \lambda \cdot \nabla[1{:}3]$
3: $u \leftarrow u_{STEP} - \lambda \cdot \nabla[4{:}\text{end}]$
4: **for** $i = 1, 2, ..., n$ **do**
5: $distance[i] \leftarrow \|p_i\|$
6: $average \leftarrow \frac{1}{n}\sum_{i=1}^{n} distance[i]$
7: **for** $i = 1, 2, ..., n$ **do**
8: $d[i] \leftarrow (distance[i] - average)^2$
9: $error_\lambda \leftarrow \frac{1}{n-1}\sum_{i=1}^{n} d[i]$
10: **return** $error_\lambda$

V algoritmu 4.4 je předložen symbolický kód hledání osy rotační válcové plochy pomocí obecné metody největšího spádu. Ukázalo se, že použití této metody v praktických aplikacích je mnohem komplikovanější než hledání řešení pomocí algoritmu 4.2 založeného na modifikované metodě největšího spádu. V jednorozměrné minimalizaci, pomocí níž určujeme vhodnou délku kroku v metodě největšího spádu, je stěžejní určení správného počátečního bodu λ_0. Bylo nutné ošetřit případy, kdy má jednorozměrná funkce více lokálních minim. Neznáme ovšem analytický předpis této funkce, předem tedy neznáme ani počet těchto lokálních extrémů. Algoritmus 4.4 jsme proto obohatili o hledání vhodného počátečního bodu λ_0 pro jednorozměrnou minimalizaci v každém kroku metody největšího spádu. Tato část algoritmu 4.4 je v pseudokódu zvýrazněna, jedná se o řádky 23 až 32. Newtonovu metodu tečen tedy začínáme řešit pro několik náhodně generovaných počátečních bodů a porovnáním výsledných hodnot určíme nejlepší odhad hledaného lokálního minima. Počet náhodně generovaných bodů i interval jejich hodnot je opět nutné zjišťovat experimentálně.

Algoritmus 4.4: *HLEDÁNÍ OSY ROTAČNÍ VÁLCOVÉ PLOCHY – VARANTA B.* Druhá varianta hledání osy rotační válcové plochy je založena na obecné metodě největšího spádu a je symbolicky popsána jako funkce. Jejími vstupními parametry jsou určující bod A a směrový vektor u přímky, kterou volíme libovolně. Výstupy funkce jsou dvě minimalizující posloupnosti bodů a směrových vektorů – *sequence_points* a *sequence_vectors*. Poslední bod a poslední vektor těchto dvou posloupností jsou určující bod a směrový vektor přímky, která tvoří odhad hledané osy rotační válcové plochy. Oproti algoritmu 4.2 je tento algoritmus obohacen o jednorozměrnou minimalizaci v každém kroku iteračního procesu.

Předpokládejme, že je opět dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které leží na rotační válcové ploše. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus 4.4 využívá externí funkci *error_function* popsanou v algoritmu 4.1 a externí funkci *Newton_minimization* popsanou v algoritmu 4.3. V následujícím symbolickém kódu jsou použity stejné proměnné jako v algoritmu 4.2 a dále proměnné, které jsou nutné pro výpočet jednorozměrné minimalizace. Výpočet jednorozměrné minimalizace je v každém kroku metody největšího spádu prováděn vícekrát pro různé náhodně volené počáteční body λ_0 z předem daného intervalu. Porovnáním výsledků jednorozměrné minimalizace pro jednotlivé počáteční body určíme nejlepší odhad hledaného lokálního minima funkce jedné proměnné. Pro porovnávání slouží proměnné λ_{LOW} a λ_{LOW}*_error_function*. Proměnná λ_{MIN} je potom odhadem bodu, v němž jednorozměrná funkce nabývá lokálního minima. Hodnota tohoto lokálního minima je ukládána do proměnné λ_{MIN}*_error_function*. Proměnné *new_A* a *new_u* jsou nový bod a vektor minimalizujících posloupností získané obecnou metodou největšího spádu. Iterační proces je ukončen, pokud je splněna jedna z podmínek.

Podotkněme, že obohacením algoritmu o část, ve které určujeme vhodný počáteční bod pro jednorozměrnou minimalizaci, jsme zavedli další parametry, pomocí nichž můžeme řídit kvalitu a průběh jednorozměrné minimalizace i celkově metody největšího spádu.

V pseudokódu používáme ještě globální proměnné $\delta, \varepsilon, a, b, c$ a *max_iteration* , které určují nastavení metody minimalizace. Vhodná je volba

$\delta = 0,001$

$\varepsilon = 1,0 \cdot 10^{-10}$

$a = 200$, $b = 100$, $c = 20$

max_iteration $= 100$.

---

function *search_axis - VariantB*(A,u) : **pair of sequences**

1: *plot*$(line, A, u)$
%{vykreslení počáteční polohy přímky s určujícím bodem A a směrovým vektorem u }%
2: $\Delta A \leftarrow A$
3: $A_{STEP} \leftarrow A$
4: $\Delta u \leftarrow u$
5: $u_{STEP} \leftarrow u$
6: *iteration* $\leftarrow 0$
7: *sequence_points*[1] $\leftarrow A$
8: *sequence_vectors*[1] $\leftarrow u$
9: **while** *True* **do** %{ukončení cyklu řízeno uvnitř}
10: | *iteration* $\leftarrow$ *iteration* $+1$
11: | $\Delta A\_x \leftarrow \Delta A + (\delta, 0, 0)$
12: | $\Delta A\_y \leftarrow \Delta A + (0, \delta, 0)$
13: | $\Delta A\_z \leftarrow \Delta A + (0, 0, \delta)$
14: | $\Delta u\_x \leftarrow \Delta u + (\delta, 0, 0)$
15: | $\Delta u\_y \leftarrow \Delta u + (0, \delta, 0)$
16: | $\Delta u\_z \leftarrow \Delta u + (0, 0, \delta)$
17: | $\partial A_{STEP} \leftarrow \left(error\_function(\Delta A, u_{STEP}) - error\_function(A_{STEP}, u_{STEP})\right) / \delta$
| %{parciální derivace funkce *error_function* podle bodu A v A_{STEP} }%
18: | $\partial u_{STEP} \leftarrow \left(error\_function(A_{STEP}, \Delta u) - error\_function(A_{STEP}, u_{STEP})\right) / \delta$
| %{parciální derivace funkce *error_function* podle vektoru u v u_{STEP} }%
19: | $\nabla \leftarrow (\partial A_{STEP}, \partial u_{STEP})$
| %{gradient funkce *error_function* }%
20: | *accuracy* $\leftarrow$ *norm*(∇)
21: | **if** *accuracy* $\le \varepsilon$ **or** *iteration* $\ge$ *max_iteration* **then**
| | %{podmínka ukončení iterace}%
22: | | **break**
23: | $\lambda_0 \leftarrow a \cdot rand - b$
| %{náhodné číslo z intervalu $(-b, a-b)$ }%
24: | $\lambda_{MIN} \leftarrow Newton\_minimization(\lambda_0, A_{STEP}, u_{STEP}, \nabla)$
25: | $\lambda_{MIN}\_error\_function \leftarrow \lambda\_error\_function(\lambda_{MIN}, A_{STEP}, u_{STEP}, \nabla)$
26: | **for** $i = 1, 2, ..., c$ **do**
27: | | $\lambda_0 \leftarrow a \cdot rand - b$
28: | | $\lambda_{LOW} \leftarrow Newton\_minimization(\lambda_0, A_{STEP}, u_{STEP}, \nabla)$
29: | | $\lambda_{LOW}\_error\_function \leftarrow \lambda\_error\_function(\lambda_{LOW}, A_{STEP}, u_{STEP}, \nabla)$
30: | | **if** $\lambda_{LOW}\_error\_function < \lambda_{MIN}\_error\_function$ **then**

31: $\lambda_{MIN} \leftarrow \lambda_{LOW}$
32: $\lambda_{MIN}\_error\_function \leftarrow \lambda_{LOW}\_error\_function$
33: $new\_A \leftarrow A_{STEP} - \lambda_{MIN} \cdot \nabla[1{:}3]$
34: $new\_u \leftarrow u_{STEP} - \lambda_{MIN} \cdot \nabla[4{:}end]$
35: $sequence\_points[iteration] \leftarrow new\_A$
36: $sequence\_vectors[iteration] \leftarrow new\_u$
37: $\Delta A \leftarrow new\_A$
38: $A_{STEP} \leftarrow new\_A$
39: $\Delta u \leftarrow new\_u$
40: $u_{STEP} \leftarrow new\_u$
41: $plot(line, A_{STEP}, u_{STEP})$
%{vykreslení aktuální polohy přímky s určujícím bodem A_{STEP} a směrovým vektorem u_{STEP}, animace minimalizační metody}%
42: $plot(line, A_{STEP}, u_{STEP})$
%{vykreslení odhadu polohy hledané osy s určujícím bodem A_{STEP} a směrovým vektorem u_{STEP}}%
43: $plot(X)$
%{vykreslení bodů vstupní množiny}%
44: **return** $(sequence\_points, sequence\_vectors)$

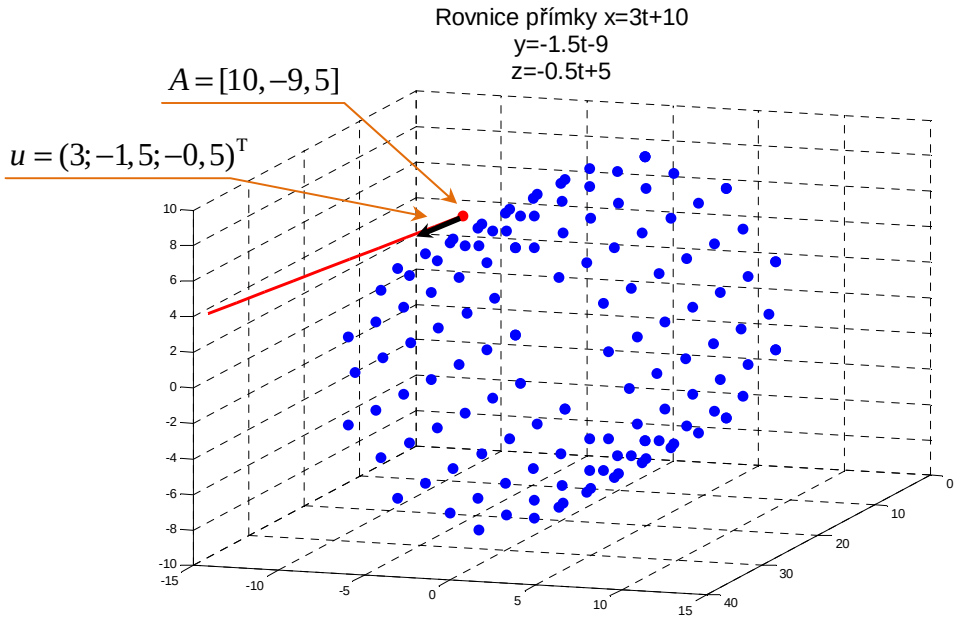
Funkčnost popsaných algoritmů demonstrujme na příkladech několika počítačově generovaných vstupních množin bodů. Bodové množiny odpovídající reálným povrchům budou zpracovány později.

Příklad 4.2: *HLEDÁNÍ OSY ROTAČNÍ VÁLCOVÉ PLOCHY.* Navržené algoritmy předvedeme na příkladech experimentálních bodových množin. Vždy předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které leží na povrchu rotační válcové plochy.

V prvním případě se jedná o body, které jsou na části rotační válcové plochy rozloženy pravidelně, jak vidíme na obrázku 4.5. Navíc je rotační válcová plocha, ze které byly body navzorkovány, volena ve speciální poloze (její osa je souřadnicová osa x), abychom mohli lépe kontrolovat správnost našich výsledků. Na obrázku 4.5 je rovněž zakreslena počáteční poloha přímky, která se postupně optimalizuje, a předloženy jsou její parametrické rovnice.

Osu rotační válcové plochy hledáme nejprve modifikovanou metodou největšího spádu. Obrázek 4.7 ilustruje postupné vylepšování polohy přímky, přičemž kvůli přehlednosti se přímka vykresluje pouze v několika pozicích. Zvlášť je vykreslena výsledná poloha osy a vypsány jsou její parametrické rovnice, viz obrázek 4.8. Dostatečně dobrý odhad osy rotační válcové plochy dostáváme pro uvedenou volbu vstupních parametrů metody (1000 iterací). Pokud bychom vyžadovali přesnější výsledek, je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroku λ.

Ta samá vstupní množina bodů je také zpracována obecnou metodou největšího spádu. Počáteční polohu přímky volíme stejně jako v metodě předchozí, viz obrázek 4.5. Postupné vylepšování polohy přímky znázorňuje obrázek 4.9. Opět přímku



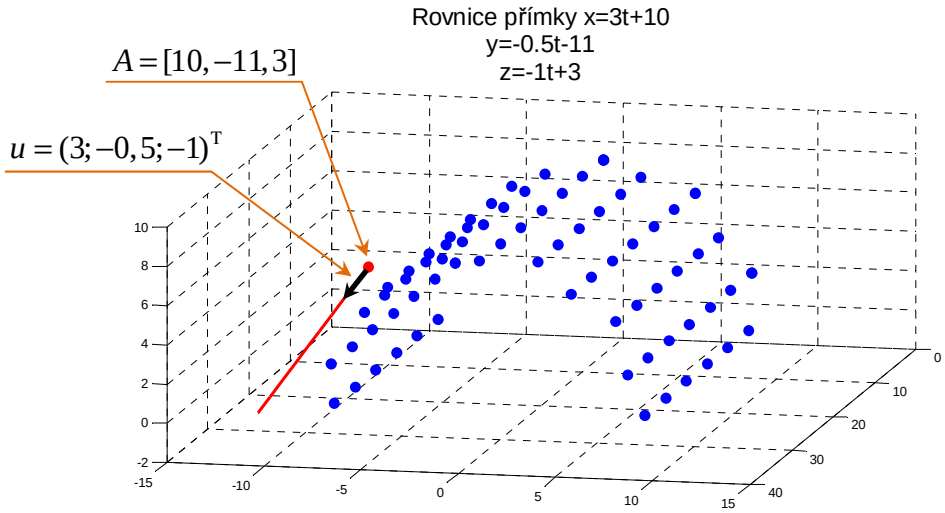
Obrázek 4.5: Body pravidelně rozmístěné na části rotační válcové plochy a počáteční poloha přímky

vykreslujeme jen v několika pozicích. Vypisujeme také vstupní parametry, které určují nastavení minimalizace metodou největšího spádu a parametry pro jednorozměrnou minimalizaci Newtonovou metodou tečen. Vhodnou volbu parametrů je třeba experimentálně vypozorovat. Z toho důvodu je tato metoda mnohem komplikovanější, obzvláště pro rozsáhlé množiny. Již po 100 iteracích získáváme dostatečně přesný odhad hledané osy rotační válcové plochy. Výsledná přímka je opět pro přehlednost vykreslena zvlášť a vypsány jsou její parametrické rovnice, viz obrázek 4.10. Vyžadujeme-li přesnější výsledek, lze zvýšit počet iterací nebo změnit vstupní parametry metody.

Pro obě metody minimalizace chybové funkce jsou vykresleny také závislosti hodnoty chybové funkce na počtu iterací, viz obrázky 4.11 a 4.12. V obou případech jsou hodnoty vypsány pro prvních sto kroků algoritmu. Z grafů konvergence hodnoty chybové funkce k hodnotě minimální je zřejmé, že obecná metoda největšího spádu konverguje výrazně rychleji. Modifikovaná metoda je na druhou stranu jednodušší a lze lépe kontrolovat její průběh. V obecné metodě jsou mezi jednotlivými polohami optimalizované přímky velké rozdíly ve směrech. To je dáno ortogonalitou směrů posunutí v bodech minimalizující posloupnosti.

Ve druhém případě, který předvedeme, se jedná o body, jenž jsou na povrchu části rotační válcové plochy rozloženy pravidelně, ovšem množina obsahuje pouze body jednoho poloprostoru určeného rovinou procházející osou, jak vidíme na obrázku 4.6. To znamená, že bodová množina neobsahuje body, které jsou s danými body souměrné podle osy rotační válcové plochy. Opět je rotační válcová plocha, ze které jsou body navzorkovány, volena ve speciální poloze (její osa je souřadnicová osa x).

Na obrázku 4.6 je rovněž zakreslena počáteční poloha přímky, která se postupně optimalizuje, a předloženy jsou její parametrické rovnice.

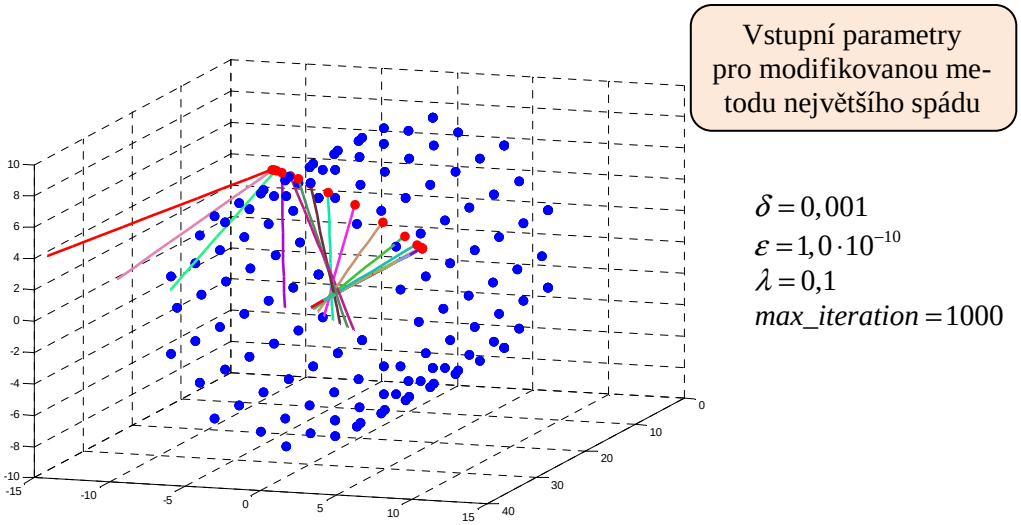


Obrázek 4.6: Body pravidelně rozmístěné na části rotační válcové plochy a počáteční poloha přímky

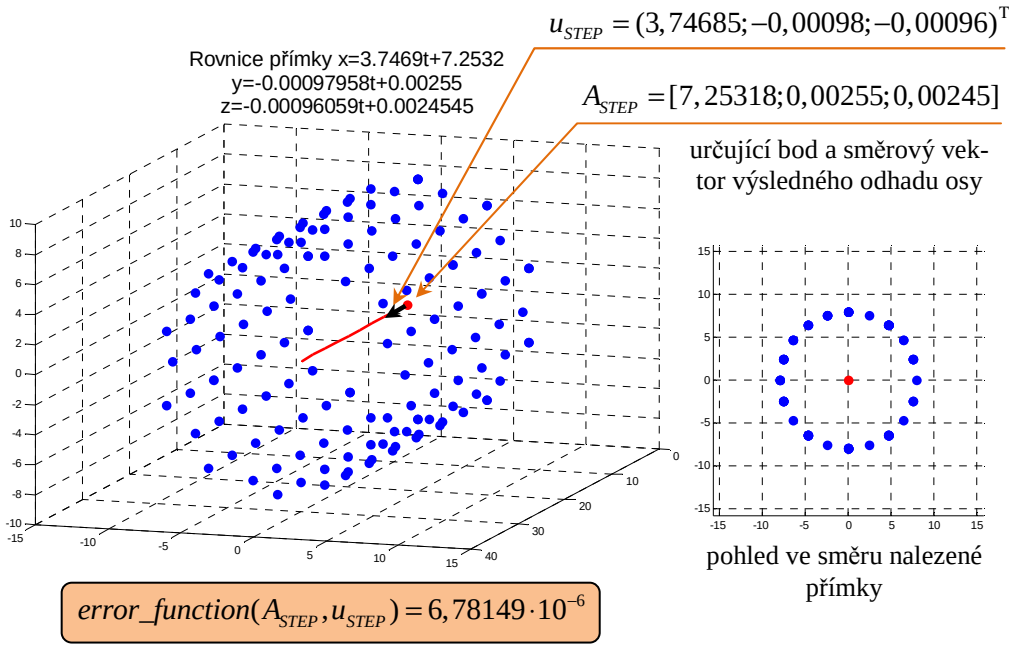
Osu rotační válcové plochy hledáme opět oběma metodami vícerozměrné minimalizace. Obrázky 4.13-4.16 sledují postupný výpočet odhadu osy. Volby vstupních parametrů minimalizačních technik jsou opět předloženy. Na závěr jsou pro obě metody minimalizace chybové funkce prezentovány také závislosti hodnoty chybové funkce na počtu iterací, viz obrázek 4.17.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

`osa_rot_valce_spad_modifik.m` (programy/osa_rot_valce),
`osa_rot_valce_spad.m` (programy/osa_rot_valce).



Obrázek 4.7: Vstupní parametry minimalizační metody a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



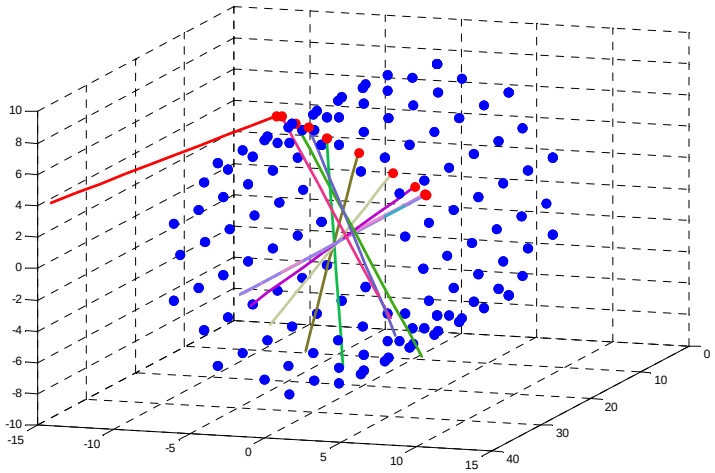
Obrázek 4.8: Výsledný odhad osy a pohled ve směru této osy – modifikovaná metoda největšího spádu

Vstupní parametry pro metodu největšího spádu

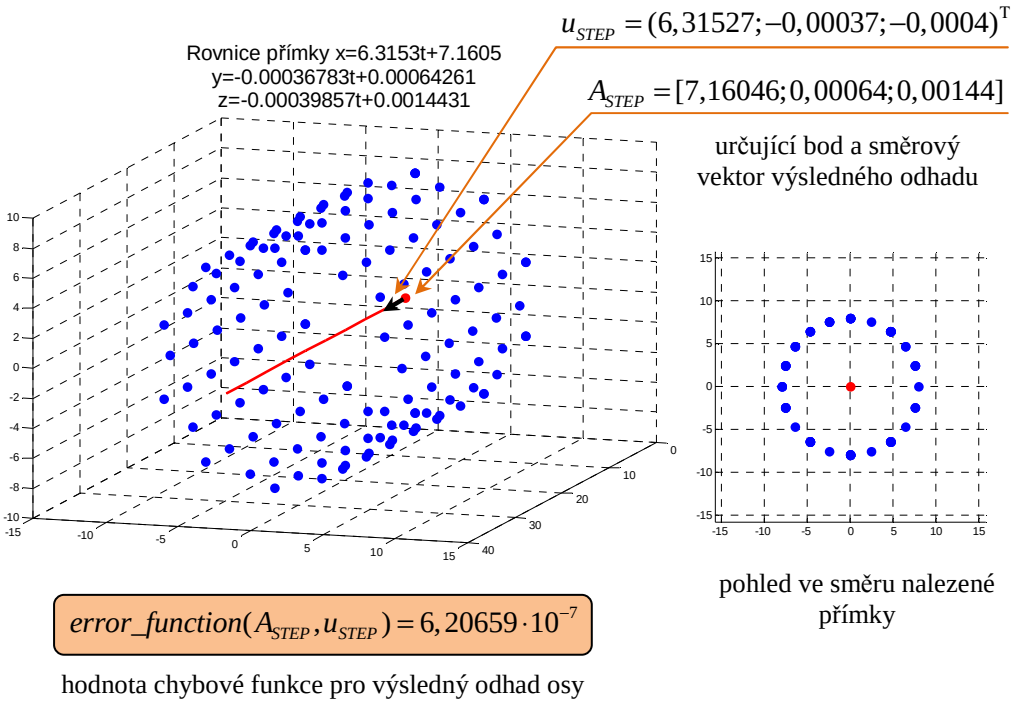
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 20$
$b = 10$
$c = 100$
$max\_iteration = 100$

Vstupní parametry pro Newtonovu metodu tečen

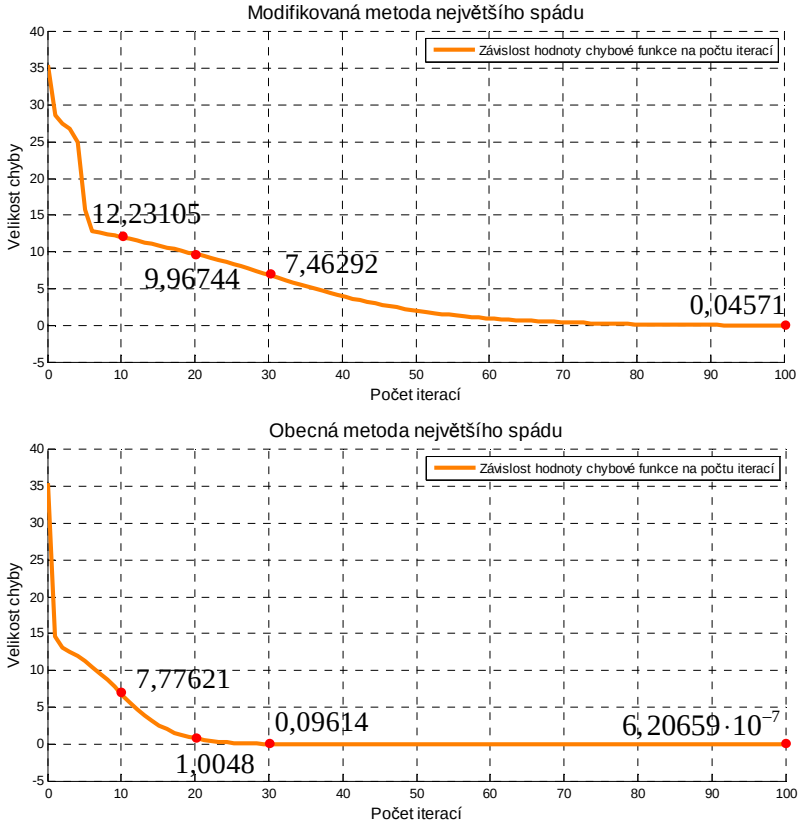
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -200$
$\lambda^{UB} = 200$
$max\_iteration = 500$



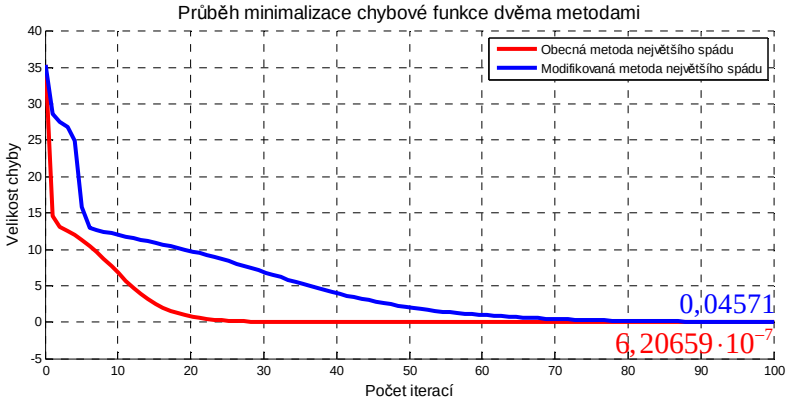
Obrázek 4.9: Vstupní parametry minimalizační metody a postupná optimalizace přímky minimalizací chybové funkce obecnou metodou největšího spádu



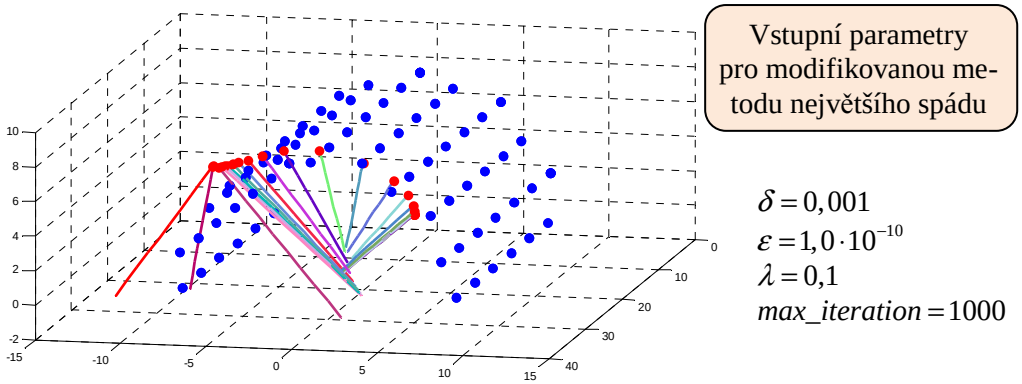
Obrázek 4.10: Výsledný odhad osy a pohled ve směru této osy – obecná metoda největšího spádu



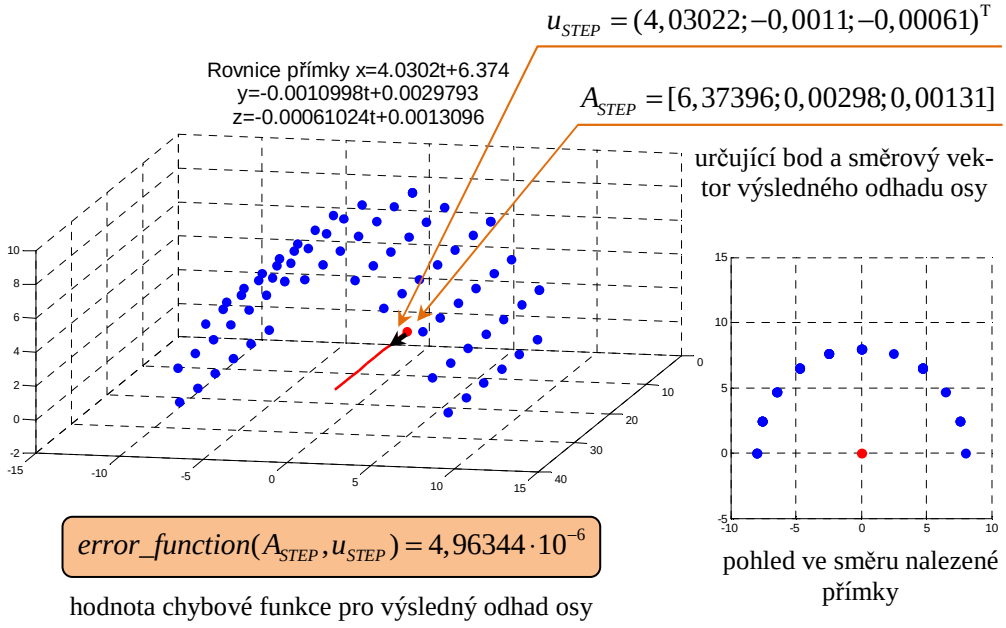
Obrázek 4.11: Minimalizace chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací



Obrázek 4.12: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací



Obrázek 4.13: Vstupní parametry minimalizační metody a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



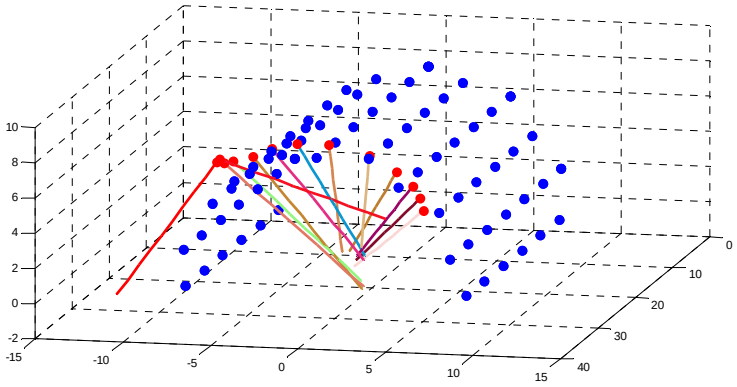
Obrázek 4.14: Výsledný odhad osy a pohled ve směru této osy – modifikovaná metoda největšího spádu

Vstupní parametry pro metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 11$
$b = 1$
$c = 100$
$max\_iteration = 100$

Vstupní parametry pro Newtonovu metodu tečen

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -10$
$\lambda^{UB} = 10$
$max\_iteration = 500$



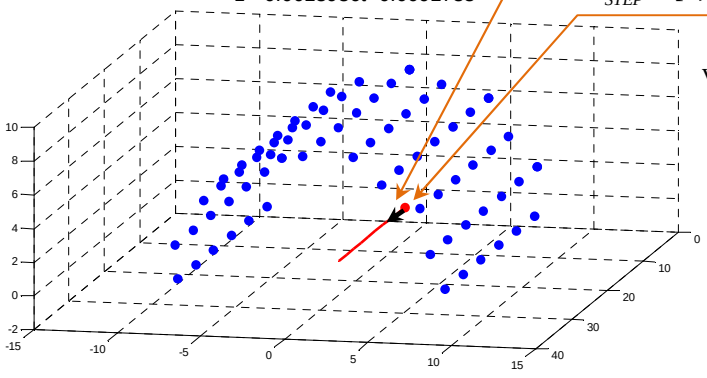
Obrázek 4.15: Vstupní parametry minimalizační metody a postupná optimalizace přímky minimalizací chybové funkce obecnou metodou největšího spádu

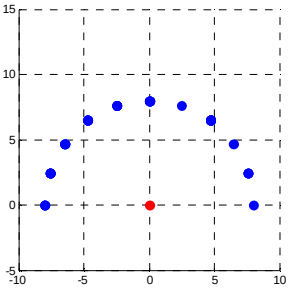
$u_{STEP} = (6,14568; -0,00092; -0,0019)^{T}$

Rovnice přímky x=6.1457t+6.2302
y=-0.00091967t+0.0015606
z=-0.0018986t+0.0091733

$A_{STEP} = [6,23016; 0,00156; 0,00917]$

určující bod a směrový vektor výsledného odhadu



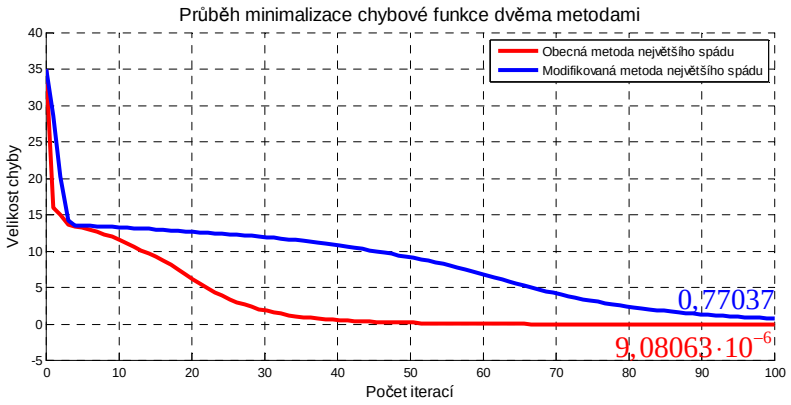


$error\_function(A_{STEP}, u_{STEP}) = 9,08063 \cdot 10^{-6}$

hodnota chybové funkce pro výsledný odhad osy

pohled ve směru nalezené přímky

Obrázek 4.16: Výsledný odhad osy a pohled ve směru této osy – obecná metoda největšího spádu

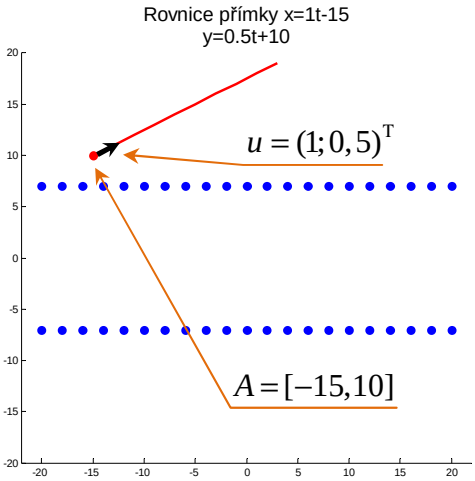


Obrázek 4.17: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

Popsané algoritmy lze snadno modifikovat pro případy bodových množin v rovině. Symbolické kódy nebudeme znovu uvádět, neboť formální zápis algoritmů se liší pouze v dimenzi. Chybová funkce z algoritmu 4.1 má pro případ eukleidovské roviny E_2 celkem čtyři parametry – souřadnice určujícího bodu aktuální přímky a souřadnice jejího směrového vektoru. Tyto parametry v každém kroku minimalizace aktualizujeme. Chybová funkce je tedy v tomto případě reálná funkce čtyř proměnných.

Příklad 4.3: *HLEDÁNÍ OSY SYMETRIE ROVNOBĚŽNÝCH PŘÍMEK.* Navržené algoritmy nyní ukažme na příkladě experimentální bodové množiny. Předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i]\}_{i=1}^{n}$ v eukleidovské rovině E_2, které leží na rovnoběžných přímkách.

V následujícím příkladě se jedná o body, které jsou pravidelně navzorkovány na rovnoběžných úsečkách, jak vidíme na obrázku 4.18. Navíc jsou úsečky voleny ve speciální poloze (jedna jejich osa je souřadnicová osa x). Na obrázku 4.18 je rovněž zakreslena počáteční poloha přímky, která se postupně optimalizuje, a předloženy jsou její parametrické rovnice.



Obrázek 4.18: Body pravidelně rozmístěné na dvou rovnoběžných úsečkách a počáteční poloha přímky

Osu symetrie bodové množiny hledáme opět oběma technikami vícerozměrné minimalizace. Nejdříve se věnujeme jednodušší modifikované metodě největšího spádu. Obrázek 4.19 ilustruje postupné vylepšování polohy přímky (vykreslujeme jen několik poloh), na obrázku 4.20 je vidět výsledná poloha osy a vypsány jsou její parametrické rovnice. Dostatečně dobrý odhad osy symetrie zís-

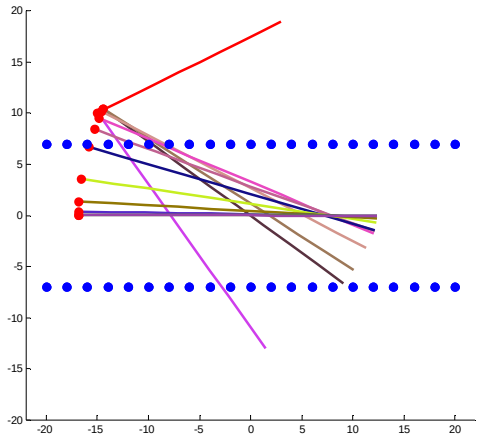
káme pro uvedenou volbu vstupních parametrů metody (200 iterací).

Stejná vstupní množina bodů je zpracována také obecnou metodou největšího spádu. Počáteční polohu přímky volíme stejně jako v metodě předchozí, viz obrázek 4.18. Postupné vylepšování polohy přímky (vykreslujeme jen několik poloh), výsledný odhad osy a parametrické rovnice odhadu osy znázorňuje obrázek 4.21. Opět je nutné experimentálně vyzkoušet vhodnou volbu vstupních parametrů. Pro uvedené hodnoty je odhad hledané osy symetrie dostatečně přesný (100 iterací).

Pro porovnání obou metod minimalizace chybové funkce jsou vykresleny také závislosti hodnoty chybové funkce na počtu iterací, viz obrázek 4.22. Hodnoty vypisujeme pro prvních sto kroků algoritmu.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

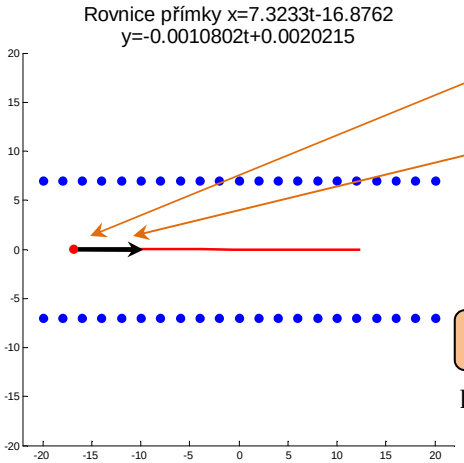
`osa_rovnobezek_spad_modifik.m` (programy/osa_rovnobezek),
`osa_rovnobezek_spad.m` (programy/ osa_rovnobezek).



Vstupní parametry pro modifikovanou metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,1$
$max\_iteration = 200$

Obrázek 4.19: Vstupní parametry minimalizační metody a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



$A_{STEP} = [-16,87619; 0,00202]$

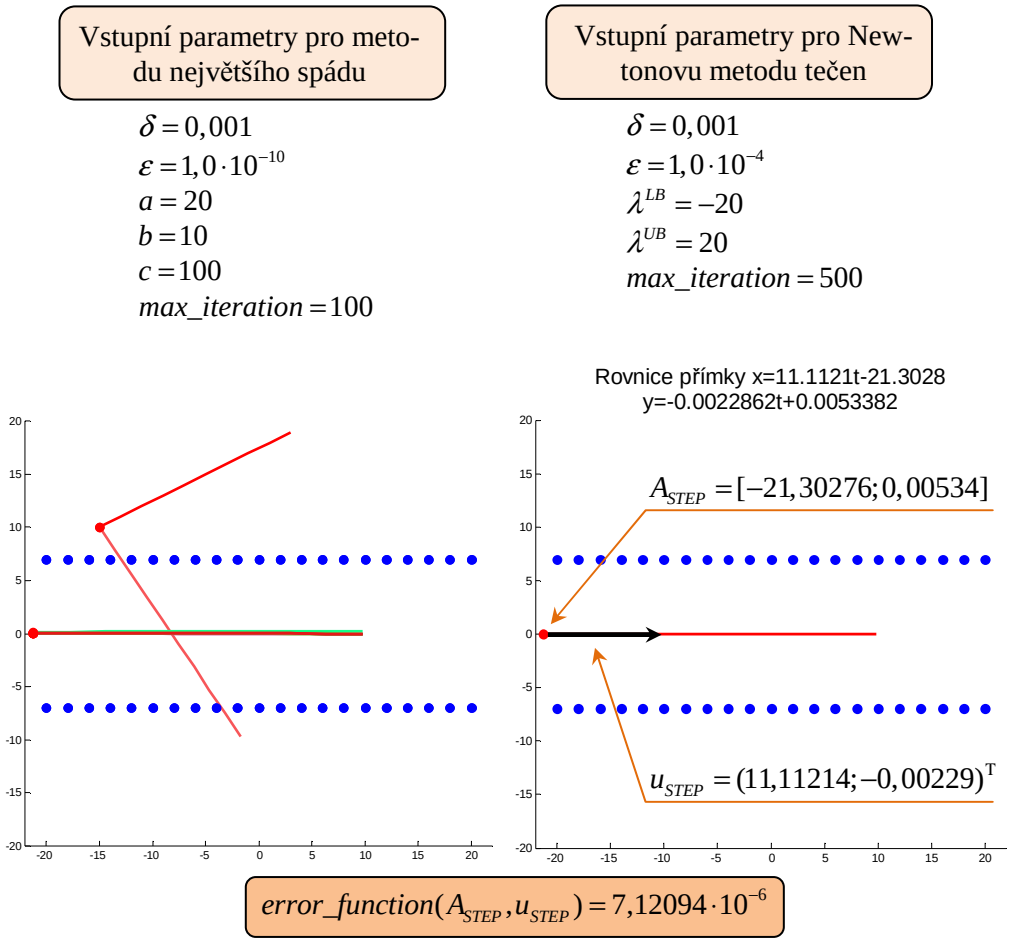
$u_{STEP} = (7,32335; -0,00108)^{T}$

určující bod a směrový vektor výsledného odhadu osy symetrie

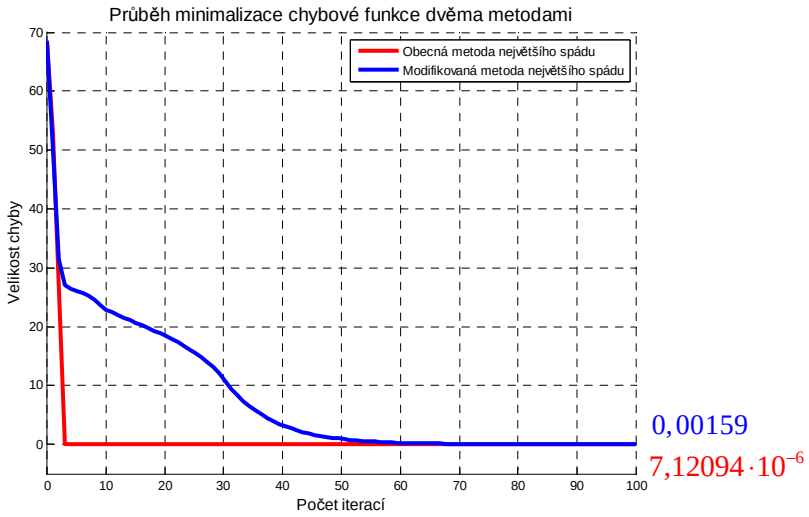
$error\_function(A_{STEP}, u_{STEP}) = 3,41003 \cdot 10^{-6}$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.20: Výsledný odhad osy – modifikovaná metoda největšího spádu



Obrázek 4.21: Vstupní parametry minimalizační metody, postupná optimalizace přímky minimalizací chybové funkce obecnou metodou největšího spádu a výsledný odhad osy symetrie



Obrázek 4.22: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

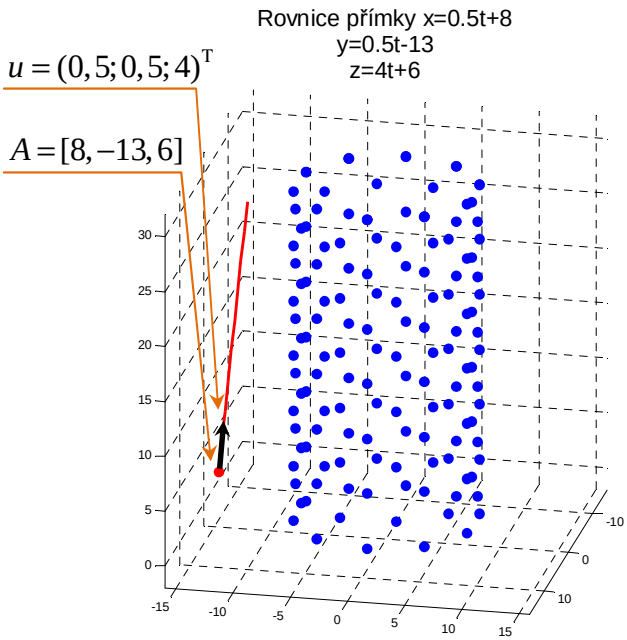
4.2.1 Případová studie v eukleidovském prostoru E_3

Zaměřme se nyní na zpracování několika vstupních bodových množin v eukleidovském prostoru E_3. Cílem je prověřit fungování navrženého postupu hledání osy rotační válcové plochy a ukázat výsledky, které metoda dává. Nebudeme již uvádět jednotlivé výpočty, předkládáme pouze hodnoty vstupních parametrů minimalizačních metod a vývoj metod prezentujeme vždy skupinami obrázků. Číselné údaje a výsledky výpočtů lze získat spuštěním programů vytvořených ve výpočetním prostředí MATLAB s názvy `osa_rot_valce_spad.m` a `osa_rot_valce_spad_modifik.m`, kde lze měnit vstupní bodové množiny. Počítačově generované množiny, které v této studii testujeme, jsou k dispozici na přiloženém médiu pod názvy `body01.txt`, `body02.txt`, … (cesta: bodove_mnoziny/osa_rot_valce).

Ve všech případech vždy předpokládejme, že je dána konečná množina bodů, které leží na povrchu rotační válcové plochy. Neuvažujeme nepřesnosti v měření, body tedy leží přesně na rotační válcové ploše. Rozlišujeme případy, kdy se jedná o pravidelné a nepravidelné množiny a případy, jak velká část rotační válcové plochy je navzorkována. Každou úlohu zpracováváme oběma minimalizačními technikami. Tradičně vždy začínáme modifikovanou metodou největšího spádu.

Regulární bodová množina

První úloha představuje doplnění příkladu 4.2. Uvažujeme opět množinu bodů, které jsou na povrchu části rotační válcové plochy rozloženy pravidelně. Polohu osy rotační válcové plochy, ze které byly body navzorkovány, volíme tentokrát rovnou ose z. Na obrázku 4.23 je vykreslena vstupní bodová množina a zvolená počáteční poloha přímky, kterou optimalizujeme, a předloženo je její parametrické vyjádření.

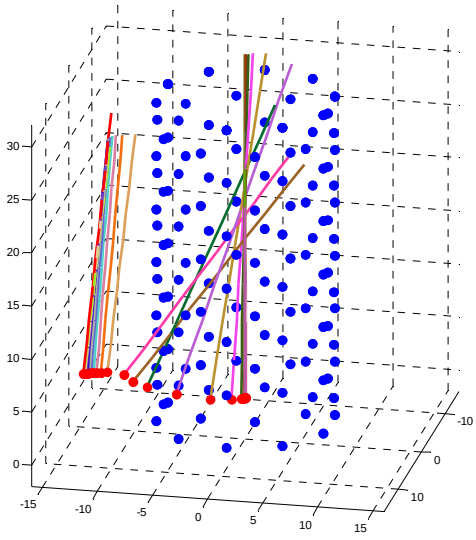


Obrázek 4.23: Body pravidelně rozmístěné na části rotační válcové plochy a počáteční poloha přímky pro obě minimalizační techniky

Obrázky 4.24-4.28 sledují postupný výpočet odhadu osy rotační válcové plochy modifikovanou a obecnou metodou největšího spádu. Výstupy metod jsou předloženy obdobnou formou jako v příkladě 4.2. Doplněn je také graf znázorňující porovnání konvergencí obou metod k minimální hodnotě chybové funkce. Zcela analogicky bychom mohli řešit úlohy, kdy je osa rotační válcové plochy, ze které body získáváme, volena v obecnějších polohách. Tyto příklady jsme

úspěšně testovali, nebudeme je zvlášť uvádět. Na přiloženém výměnném médiu jsou k dispozici další počítačově generovaná data, která lze použít k testování algoritmů.

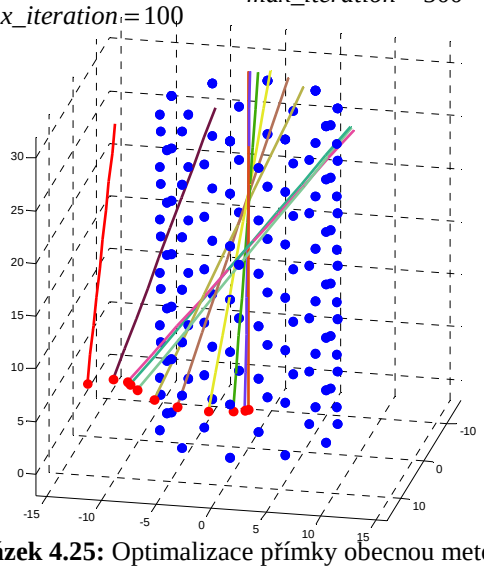
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,2$
$max\_iteration = 300$



Obrázek 4.24: Optimalizace přímky modifikovanou metodou největšího spádu a vstupní parametry metody

Metoda největšího spádu
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 20$
$b = 10$
$c = 100$
$max\_iteration = 100$

Newtonova metoda tečen
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -200$
$\lambda^{UB} = 200$
$max\_iteration = 500$



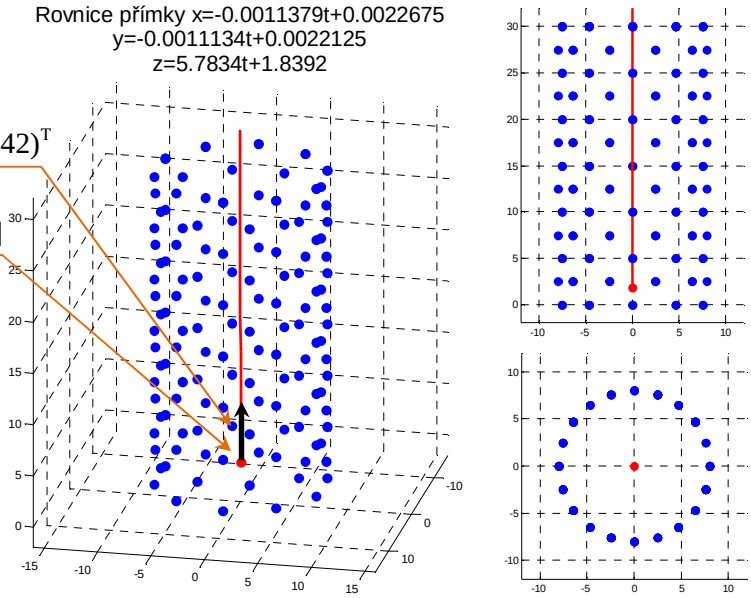
Obrázek 4.25: Optimalizace přímky obecnou metodou největšího spádu a vstupní parametry metody

$u_{STEP} = (-0,00114; -0,00111; 5,78342)^{T}$

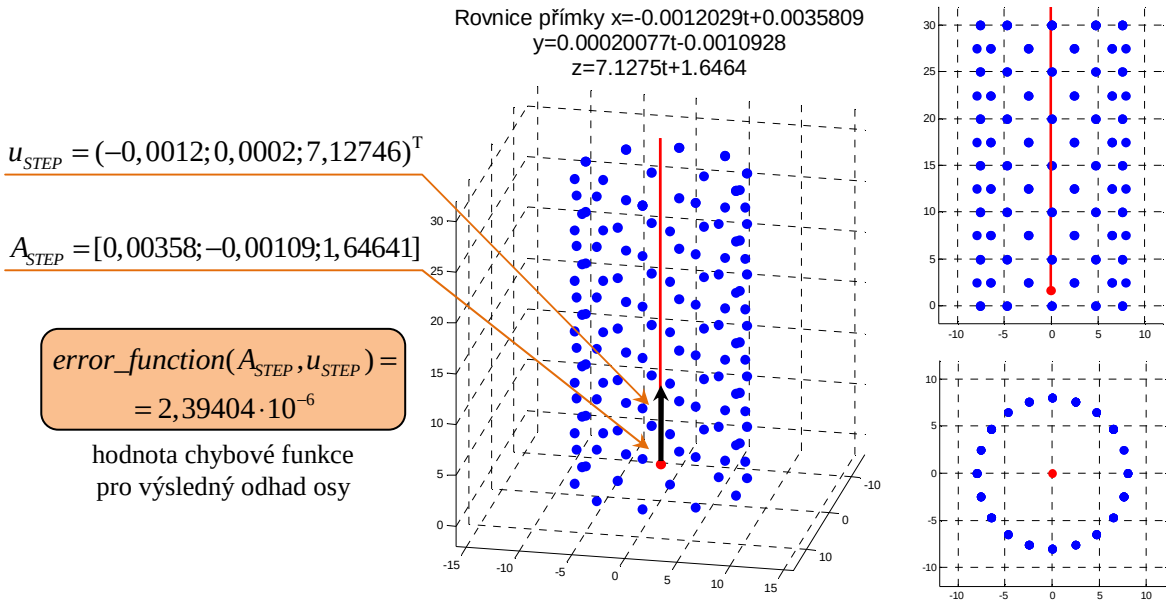
$A_{STEP} = [0,00227; 0,00221; 1,83919]$

$error\_function(A_{STEP}, u_{STEP}) = 3,99649 \cdot 10^{-6}$

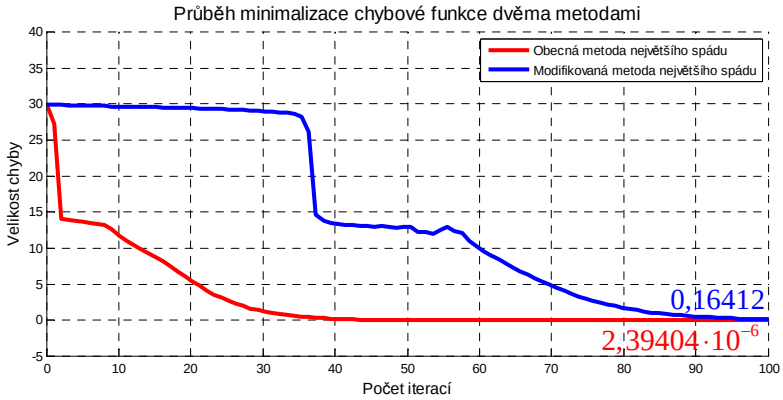
hodnota chybové funkce pro výsledný odhad osy



Obrázek 4.26: Výsledný odhad osy v názorném promítání a v pravoúhlých pohledech – modifikovaná metoda největšího spádu



Obrázek 4.27: Výsledný odhad osy v názorném promítání a v pravoúhlých pohledech – obecná metoda největšího spádu

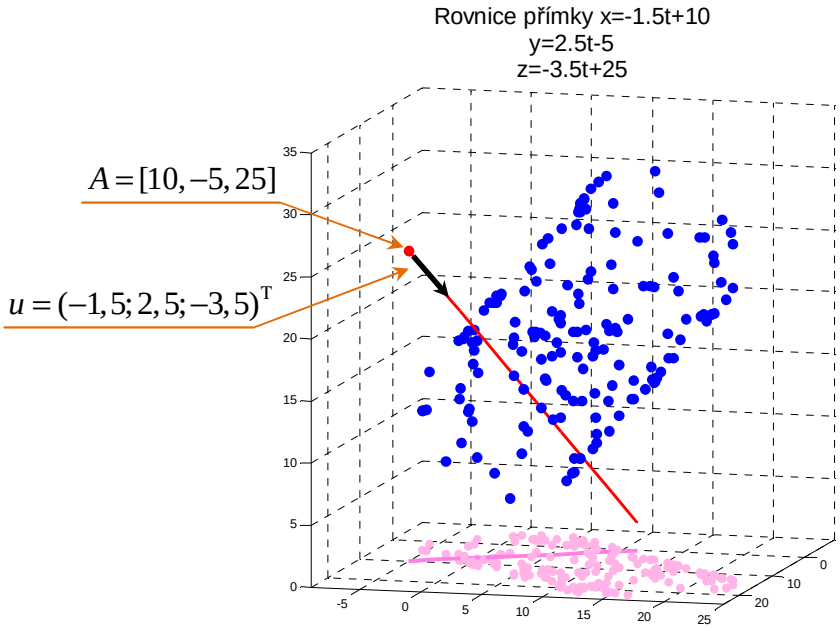


Obrázek 4.28: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

Náhodná bodová množina – případ (a)

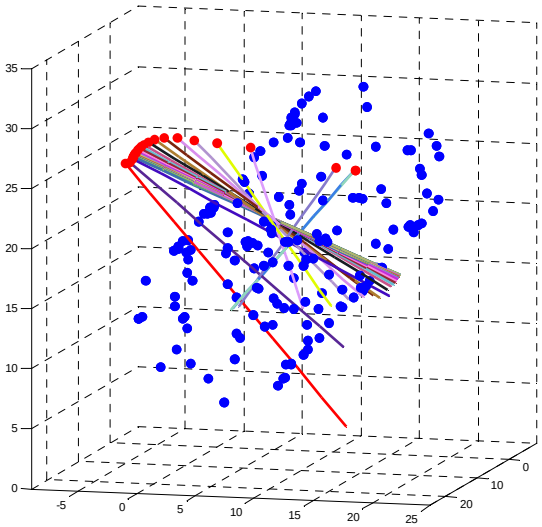
V dalším příkladě se věnujme množině bodů, které jsou na rotační válcové ploše navzorkovány náhodně a netvoří tedy pravidelnou síť. V úvahu nebereme nepřesnost, která by při vzorkování mohla vzniknout. Body vstupní množiny tedy leží přesně na rotační válcové ploše. Uvažujme množinu bodů, která je znázorněna na obrázku 4.29. Osa rotační válcové plochy, ze které byly body získány, je určena směrovým vektorem $(1,1,1)^T$. Tuto speciální polohu volíme z toho důvodu, abychom mohli lépe posoudit přesnost výsledného odhadu osy. Opět je zakreslena výchozí poloha přímky a zapsány jsou její parametrické rovnice. Pro lepší názornost a orientaci v prostorové situaci přidáváme do obrázku ještě půdorysy bodů vstupní množiny a půdorys počáteční polohy přímky. Obrázky 4.30-4.34 ilustrují postupný výpočet odha-

du osy rotační válcové plochy oběma minimalizačními technikami. V obrázcích znázorňujících výsledný odhad osy jsou opět doplněny půdorysy bodů vstupní množiny a půdorys výsledné přímky. Opět je předloženo také porovnání konvergence obou metod.



Obrázek 4.29: Body náhodně rozmístěné na části rotační válcové plochy a počáteční poloha přímky pro obě minimalizační techniky

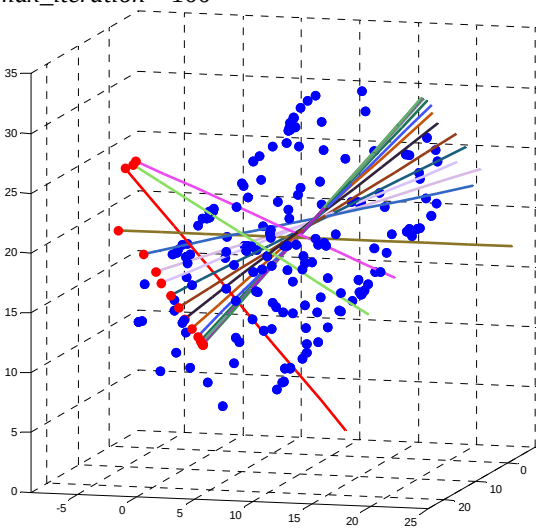
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,1$
$max\_iteration = 800$



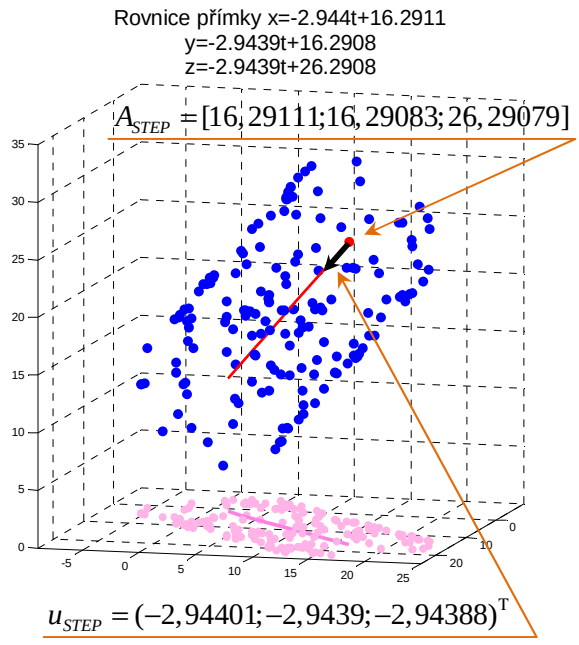
Obrázek 4.30: Optimalizace přímky modifikovanou metodou největšího spádu a vstupní parametry metody

Metoda největšího spádu
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 20$
$b = 10$
$c = 100$
$max\_iteration = 100$

Newtonova metoda tečen
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -200$
$\lambda^{UB} = 200$
$max\_iteration = 500$



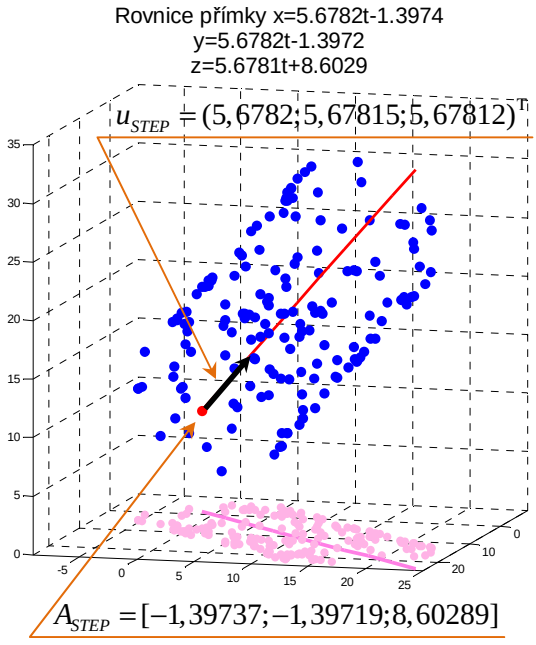
Obrázek 4.31: Optimalizace přímky obecnou metodou největšího spádu a vstupní parametry metody



$error\_function(A_{STEP}, u_{STEP}) = 1,58688 \cdot 10^{-8}$

hodnota chybové funkce pro výsledný odhad osy

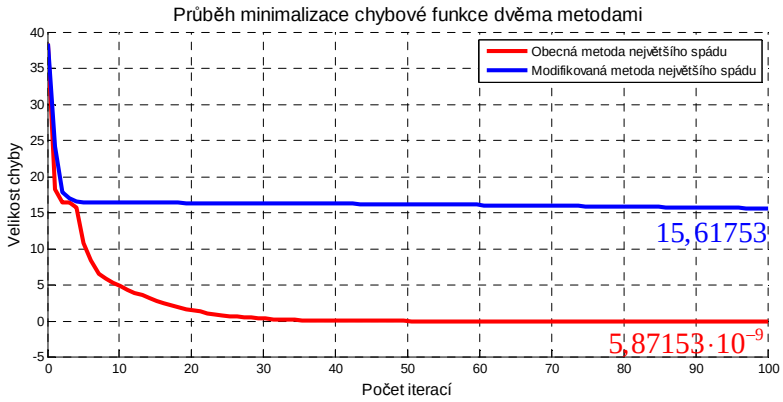
Obrázek 4.32: Výsledný odhad osy – modifikovaná metoda největšího spádu



$error\_function(A_{STEP}, u_{STEP}) = 5,87153 \cdot 10^{-9}$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.33: Výsledný odhad osy – obecná metoda největšího spádu

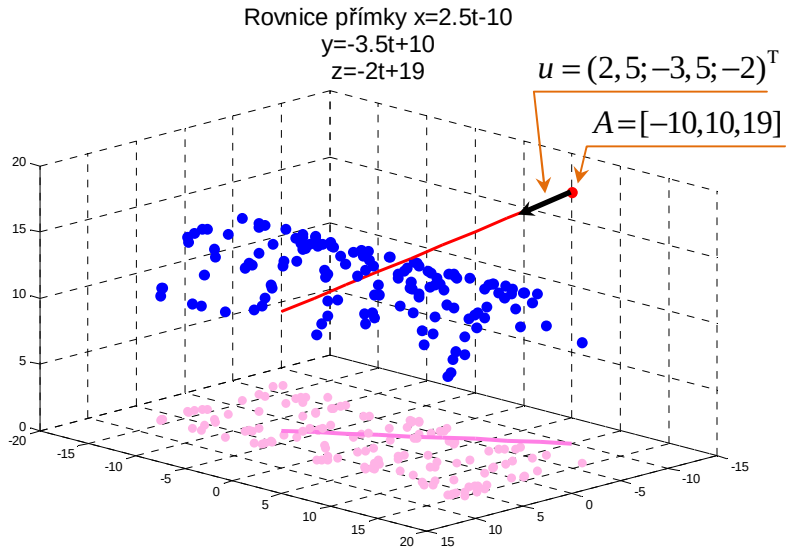


Obrázek 4.34: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

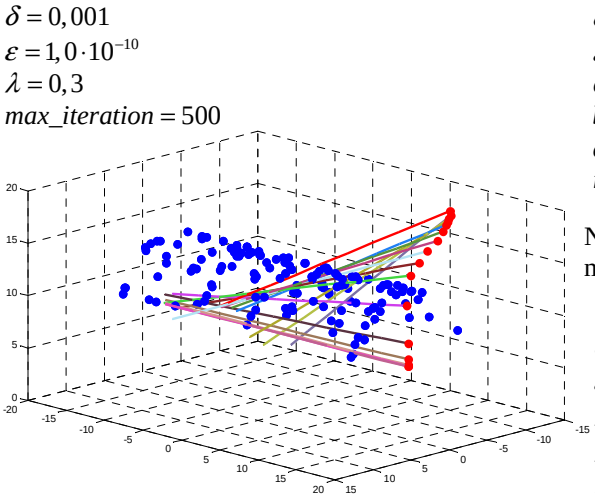
Náhodná bodová množina – případ (b)

Další příklad vstupní bodové množiny se více blíží reálné situaci. Body jsou opět navzorkovány náhodně na části rotační válcové plochy, ovšem jedná se pouze o body ležící na její úse-

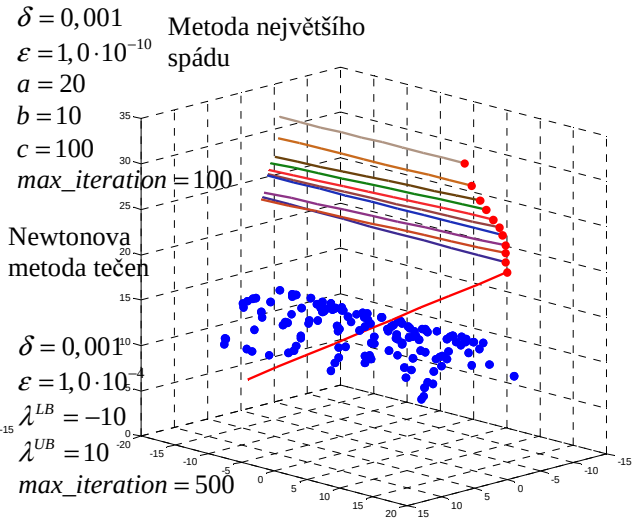
či, viz obrázek 4.35. Polohu osy rotační válcové plochy, ze které byly body získány, volíme tentokrát rovnou ose y. Výstupy minimalizačních technik jsou předloženy obdobně jako v předchozích případech. Modifikovanou metodou největšího spádu získáváme dostatečně dobrý odhad hledané osy rotační válcové plochy.



Obrázek 4.35: Body nepravidelně rozmístěné na části úseče rotační válcové plochy a počáteční poloha přímky pro obě minimalizační techniky

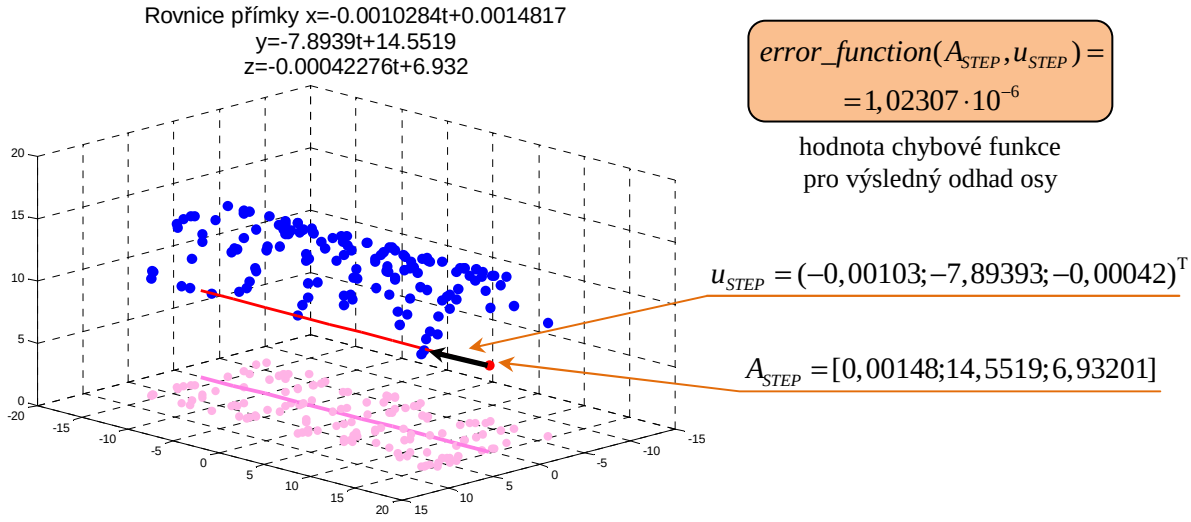


Obrázek 4.36: Optimalizace přímky modifikovanou metodou největšího spádu a vstupní parametry metody

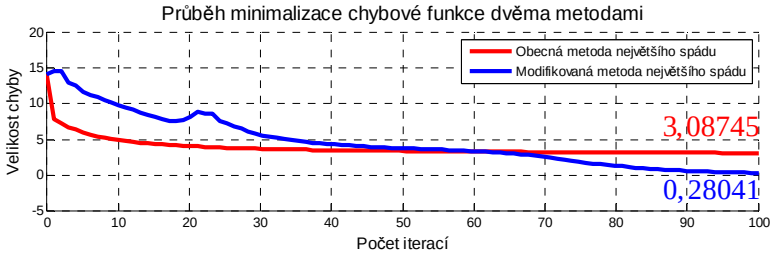


Obrázek 4.37: Optimalizace přímky obecnou metodou největšího spádu a vstupní parametry metody – nestabilní chování

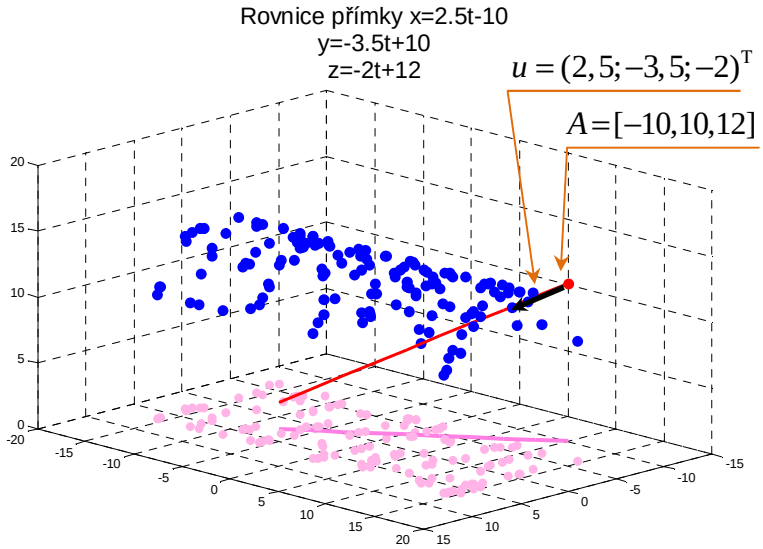
Průběh obecné metody největšího spádu je v tomto případě zajímavý. Pro zvolenou počáteční polohu přímky se metoda chová nestabilně, neboť dochází k tomu, že konverguje k minimu, které je v nekonečnu. Pro tuto volbu tedy odhad osy rotační válcové plochy nezískáme. Vývoj výpočtů je znázorněn na obrázcích 4.36-4.39. I v tomto případě předkládáme porovnání konvergencí obou metod. Abychom dostali odhad osy i pro obecnou metodu největšího spádu, je nutné zvolit jinou počáteční polohu přímky. Na obrázku 4.40 je ukázáno, z jaké počáteční polohy přímky vycházíme, doplněny jsou také parametrické rovnice výchozí přímky. Ilustrace vývoje metody jsou znázorněny na obrázcích 4.41-4.43.



Obrázek 4.38: Výsledný odhad osy – modifikovaná metoda největšího spádu



Obrázek 4.39: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací, nestabilní chování obecné metody



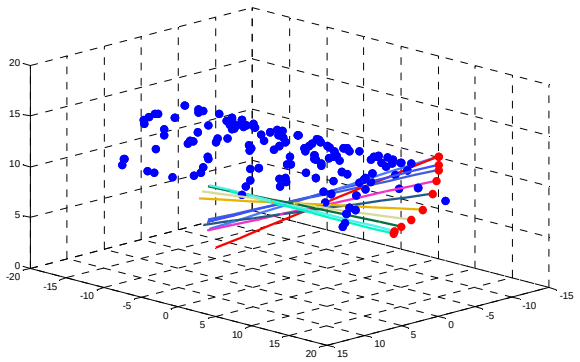
Obrázek 4.40: Body nepravidelně rozmístěné na části úseče rotační válcové plochy a nová volba počáteční polohy přímky pro obecnou metodu největšího spádu

Metoda největšího spádu

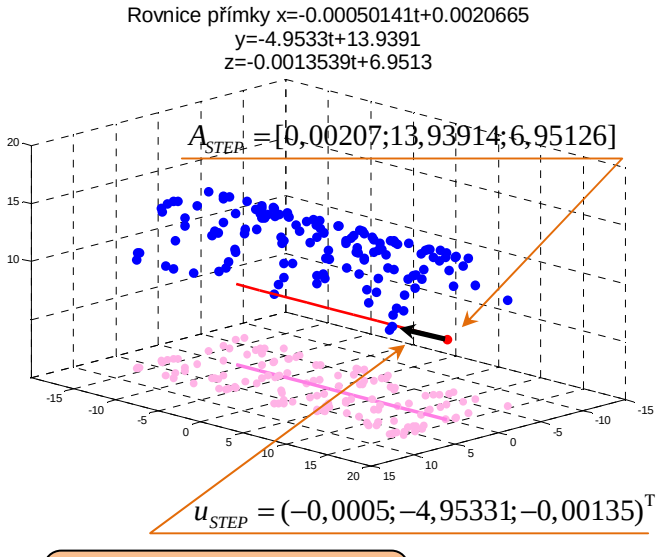
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 20$
$b = 10$
$c = 100$
$max\_iteration = 100$

Newtonova metoda tečen

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -10$
$\lambda^{UB} = 10$
$max\_iteration = 500$



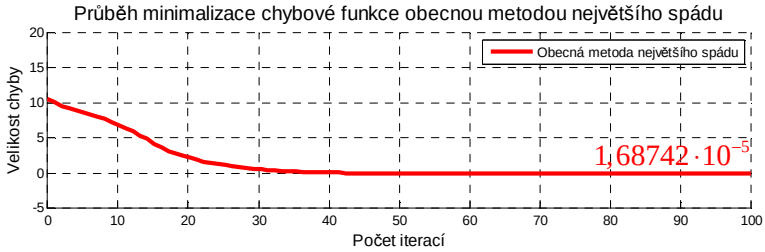
Obrázek 4.41: Optimalizace přímky obecnou metodou největšího spádu a vstupní parametry metody



$error\_function(A_{STEP}, u_{STEP}) = 1,68742 \cdot 10^{-5}$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.42: Výsledný odhad osy – obecná metoda největšího spádu



Obrázek 4.43: Minimalizace chybové funkce obecnou metodou největšího spádu pro novou volbu počáteční polohy přímky – závislost hodnoty chybové funkce na počtu iterací

Další bodové množiny, které jsme úspěšně testovali, odpovídají reálným situacím, kdy dochází k nepřesnostem při snímání bodů. Body vstupních množin tedy nemusí ležet na rotační válcové ploše přesně. Tyto testovací množiny jsou na přiloženém výměnném médiu k dispozici.

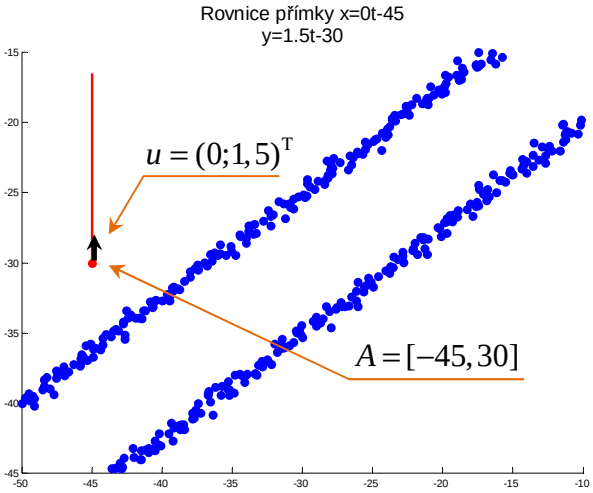
4.2.2 Případová studie v eukleidovské rovině E_2

V eukleidovské rovině E_2 se zaměřme na ukázku jedné vstupní bodové množiny. Číselné údaje a výsledky výpočtů lze získat spuštěním programů vytvořených ve výpočetním prostředí MATLAB s názvy `osa_rovnobezek_spad_modifik.m` a `osa_rovnobezek_spad.m`, kde lze měnit vstupní bodové množiny. Počítačově generovaná bodová množina, kterou v této studii testujeme, a další množiny jsou k dispozici na přiloženém výměnném médiu pod názvy `body01.txt`, `body02.txt`, … (cesta: bodove_mnoziny/osa_rovnobezek).

Náhodná zašuměná bodová množina

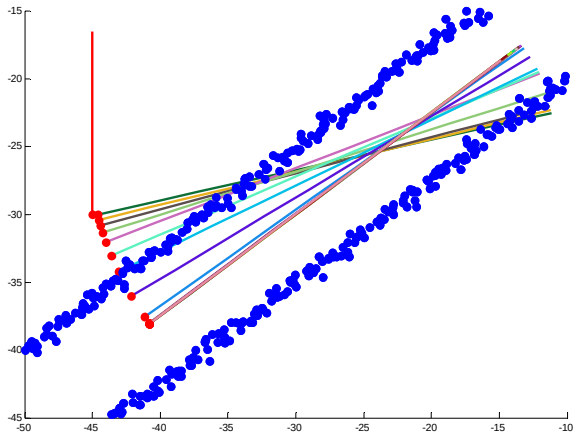
V následujícím případě předpokládejme, že je dána konečná množina bodů, které leží na rovnoběžných přímkách. Simulujeme reálnou situaci, kdy dochází k nepřesnostem v měření, body množiny tedy na přímkách neleží přesně a jsou zašuměny (opět modelujeme situaci normálním rozdělením), viz obrázek 4.44. Nacházíme tedy hrubší odhad osy symetrie této bodové množiny. Tradičně tuto úlohu zpracováváme oběma minimalizačními technikami.

Na obrázku 4.44 je zakreslena prvotní poloha přímky a zapsány jsou její parametrické rovnice. Obrázky 4.45-4.49 sledují postupný výpočet odhadu osy symetrie této bodové množiny modifikovanou a obecnou metodou největšího spádu. Rovněž je předloženo porovnání konvergencí obou minimalizačních technik.



Obrázek 4.44: Body náhodně rozmístěné a zašuměné na dvou rovnoběžných úsečkách a počáteční poloha přímky pro obě minimalizační techniky

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,1$
$max\_iteration = 1000$



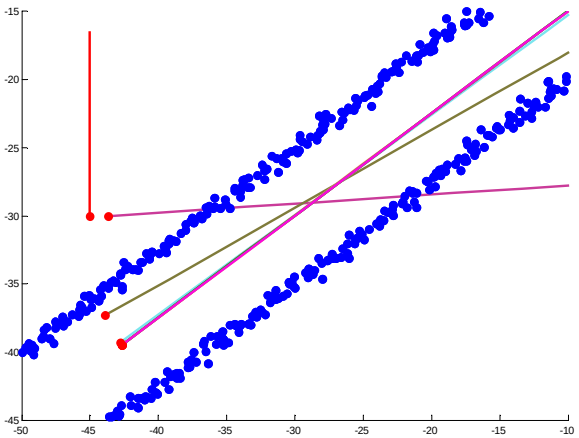
Obrázek 4.45: Optimalizace přímky modifikovanou metodou největšího spádu a vstupní parametry metody

Metoda největšího spádu

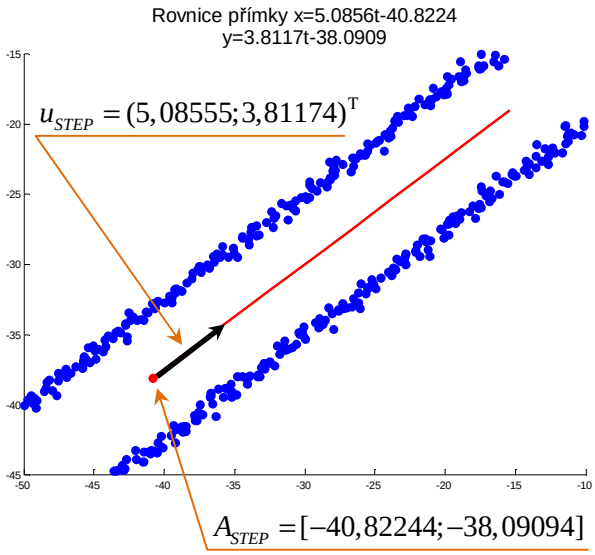
$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 200$
$b = 100$
$c = 100$
$max\_iteration = 100$

Newtonova metoda tečen

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -200$
$\lambda^{UB} = 200$
$max\_iteration = 500$



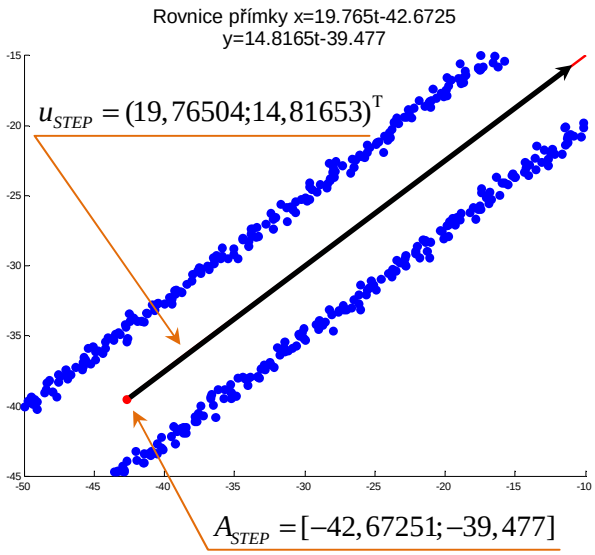
Obrázek 4.46: Optimalizace přímky obecnou metodou největšího spádu a vstupní parametry metody



$error\_function(A_{STEP}, u_{STEP}) = 0,11255$

hodnota chybové funkce pro výsledný odhad osy

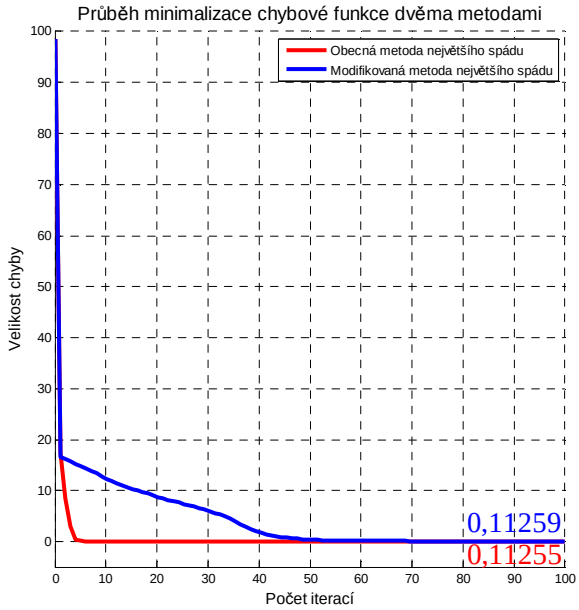
Obrázek 4.47: Výsledný odhad osy – modifikovaná metoda největšího spádu



$error\_function(A_{STEP}, u_{STEP}) = 0,11255$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.48: Výsledný odhad osy – obecná metoda největšího spádu



Obrázek 4.49: Porovnání minimalizací chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

V případové studii jsme zkoumali speciální počítačově generované bodové množiny. V kapitole 6 (*Experimenty s reálnými daty*) budou obdobně zpracována bodová mračna odpovídající reálným povrchům.

4.3 Hledání osy obecné rotační plochy

Rozšiřme předchozí postupy a představme nově navrženou metodu pro hledání osy obecné rotační plochy. Opět se bude jednat o iterační algoritmus založený na minimalizacích chybové funkce. Je zřejmé, že nelze použít chybovou funkci popsanou v algoritmu 4.1 vycházející z vlastnosti osy rotační válcové plochy, což je přímka, pro kterou platí, že má od každého bodu plochy stejnou vzdálenost rovnající se poloměru této rotační válcové plochy. Nyní je tedy nutné zavést chybovou funkci novou.

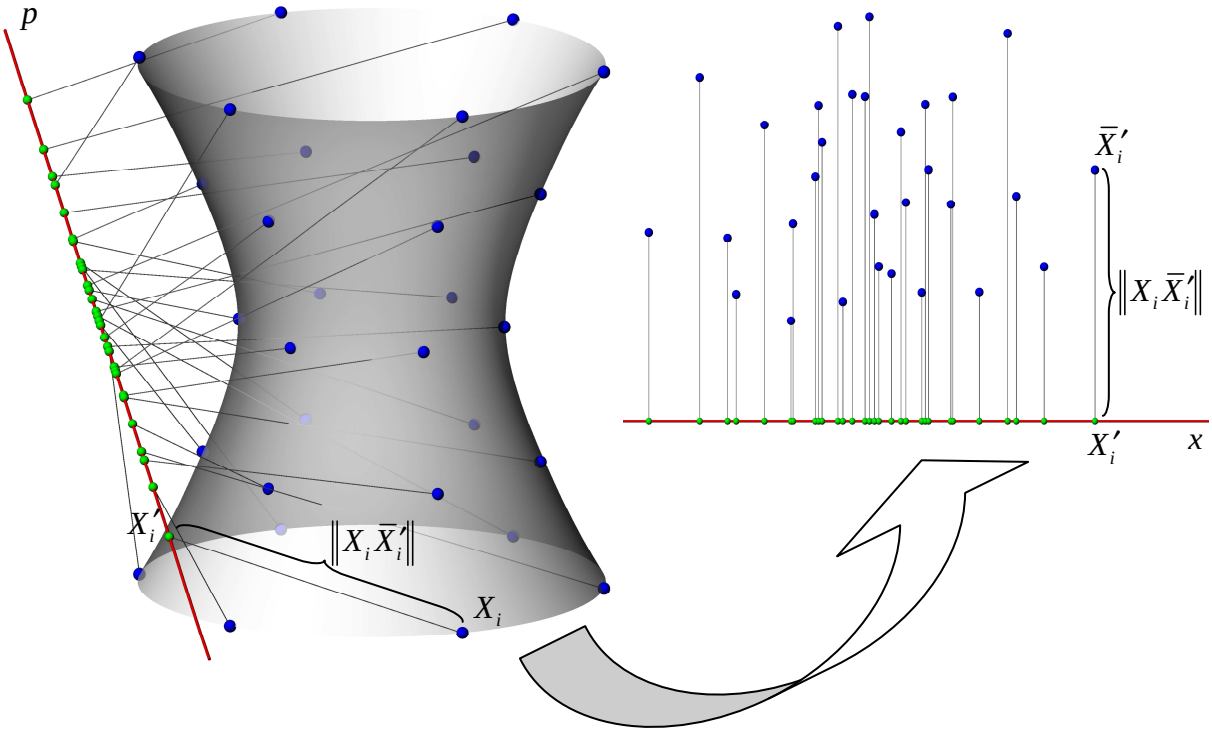
V oddíle o ortogonálním prokládání dat přímkou jsme při hledání osy obecné rotační plochy v případové studii použili vlastnost, že rotační plochy jsou podle své osy souměrné. Z toho rovněž plyne, že je rotační plocha souměrná podle každé roviny procházející osou a v této rovině vzniká osová symetrie, (Surynková, 2008a). Ze stejných vlastností budeme vycházet i nyní.

Předpokládejme, že máme k dispozici neprázdnou množinu $\mathcal{X}$ n bodů v eukleidovském prostoru E_3 získané navzorkováním libovolné rotační plochy, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^n$. Opět připouštíme různé možnosti. Body jsou navzorkovány pravidelně, nepravidelně nebo některé části bodové množiny chybí a množina neobsahuje s každým bodem také bod, který je souměrný podle osy rotační plochy, ze které byly body získány. Hledáme-li tedy osy těchto částí ploch, nelze použít metodu ortogonálního prokládání přímkou nebo metodu PCA. I v tomto případě hledáme osu iteračně s použitím odvozených minimalizačních postupů. V prvním kroku algoritmu zvolíme libovolnou počáteční polohu přímky, kterou v dalších krocích optimalizujeme, dokud se přímka dostatečně nepřiblíží ose rotační plochy. V každém kroku iteračního algoritmu počítáme vektor $distance \in \mathbb{R}_n$, jehož složky jsou vzdálenosti všech bodů $\{X_i\}_{i=1}^n$ rotační plochy od aktuální přímky (v prvním kroku od zvolené přímky), tj. *i*-tá složka vektoru $distance$ je určena následovně

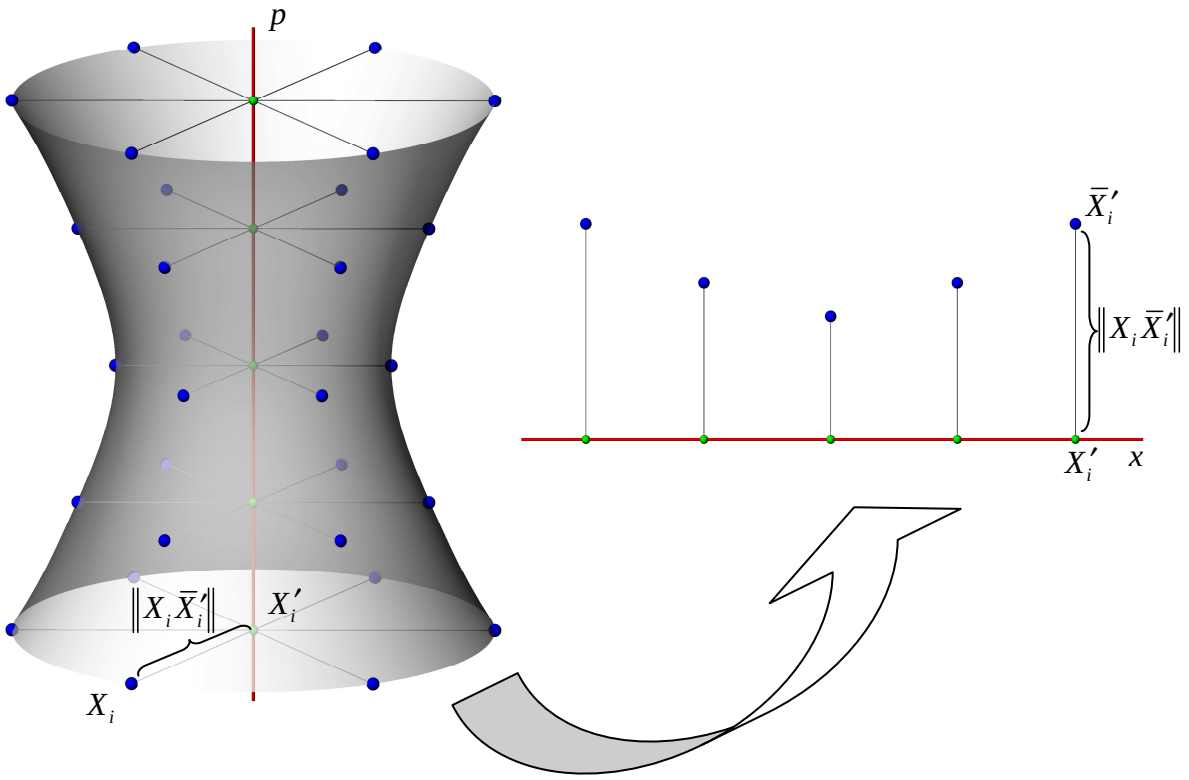
$$distance_i = \|X_i - X_i'\|, \tag{4.9}$$

kde bod X_i' je ortogonální průmět bodu X_i na aktuální přímku.

Vzdálenosti od aktuální přímky, tedy složky vektoru $distance$, přirozeně určují funkční závislost. Tuto závislost zakreslujeme do nově zvoleného souřadného systému v rovině tak, že aktuální přímka splývá s osou *x*, na kterou vynášíme pravoúhlé průměty $\{X_i'\}_{i=1}^n$ bodů $\{X_i\}_{i=1}^n$ vstupní množiny na aktuální přímku. Funkční hodnoty odpovídají vzdálenostem bodů od aktuální přímky, tj. složkám vektoru $distance$. Obrazy bodů $\{X_i\}_{i=1}^n$ ve zvolené soustavě souřadnic v rovině značíme $\{\overline{X}_i\}_{i=1}^n$. Orientace přímky a počátek nové soustavy souřadnic jsou určeny libovolně a to například tak, že počátek soustavy je zvolen v určujícím bodě aktuální přímky a orientace této přímky je určena podle směrového vektoru. Tento proces, který opakujeme v každém kroku algoritmu, je zakreslen na obrázku 4.50. Kvůli přehlednosti volíme na rotační ploše řídkou pravidelnou množinu bodů.



Obrázek 4.50: Výpočet vzdáleností všech bodů vstupní množiny od aktuální přímky a zakreslení těchto vzdáleností do nové soustavy souřadnic v rovině



Obrázek 4.51: Výpočet vzdáleností všech bodů vstupní množiny od aktuální přímky, která splývá s osou rotační plochy a zakreslení těchto vzdáleností do nové soustavy souřadnic v rovině

Podívejme se, jaký výsledek získáme, zvolíme-li za aktuální přímku přímo osu rotační plochy. Tento případ je znázorněn na obrázku 4.51. Obrazy bodů $\{\overline{X}_i\}_{i=1}^n$ ve zvolené soustavě souřadnic v rovině lze proložit křivku, která odpovídá *polomeridiánu* rotační plochy. Polomeridián získáme jako libovolný rovinný osový řez rotační plochou, který leží v jedné polorovině určené osou plochy (Surynková, 2008a). Všechny polomeridiány jsou navzájem shodné křivky. K této situaci chceme dospět iteračním algoritmem, kdy v každém kroku vylepšujeme polohu přímky.

Zaveďme metodu hledání osy rotační plochy nejdříve intuitivně. V každém kroku iteračního algoritmu chceme vylepšit polohu přímky tak, abychom v nové soustavě souřadnic v rovině dostali takové obrazy bodů, které leží na křivce tvarem se blížící polomeridiánu. Docílíme toho algoritmicky následovně. Pro každý obraz $\{\overline{X}_i\}_{i=1}^n$ bodů vstupní množiny zvolíme v soustavě souřadnic v rovině nějaké ε-okolí, $\varepsilon > 0$, $\varepsilon \in \mathbb{R}$. Pro bod $\overline{X}_i$ budeme toto okolí značit $B_\varepsilon(\overline{X}_i)$ (*z ang. ball*). Dále zavádíme chybovou funkci, kterou definujeme jako obsah množiny sjednocení všech okolí, tj.

$$S = \bigcup_{i=1}^{n} B_\varepsilon(\overline{X}_i). \tag{4.10}$$

Hledáme minimum chybové funkce, neboť víme, že pro případ, kdy je aktuální přímka přímo osou rotační plochy, se okolí obrazů $\{\overline{X}_i\}_{i=1}^n$ nejvíce překrývají. Obsah S definovaný v (4.10) je tedy minimální. Minimalizaci chybové funkce provádíme pomocí diferenciální numerické metody největšího spádu a to jak její obecnou, tak i modifikovanou verzí s konstantní délkou kroku.

Chybová funkce má stejně jako v případě hledání osy rotační válcové plochy celkem šest parametrů (pracujeme v eukleidovském prostoru E_3) – souřadnice určujícího bodu aktuální přímky a souřadnice jejího směrového vektoru, které v každém kroku algoritmu aktualizujeme. Jedná se tedy o reálnou funkci šesti proměnných.

Výpočet chybové funkce popišme v následujícím symbolickém kódu jako algoritmus 4.5. Chybová funkce je v pseudokódu označena jako *error_function* a její hodnota pro konkrétních šest parametrů jako *error*.

Algoritmus 4.5: *VÝPOČET CHYBOVÉ FUNKCE ERROR_FUNCTION PRO OBECNOU ROTAČNÍ PLOCHU.* Vstupními parametry chybové funkce jsou určující bod A a směrový vektor u aktuální přímky, výstupem je reálná hodnota *error*.

Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, které leží na rotační ploše. Symbolický kód využívá několik proměnných. Složky vektoru *distance* jsou vzdálenosti všech bodů rotační plochy od aktuální přímky (v prvním kroku od zvolené přímky). Proměnná *projection* jsou ortogonální průměty bodů vstupní množiny na aktuální přímku, $coords_x$ a $coords_y$ jsou souřadnice obrazů bodů vstupní množiny v nové soustavě souřadnic v rovině a *rand_points* jsou náhodně generované body v nové soustavě souřadnic v rovině v obdélníkové ob-

lasti vymezené nejmenšími a největšími souřadnicemi $coords_x$ a $coords_y$. Proměnné *surface_area*, *surface_count* a *surface_ratio* slouží k výpočtu obsahu sjednocení všech okolí obrazů bodů vstupní množiny. Proměnná t je hodnota parametru, pro kterou získáme bod na aktuální přímce, tedy ortogonální průmět nějakého bodu vstupní množiny. V nové soustavě souřadnic v rovině je hodnota t x-ovou souřadnicí obrazu bodu vstupní množiny. Proměnné x_{min} a x_{max} jsou nejmenší a největší souřadnice z $coords_x$, y_{min} a y_{max} jsou nejmenší a největší souřadnice z $coords_y$. Do proměnné *inside* ukládáme pravdivostní hodnotu, která určuje, zda je náhodný bod uvnitř nebo vně okolí obrazu bodu vstupní množiny. Výpočet obsahu provádíme známou metodou Monte Carlo, kterou popíšeme dále. Proměnná *error* je výstupní hodnotou chybové funkce *error_function* pro vstupní parametry A a u, jedná se o obsah sjednocení všech okolí obrazů bodů vstupní množiny podle (4.10).

V pseudokódu používáme ještě globální proměnné *accuracy* a ε, které určují nastavení metody určení obsahu množiny sjednocení všech okolí. Vhodná je například volba

$accuracy = 1000$

$\varepsilon = 10$.

---

function *error_function*(A,u): **real**

1: $distance \leftarrow \emptyset$
2: $projection \leftarrow \emptyset$
3: $coords_x \leftarrow \emptyset$
4: $coords_y \leftarrow \emptyset$
5: $rand\_points \leftarrow \emptyset$
6: $surface\_count \leftarrow 0$
7: $unit\_vector \leftarrow u / \|u\|$
8: **for** $i = 1,2,...,n$ **do**
9: $\quad projection[i] \leftarrow A + (unit\_vector \cdot (X[i] - A))unit\_vector$
$\quad$ %{výpočet ortogonálního průmětu bodu na aktuální přímku odvozen v oddíle o ortogonálním prokládání přímkou}
10: $\quad distance[i] \leftarrow \|X[i] - projection[i]\|$
11: $\quad$ **let** $t \in \mathbb{R}$ **such that** $projection[i] = A + t \cdot unit\_vector$
12: $\quad coords_x[i] \leftarrow t$
13: $\quad coords_y[i] \leftarrow \|X[i] - projection[i]\|$
14: $x_{min} \leftarrow min(coords_x)$
15: $x_{max} \leftarrow max(coords_x)$
16: $y_{min} \leftarrow min(coords_y)$
17: $y_{max} \leftarrow max(coords_y)$
18: **for** $j = 1,2,...,accuracy$ **do**
19: $\quad rand\_points_x \leftarrow (x_{max} - x_{min}) \cdot rand + x_{min}$
$\quad$ %{náhodné číslo z intervalu (x_{min}, x_{max}) }%
20: $\quad rand\_points_y \leftarrow (y_{max} - y_{min}) \cdot rand + y_{min}$
$\quad$ %{náhodné číslo z intervalu (y_{min}, y_{max}) }%
21: **for** $j = 1,2,...,accuracy$ **do**
22: $\quad inside \leftarrow False$

23: | **for** $i = 1, 2, ..., n$ **do**
24: | | **if** $\left\| (coords_x[i], coords_y[i]) - (rand\_points_x[j], rand\_points_y[j]) \right\| < \varepsilon$ **then**
25: | | | $inside \leftarrow True$
26: | | | **break**
27: | **if** *inside* **then**
28: | | $surface\_count \leftarrow surface\_count + 1$
29: $surface\_area \leftarrow (x_{max} - x_{min}) \cdot (y_{max} - y_{min})$
30: $surface\_ratio \leftarrow surface\_count / accuracy$
31: $error \leftarrow surface\_ratio \cdot surface\_area$
32: **return** *error*

Výpočet obsahu množiny sjednocení všech okolí z (4.10) provádíme pomocí známé stochastické výpočetní metody *Monte Carlo* (Fabian a Kluiber, 1998). Tato část algoritmu 4.5 je v pseudokódu zvýrazněna, jedná se o řádky 18 až 28.

V nové soustavě souřadnic v rovině volíme obdélníkovou oblast vymezenou nejmenšími a největšími *x*-ovými a *y*-ovými souřadnicemi bodů $\left\{\overline{X}_i\right\}_{i=1}^{n}$ a v této oblasti generujeme náhodné body (řádky 18 až 20). Počet těchto bodů, který je dán proměnnou *accuracy*, určuje přesnost metody Monte Carlo. Dále zjišťujeme, zda náhodně vygenerovaný bod leží v obdélníkové oblasti uvnitř ε-okolí některého bodu z $\left\{\overline{X}_i\right\}_{i=1}^{n}$. Při nalezení prvního takového okolí zvýšíme hodnotu proměnné *surface_count* o jedna a přejdeme ke zkoumání dalšího náhodně vygenerovaného bodu (řádky 21 až 28). Když náhodný bod neleží v žádném okolí bodů z $\left\{\overline{X}_i\right\}_{i=1}^{n}$, proměnná *surface_count* se nemění. Obsah obdélníkové oblasti je uložen v proměnné *surface_area*. Výraz *surface_count* / *accuracy* aproximuje poměr obsahu množiny sjednocení všech okolí ku obsahu obdélníkové oblasti. Obsah, tj. hodnota chybové funkce, je pak dána výrazem *surface_ratio* · *surface_area*.

Poznamenejme, že volba hodnoty ε je velice problematická, není znám postup, jak ji automaticky určit. Správnou hodnotu je třeba nalézt experimentálně. Navíc na volbě této hodnoty závisí přesnost výpočtu hodnoty chybové funkce. I přes vhodnou volbu ε se může stát, že se metoda výpočtu chybové funkce chová nestabilně, do jisté míry lze stabilitu metody ovlivnit také tvarem okolí. Nestabilita metody je zapříčiněna rovněž volbou konečného počtu náhodných bodů v obdélníkové oblasti, neboť při dalších iteracích algoritmu dostáváme nepatrně jiný odhad obsahu množiny sjednocení všech okolí.

Poznámka 4.1 (*Reálná implementace pseudokódu*)**.** Uveďme několik důležitých poznámek k reálné implementaci symbolického kódu popsaného v algoritmu 4.5. Algoritmus 4.5 je navržen tak, aby byl snadno optimalizovatelný zavedením datových struktur, které by umožnily rychlejší vyhledávání blízkých okolí. Zároveň je tato metoda paralizovatelná, to znamená, že ji lze uzpůsobit pro paralelní architektury. Máme-li *n* procesorů, každý procesor může zpracovávat *1/n* bodů. Tato úprava je zcela nezbytná v případech zpracování množin popisující reálné povrchy, které obsahují statisíce až milióny bodů. Kvůli implementacím našich algoritmů ve výpočetním prostředí MATLAB se těmito optimalizacemi nezabýváme a zmiňujeme

je pro případ přepsání algoritmů do programovacích jazyků typu *C* (Kernighan a Ritchie, 1988). ■

Chybovou funkci popsanou v pseudokódu jako algoritmus 4.5 nyní minimalizujeme nejdříve modifikovanou metodou největšího spádu s konstantní délkou kroku. Využijeme k tomu algoritmus 4.2, který slouží k nalezení osy rotační válcové plochy, neboť proces minimalizace zůstává stejný i pro hledání osy obecné rotační plochy. Algoritmus 4.2 však bude využívat externí funkci *error_function* popsanou symbolickým kódem v algoritmu 4.5. Je nutné vhodně zvolit globální proměnné δ, ε, λ a *max_iteration*, které v algoritmu 4.2 používáme a které určují nastavení minimalizace. Vhodná volba globálních proměnných δ a ε zůstává stejná, tedy $\delta = 0,001$ a $\varepsilon = 1,0 \cdot 10^{-10}$. Volbu ostatních globálních proměnných uvedeme u konkrétních příkladů, neboť je nutné správné vstupní parametry metody vyzkoušet experimentálně.

Mnohem komplikovanější situace nastává v případě, chceme-li chybovou funkci popsanou v algoritmu 4.5 minimalizovat obecnou metodou největšího spádu. Je totiž velice složité vypozorovat správnou volbu parametrů nejen obecné metody největšího spádu, ale také Newtonovy metody tečen pro jednorozměrnou minimalizaci v každém kroku algoritmu. Hodnoty volitelných parametrů metod budeme proto uvádět až u konkrétních příkladů. I v tomto případě tedy můžeme využít již popsaný algoritmus 4.4 pro výpočet osy rotační válcové plochy, protože i zde zůstává princip minimalizace vícerozměrné funkce stejný. Do algoritmu 4.4 však vstupuje chybová funkce popsaná v algoritmu 4.5.

Newtonova metoda tečen popsaná v algoritmu 4.3, kterou algoritmus 4.4 využívá k jednorozměrné minimalizaci, zůstává stejná. Externí funkce λ*_error_function*, kterou využívá funkce *Newton_minimization* je o něco komplikovanější než u rotační válcové plochy. Pro přehlednost ji popišme v následujícím pseudokódu jako algoritmus 4.6. Je nutné si uvědomit, že v jednom kroku minimalizace metodou největšího spádu je chybová funkce funkcí jedné proměnné λ pro konkrétní hodnoty A_{STEP} a u_{STEP}.

Algoritmus 4.6: *VÝPOČET HODNOTY CHYBOVÉ FUNKCE ERROR_FUNCTION PRO OBECNOU ROTAČNÍ PLOCHU V BODĚ* λ. Jednorozměrná minimalizace je realizována Newtonovou metodou tečen, je tedy potřeba vyhodnocovat chybovou funkci *error_function* v jednom kroku minimalizace metodou největšího spádu v různých bodech λ. Vstupními parametry následující funkce jsou určující bod A_{STEP} a směrový vektor u_{STEP} aktuální přímky, hodnota λ a gradient funkce *error_function* pro hodnoty A_{STEP} a u_{STEP}, výstupem je reálná hodnota.

---

function λ*_error_function*$(\lambda, A_{STEP}, u_{STEP}, \nabla)$: **real**

1: *distance* $\leftarrow \varnothing$
2: $A \leftarrow A_{STEP} - \lambda \cdot \nabla[1:3]$
3: $u \leftarrow u_{STEP} - \lambda \cdot \nabla[4:\text{end}]$
4: *projection* $\leftarrow \varnothing$

5: $coords_x \leftarrow \emptyset$
6: $coords_y \leftarrow \emptyset$
7: $rand\_points \leftarrow \emptyset$
8: $surface\_count \leftarrow 0$
9: $unit\_vector \leftarrow u / \|u\|$
10: **for** $i = 1, 2, ..., n$ **do**
11: $projection[i] \leftarrow A + (unit\_vector \cdot (X[i] - A))unit\_vector$
%{výpočet ortogonálního průmětu bodu na aktuální přímku odvozen v oddíle o ortogonálním prokládání přímkou}
12: $distance[i] \leftarrow \|X[i] - projection[i]\|$
13: **let** $t \in \mathbb{R}$ **such that** $projection[i] = A + t \cdot unit\_vector$
14: $coords_x[i] \leftarrow t$
15: $coords_y[i] \leftarrow \|X[i] - projection[i]\|$
16: $x_{min} \leftarrow min(coords_x)$
17: $x_{max} \leftarrow max(coords_x)$
18: $y_{min} \leftarrow min(coords_y)$
19: $y_{max} \leftarrow max(coords_y)$
20: **for** $j = 1, 2, ..., accuracy$ **do**
21: $rand\_points_x \leftarrow (x_{max} - x_{min}) \cdot rand + x_{min}$
%{náhodné číslo z intervalu (x_{min}, x_{max}) }%
22: $rand\_points_y \leftarrow (y_{max} - y_{min}) \cdot rand + y_{min}$
%{náhodné číslo z intervalu (y_{min}, y_{max}) }%
23: **for** $j = 1, 2, ..., accuracy$ **do**
24: $inside \leftarrow False$
25: **for** $i = 1, 2, ..., n$ **do**
26: **if** $\|(coords_x[i], coords_y[i]) - (rand\_points_x[j], rand\_points_y[j])\| < \varepsilon$ **then**
27: $inside \leftarrow True$
28: **break**
29: **if** *inside* **then**
30: $surface\_count \leftarrow surface\_count + 1$
31: $surface\_area \leftarrow (x_{max} - x_{min}) \cdot (y_{max} - y_{min})$
32: $surface\_ratio \leftarrow surface\_count / accuracy$
33: $error_\lambda \leftarrow surface\_ratio \cdot surface\_area$
34: **return** *error*

Shrňme v následujícím přehledu jednotlivé kroky navrženého postupu hledání osy obecné rotační plochy (platné pro libovolnou minimalizační techniku).

- **Volba libovolné počáteční polohy přímky pro optimalizaci**

- **Opakujeme**
  - výpočet vektoru *distance*, jehož složky jsou vzdálenosti bodů $\{X_i\}_{i=1}^n$ vstupní množiny od aktuální přímky

- zavedení nové soustavy souřadnic v rovině, ve které aktuální přímka s pravoúhlými průměty $\{X_i\}_{i=1}^n$ bodů vstupní množiny splývá s osou x (počátek je určující bod aktuální přímky), funkční hodnoty v těchto bodech odpovídají vzdálenostem bodů $\{X_i\}_{i=1}^n$ vstupní množiny od aktuální přímky
- výpočet hodnoty chybové funkce *error_function* pro aktuální přímku jako obsahu množiny sjednocení všech okolí obrazů $\{\bar{X}_i\}_{i=1}^n$, tj. $\bigcup_{i=1}^n B_\varepsilon(\bar{X}_i)$
- určení nové polohy optimalizované přímky na základě minimalizace chybové funkce

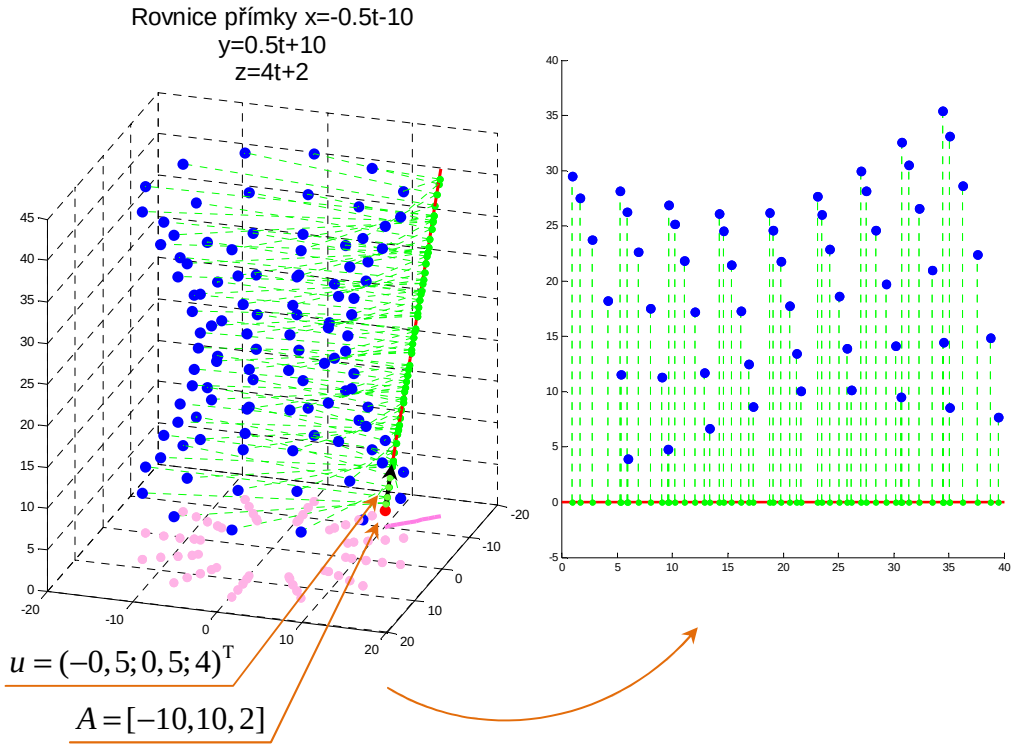
- **Odhad osy obecné rotační plochy a její vykreslení**

Funkčnost popsaných algoritmů demonstrujme na příkladech několika počítačově generovaných bodových množin. Zkoumat budeme plochy nízkých stupňů, především kvadriky, i obecnější rotační plochy vzniklé rotací skupiny elementárních křivek. O jaký typ rotační plochy se jedná, vždy upřesníme u konkrétního příkladu. Minimalizaci chybové funkce budeme provádět především modifikovanou metodou největšího spádu, neboť je tato metoda stabilnější.

Příklad 4.4: *HLEDÁNÍ OSY OBECNÉ ROTAČNÍ PLOCHY*. Navržené algoritmy předvedeme na příkladech experimentálních bodových množin. Vždy předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, které leží na povrchu rotační plochy, dále upřesníme, o jaký typ rotační plochy se jedná.

V následujícím vzorovém případě se jedná o body, které leží na povrchu části rotačního jednodílného hyperboloidu. Body jsou v tomto případě rozloženy pravidelně, jak vidíme na obrázku 4.52, zobrazeny jsou i půdorysy bodů. Rotační plocha je volena ve speciální poloze, její osa je souřadnicová osa z, abychom mohli lépe kontrolovat správnost našeho řešení. Na obrázku 4.52 je rovněž zakreslena počáteční poloha přímky (a její půdorys), kterou postupně optimalizujeme, a předloženy jsou její parametrické rovnice. Dále jsou na obrázku vyznačeny vzdálenosti bodů vstupní množiny od této přímky. Na témže obrázku jsou vzdálenosti bodů vstupní množiny od počáteční polohy přímky zakresleny v nové soustavě souřadnic v rovině.

Osu rotační plochy hledáme nejprve modifikovanou metodou největšího spádu. Obrázek 4.53 ilustruje postupné vylepšování polohy přímky, přičemž kvůli přehlednosti se přímka vykresluje pouze v několika pozicích. Zvlášť je vykreslena výsledná poloha osy a vypsány jsou její parametrické rovnice, viz obrázek 4.54. Vykreslena je také výsledná situace v soustavě souřadnic v rovině. Dostatečně dobrý odhad osy rotační plochy dostáváme pro uvedenou volbu vstupních parametrů metody (2000 iterací). Pokud bychom vyžadovali přesnější výsledek, je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroku λ. Obrázek 4.55 ukazuje rovněž situaci pro několik kroků v soustavě souřadnic v rovině.



Obrázek 4.52: Body pravidelně rozmístěné na části rotačního jednodílného hyperboloidu, počáteční poloha přímky, pravoúhlé průměty bodů plochy na tuto přímku a zakreslení vzdáleností od přímky do nové soustavy souřadnic v rovině

Ta samá vstupní množina bodů je zpracována také obecnou metodou největšího spádu. Počáteční polohu přímky volíme stejně jako v metodě předchozí, viz obrázek 4.52. Postupné vylepšování polohy přímky znázorňuje obrázek 4.56. Opět přímku vykreslujeme jen v několika pozicích. Vypisujeme také vstupní parametry, které určují nastavení minimalizace metodou největšího spádu a parametry pro jednorozměrnou minimalizaci Newtonovou metodou tečen. Výsledný odhad osy rotační plochy je pro přehlednost vykreslen zvlášť, tento odhad získáme po 500 iteracích. Vypisujeme též parametrické rovnice odhadu osy, viz obrázek 4.57.

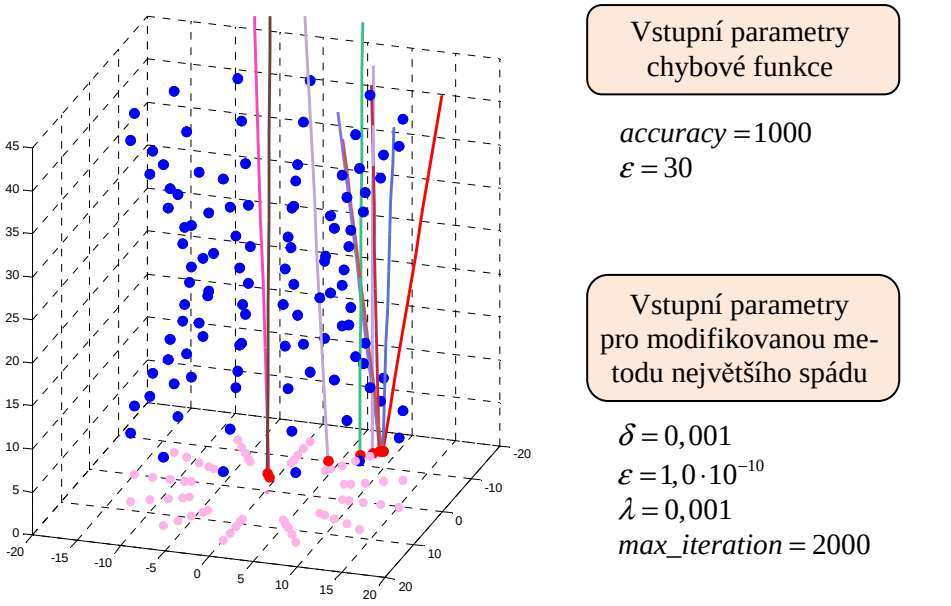
Ve většině případů použití obecné metody největšího spádu ale nedosahujeme takto uspokojivého odhadu osy, což je způsobeno nestabilitou výpočtu chybové funkce. Proto je také pro obecné rotační plochy náročné vypozorovat správné hodnoty vstupních parametrů obecné metody největšího spádu. Může se dokonce stát, že taková optimalizace není vůbec možná. Sama metoda Monte Carlo je nestabilní kvůli volbě ε-okolí a náhodně generovaným bodům. Proto i minimalizace chybové funkce se může často chovat nestabilně. V takových případech se při hledání osy rotační plochy rozhodujeme pro modifikovanou metodu největšího spádu. Výpočet minimalizace obecnou metodou největšího spádu uvádíme v tomto případě pro úplnost srovnání.

Pro obě metody minimalizace chybové funkce jsou vykresleny také závislosti hodnoty chybové funkce na počtu iterací, viz obrázek 4.58. V obou případech jsou hodnoty vypsány pro prvních 500 kroků algoritmu. Z grafů konvergence hodnoty chybové funkce k hodnotě minimální je zřejmé, že obecná metoda největšího spádu se

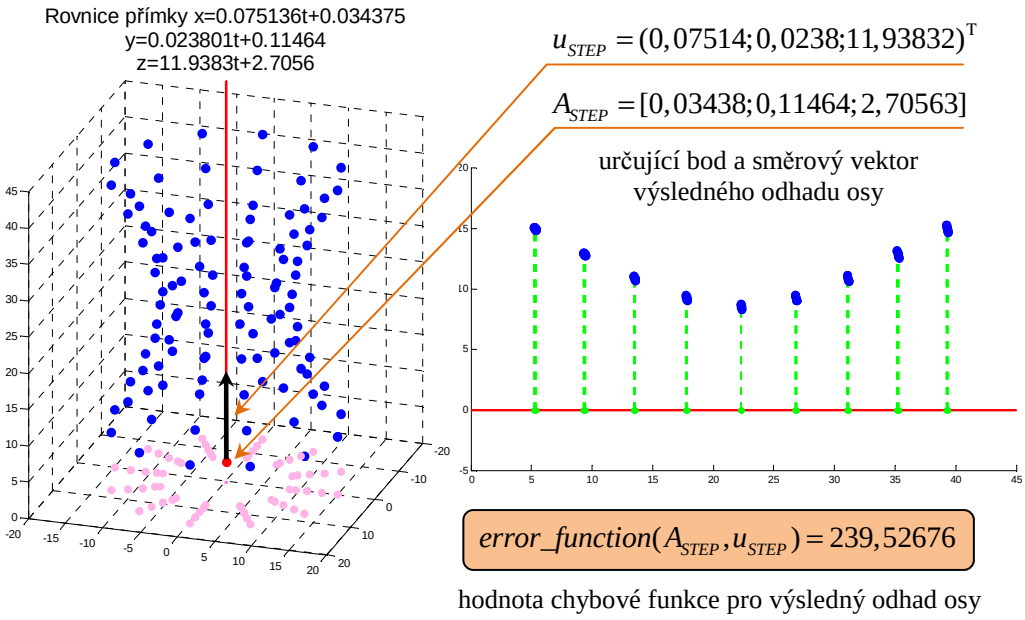
chová výrazně nestabilněji. Modifikovaná metoda je jednodušší, lze lépe kontrolovat její průběh a navíc je její chování stabilnější.

Programy z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

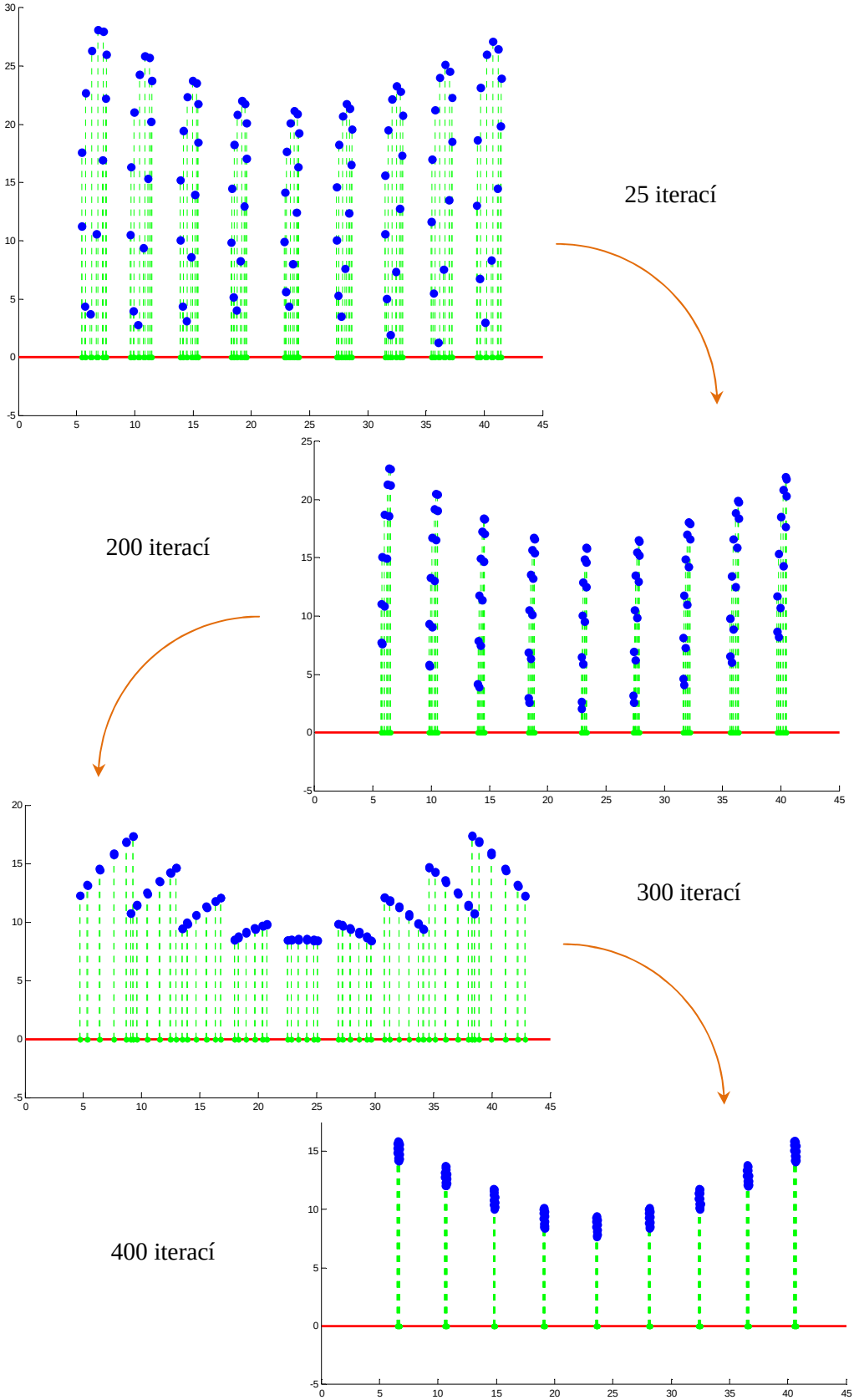
`osa_rot_plochy_spad_modifik.m` (programy/osa_rot_plochy),
`osa_rot_plochy_spad.m` (programy/osa_rot_plochy).



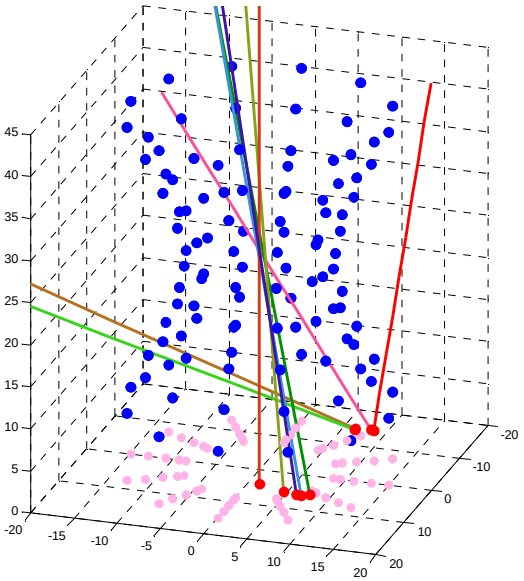
Obrázek 4.53: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



Obrázek 4.54: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – modifikovaná metoda největšího spádu



Obrázek 4.55: Postupná modifikace situace v soustavě souřadnic v rovině – zmenšování obsahu oblasti, kterou body vyplňují



Vstupní parametry pro Newtonovu metodu tečen

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-4}$
$\lambda^{LB} = -1$
$\lambda^{UB} = 1$
$max\_iteration = 500$

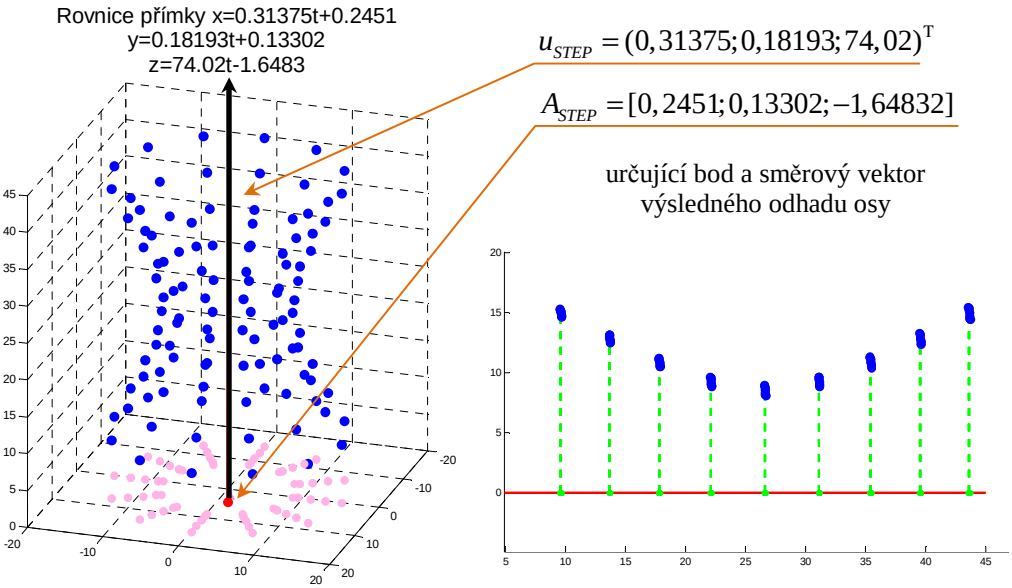
Vstupní parametry pro metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$a = 0,5$
$b = 0,25$
$c = 50$
$max\_iteration = 500$

Vstupní parametry chybové funkce

$accuracy = 1000$
$\varepsilon = 30$

Obrázek 4.56: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce obecnou metodou největšího spádu



$u_{STEP} = (0,31375; 0,18193; 74,02)^{T}$

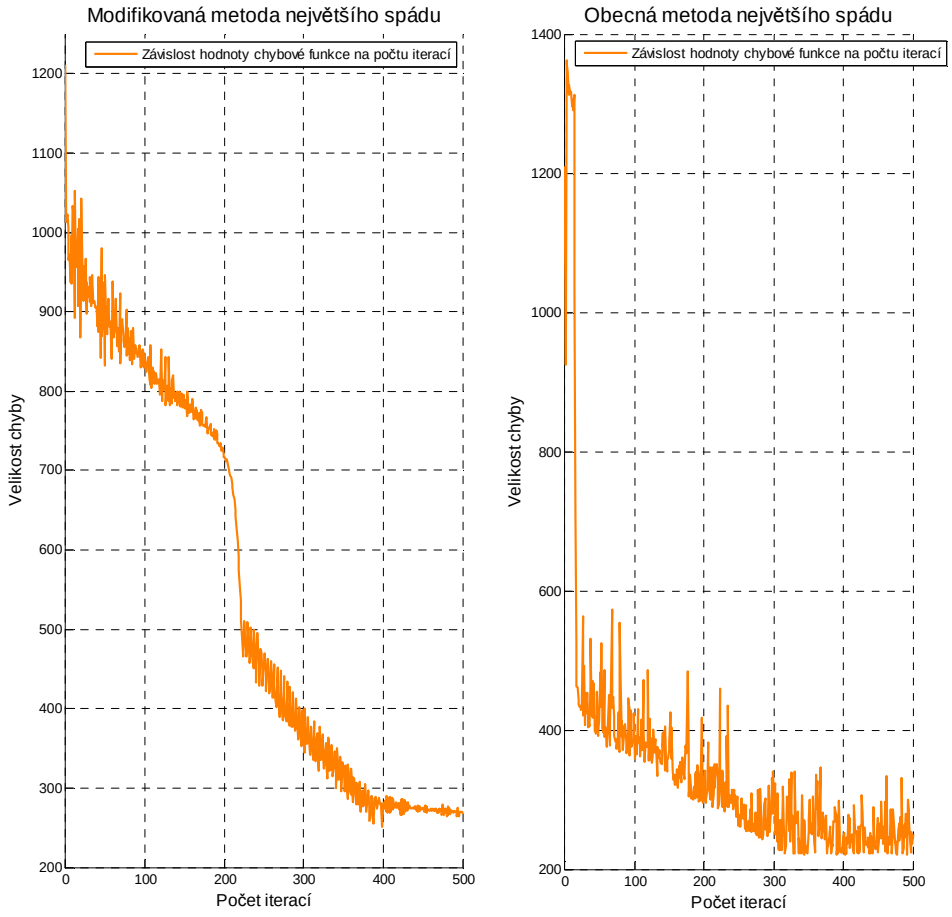
$A_{STEP} = [0,2451; 0,13302; -1,64832]$

určující bod a směrový vektor výsledného odhadu osy

$error\_function(A_{STEP}, u_{STEP}) = 250,5032$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.57: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – obecná metoda největšího spádu



Obrázek 4.58: Minimalizace chybové funkce modifikovanou a obecnou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

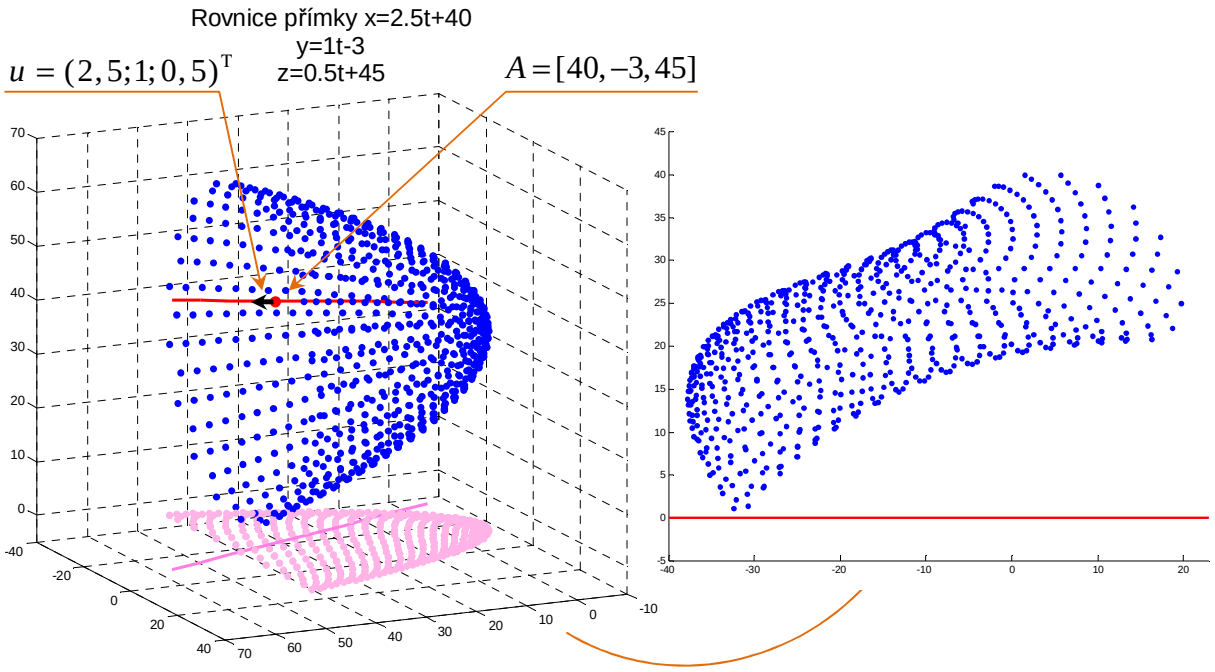
4.3.1 Případová studie v eukleidovském prostoru E_3

Zaměřme se na zpracování několika vstupních bodových množin v eukleidovském prostoru E_3. Tentokrát budeme provádět výpočty pomocí modifikované metody největšího spádu, neboť je tato metoda stabilnější, a dále na základě experimentů popíšeme další možný postup při minimalizaci chybové funkce. Cílem je prověřit fungování navrženého postupu hledání osy obecné rotační plochy. Nebudeme již uvádět jednotlivé výpočty minimalizace, předkládáme pouze hodnoty vstupních parametrů minimalizační metody a vývoj hledání osy rotačních ploch prezentujeme vždy skupinami obrázků. Číselné údaje a výsledky výpočtů lze získat spuštěním programů vytvořených ve výpočetním prostředí MATLAB s názvy `osa_rot_plochy_spad_modifik.m` a `osa_rot_plochy_spad.m`, kde lze měnit vstupní bodové množiny. Počítačově generované množiny, které v této studii testujeme, jsou k dispozici na přiloženém médiu pod názvy `body01.txt`, `body02.txt`, … (cesta: bodove_mnoziny/osa_rot_plochy).

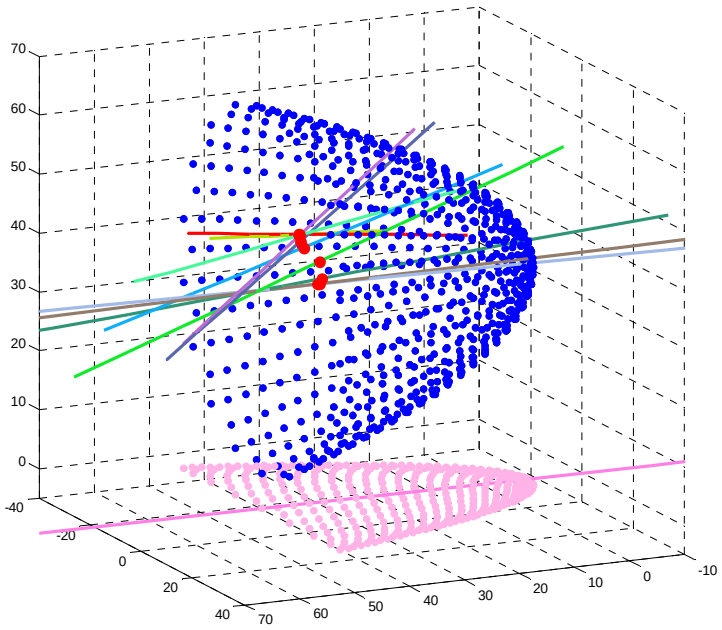
Ve všech případech vždy předpokládejme, že je dána konečná množina bodů, které leží na části povrchu nějaké rotační plochy. Kvůli lepšímu posouzení kvality výsledného odhadu volíme osy rotačních ploch, ze kterých body získáváme, ve speciálních polohách, tj. osy rotační plochy splývají s některou ze souřadnicových os. Neuvažujeme nepřesnosti v měření, body jsou tedy umístěny na rotační ploše přesně. Rozlišujeme případy, kdy se jedná o pravidelné a nepravidelné množiny a případy, jak velká část rotační plochy je navzorkována. Poznamenejme rovněž, že v této případové studii volíme vstupní bodové množiny hustší, abychom se více přiblížili množinám reprezentující reálné povrchy.

Regulární bodová množina – případ (a)

V první úloze uvažujme množinu bodů, které jsou rozmístěny pravidelně na části rotačního paraboloidu. Polohu osy rotační plochy, ze které byly body navzorkovány, volíme totožnou s osou *x*. Vstupní bodová množina a zvolená počáteční poloha přímky, kterou optimalizujeme, jsou zakresleny na obrázku 4.59. Uvedeny jsou také parametrické rovnice přímky a znázorněna je situace v soustavě souřadnic v rovině. Na obrázku 4.60 je zaznamenáno postupné vylepšování polohy přímky, přičemž vykreslovány jsou pouze některé polohy. Výsledný odhad osy ukazuje obrázek 4.61. Pro lepší názornost přidáváme do prostorových obrázků také půdorysy bodů vstupní množiny a půdorysy počáteční a výsledné polohy optimalizované přímky. Obdobně jako v příkladě 4.4 předkládáme situaci v soustavě souřadnic v rovině pro několik kroků iterační metody, viz obrázek 4.62, a průběh konvergence minimalizační techniky, viz obrázek 4.63.



Obrázek 4.59: Body pravidelně rozmístěné na části rotačního paraboloidu, počáteční poloha přímky a zakreslení vzdáleností od přímky do nové soustavy souřadnic v rovině



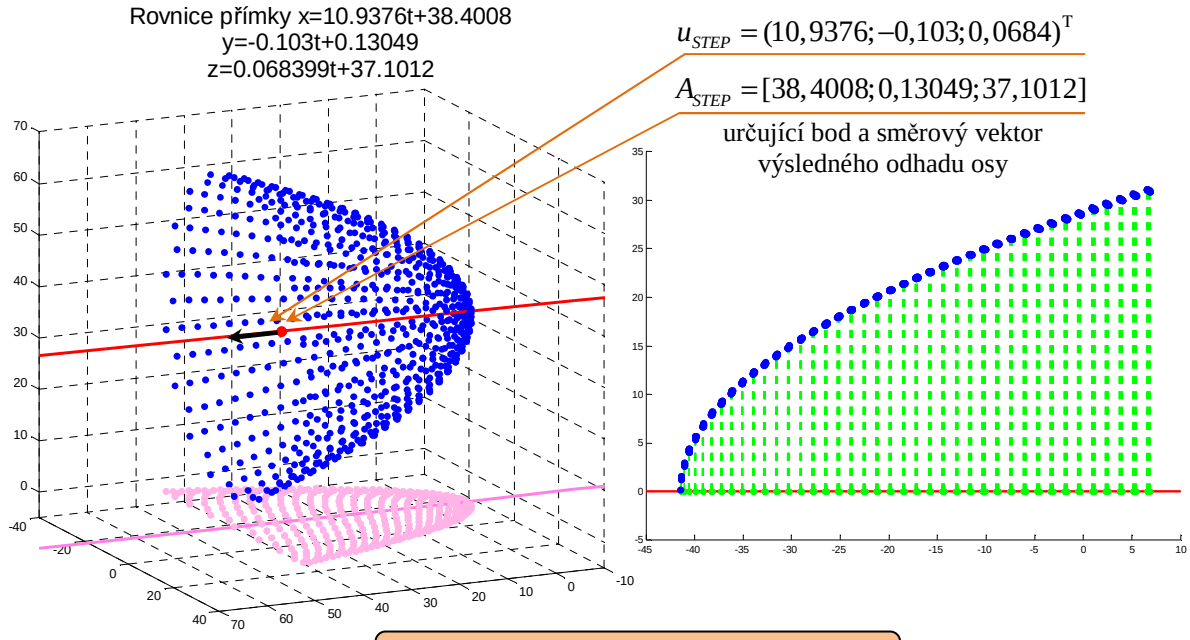
Vstupní parametry chybové funkce

$accuracy = 4000$
$\varepsilon = 30$

Vstupní parametry pro modifikovanou metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,001$
$max\_iteration = 1500$

Obrázek 4.60: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



$u_{STEP} = (10,9376; -0,103; 0,0684)^T$

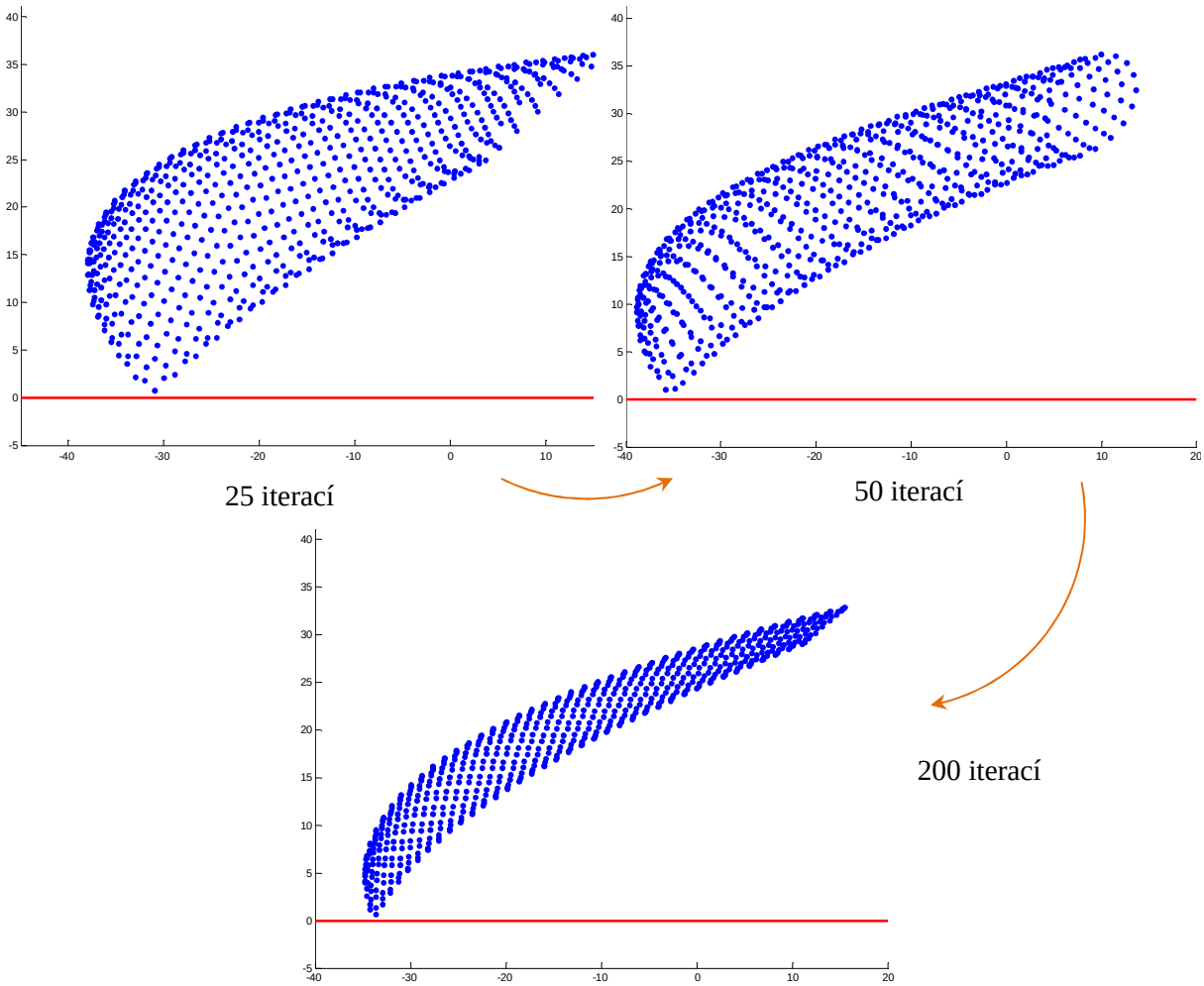
$A_{STEP} = [38,4008; 0,13049; 37,1012]$

určující bod a směrový vektor výsledného odhadu osy

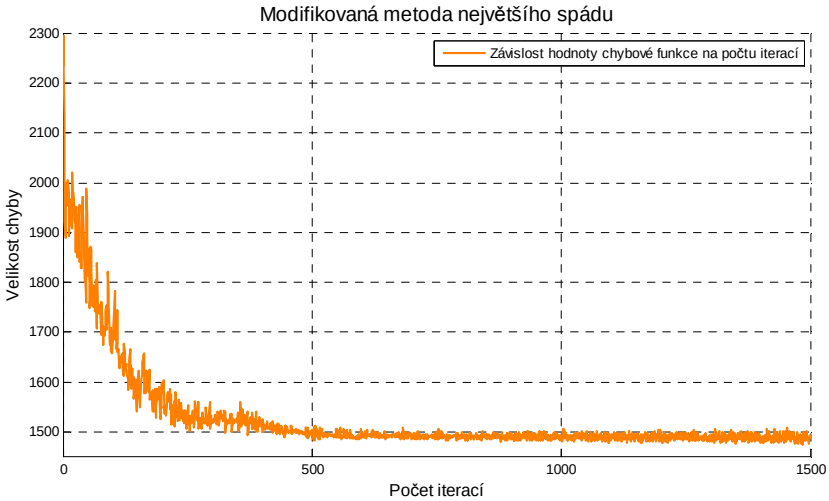
$error\_function(A_{STEP}, u_{STEP}) = 1497,19618$

hodnota chybové funkce pro výsledný odhad osy

Obrázek 4.61: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – modifikovaná metoda největšího spádu

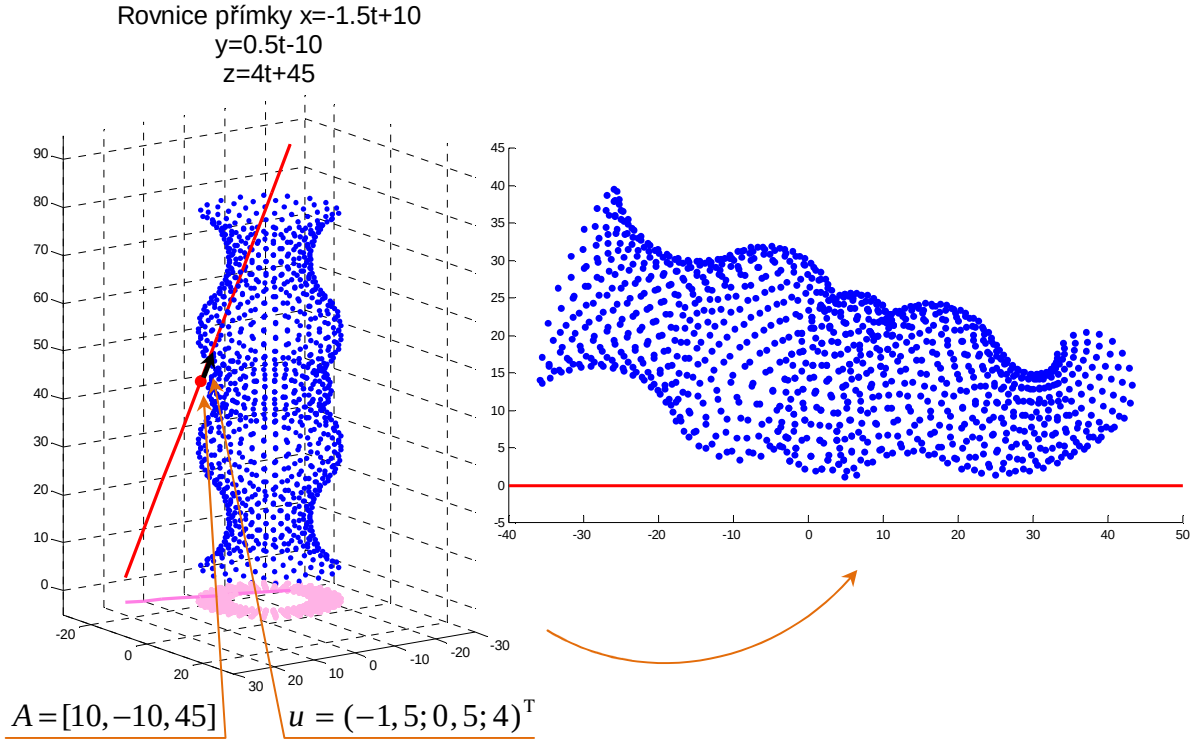


Obrázek 4.62: Postupná modifikace situace v soustavě souřadnic v rovině – zmenšování obsahu oblasti, kterou body vyplňují

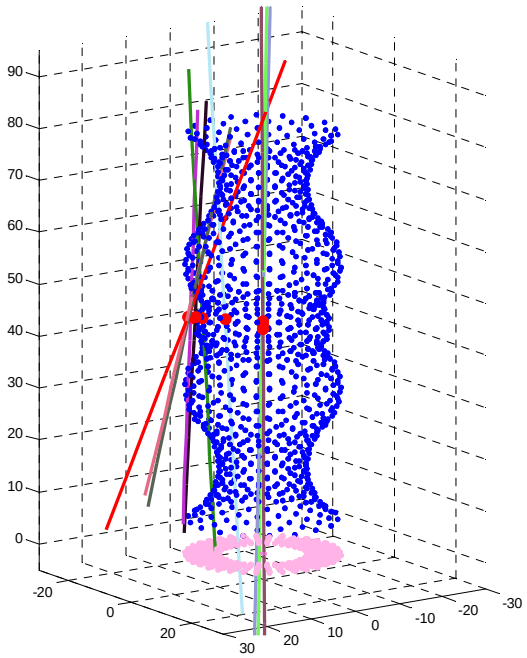


Obrázek 4.63: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

Regulární bodová množina – případ (b)



Obrázek 4.64: Body pravidelně rozmístěné na části obecné rotační plochy, počáteční poloha přímky a zakreslení vzdáleností od přímky do nové soustavy souřadnic v rovině



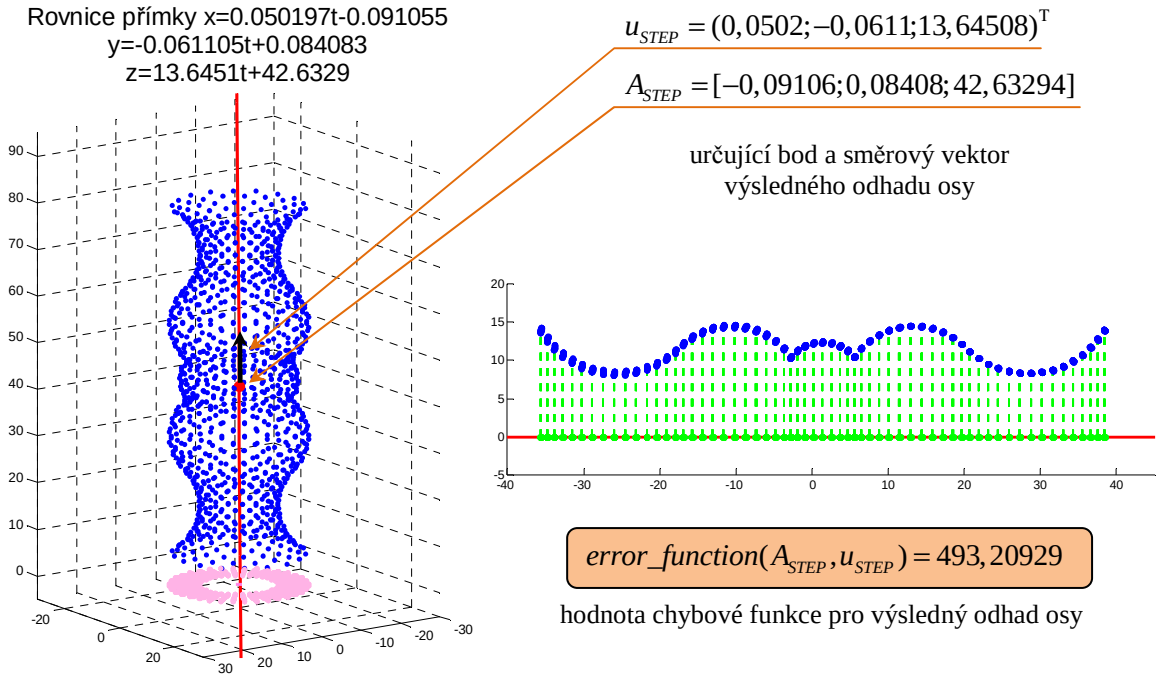
Vstupní parametry chybové funkce

$accuracy = 4000$
$\varepsilon = 30$

Vstupní parametry pro modifikovanou metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,001$
$max\_iteration = 1500$

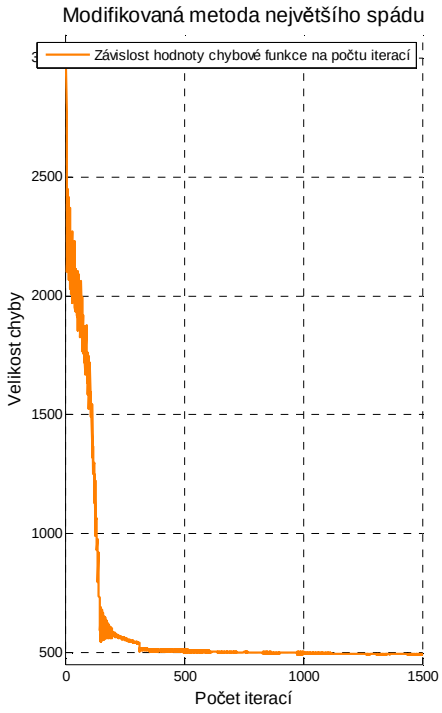
Obrázek 4.65: Vstupni parametry minimalizačni metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



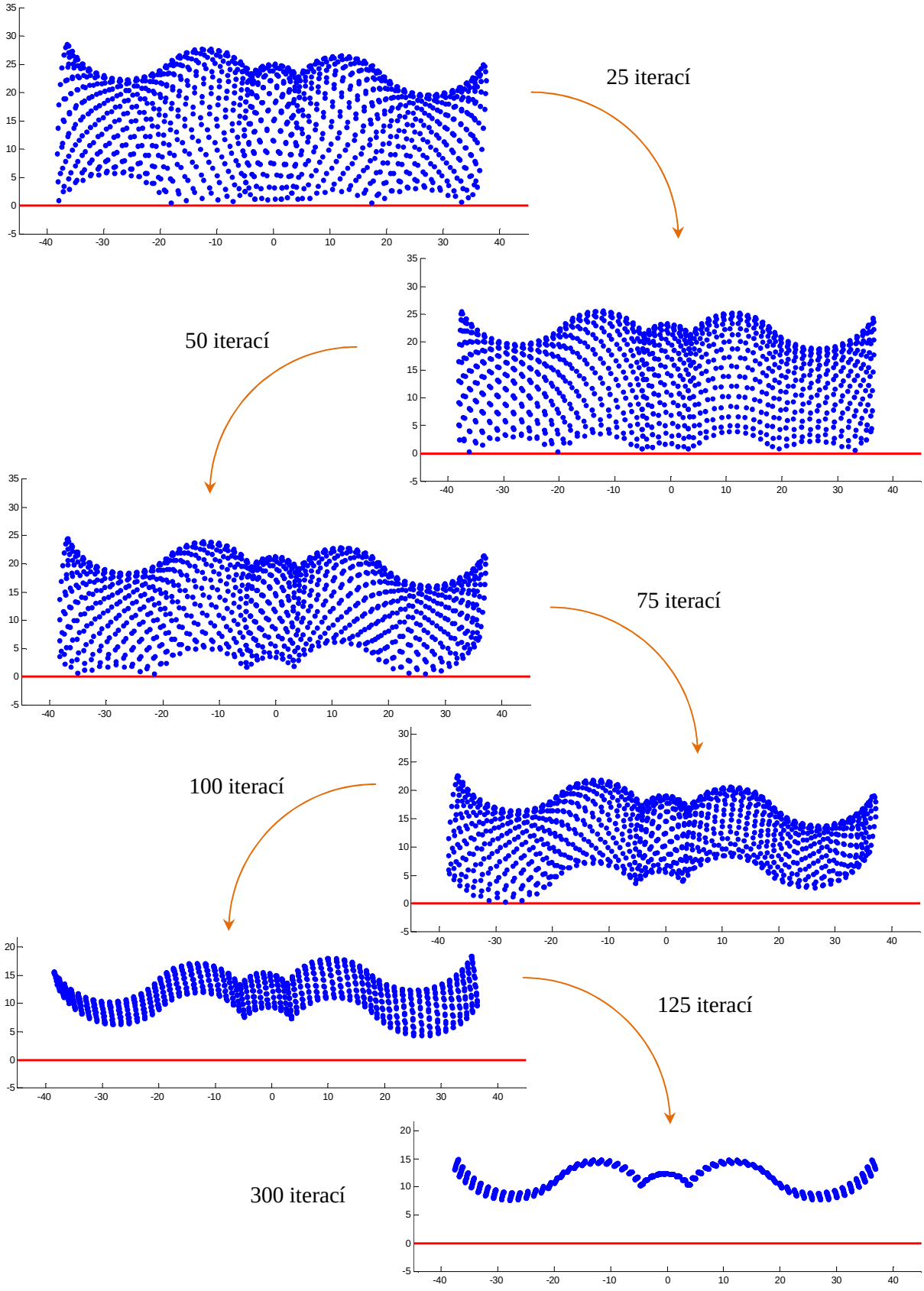
Obrázek 4.66: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – modifikovaná metoda největšího spádu

V dalším příkladě se věnujme množině bodů, které jsou navzorkovány na obecnější rotační ploše. Rotační plochu, ze které jsme získali body pro testování, jsme určili pomocí polomeridiánu, který je složen z oblouků kružnic, osu rotační plochy jsme zvolili v ose *z*. Speciální polohu rotační plochy volíme opět z toho důvodu, abychom mohli lépe posoudit přesnost výsledného odhadu osy. Body na rotační ploše tvoří pravidelnou síť, viz obrázek 4.64. Na témže obrázku je zakreslena počáteční poloha přímky a zapsány jsou její parametrické rovnice. Pro počáteční polohu přímky můžeme na obrázku 4.64 pozorovat situaci v soustavě souřadnic v rovině.

Obrázky 4.65-4.68 ilustrují postupný výpočet odhadu osy rotační plochy modifikovanou metodou největšího spádu, výstupy metody jsou tedy předloženy obdobnou formou jako v předchozím případě. Situace v soustavě souřadnic v rovině v průběhu metody je velice zajímavá, proto v tomto případě vykreslujeme více kroků. Pro lepší názornost přidáváme do prostorových obrázků i půdorysy bodů vstupní množiny a půdorysy počáteční a výsledné polohy optimalizované přímky.

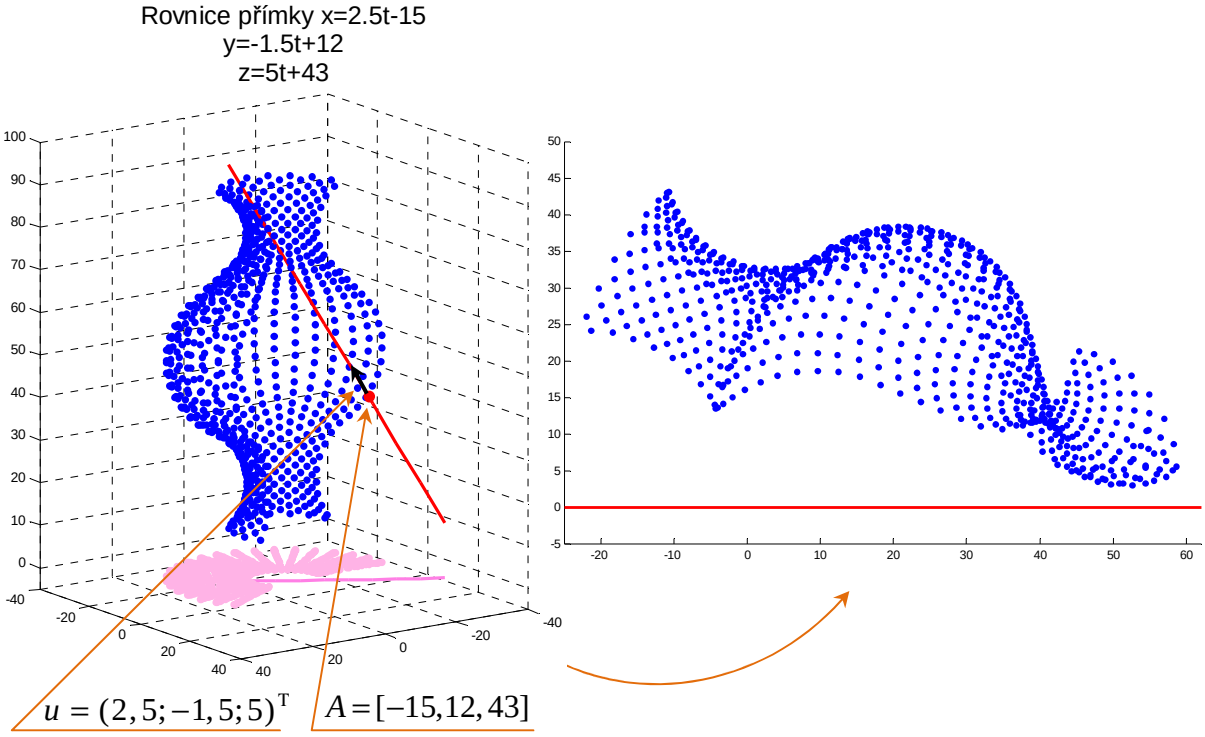


Obrázek 4.67: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

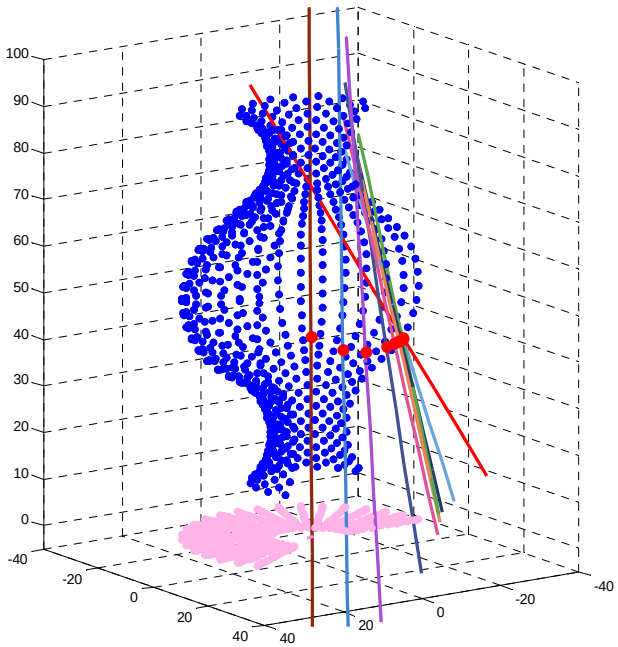


Obrázek 4.68: Postupná modifikace situace v soustavě souřadnic v rovině – zmenšování obsahu oblasti, kterou body vyplňují

Regulární bodová množina – případ (c)



Obrázek 4.69: Body pravidelně rozmístěné na části obecné rotační plochy, počáteční poloha přímky a zakreslení vzdáleností od přímky do nové soustavy souřadnic v rovině



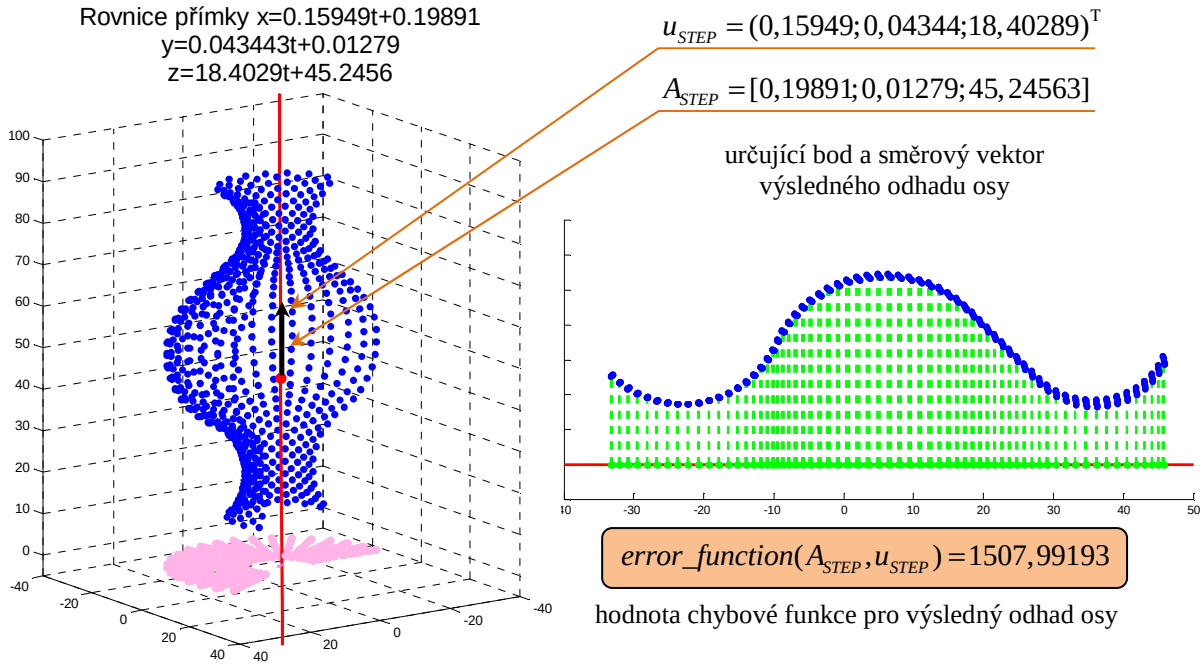
Vstupní parametry chybové funkce

$accuracy = 5000$
$\varepsilon = 35$

Vstupní parametry pro modifikovanou metodu největšího spádu

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,002$
$max\_iteration = 1000$

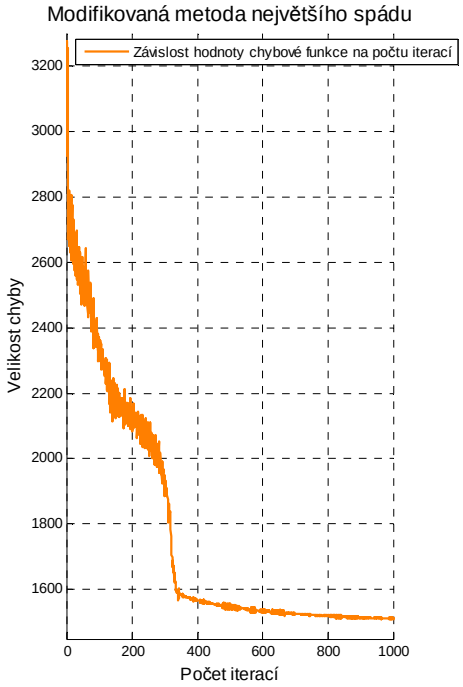
Obrázek 4.70: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



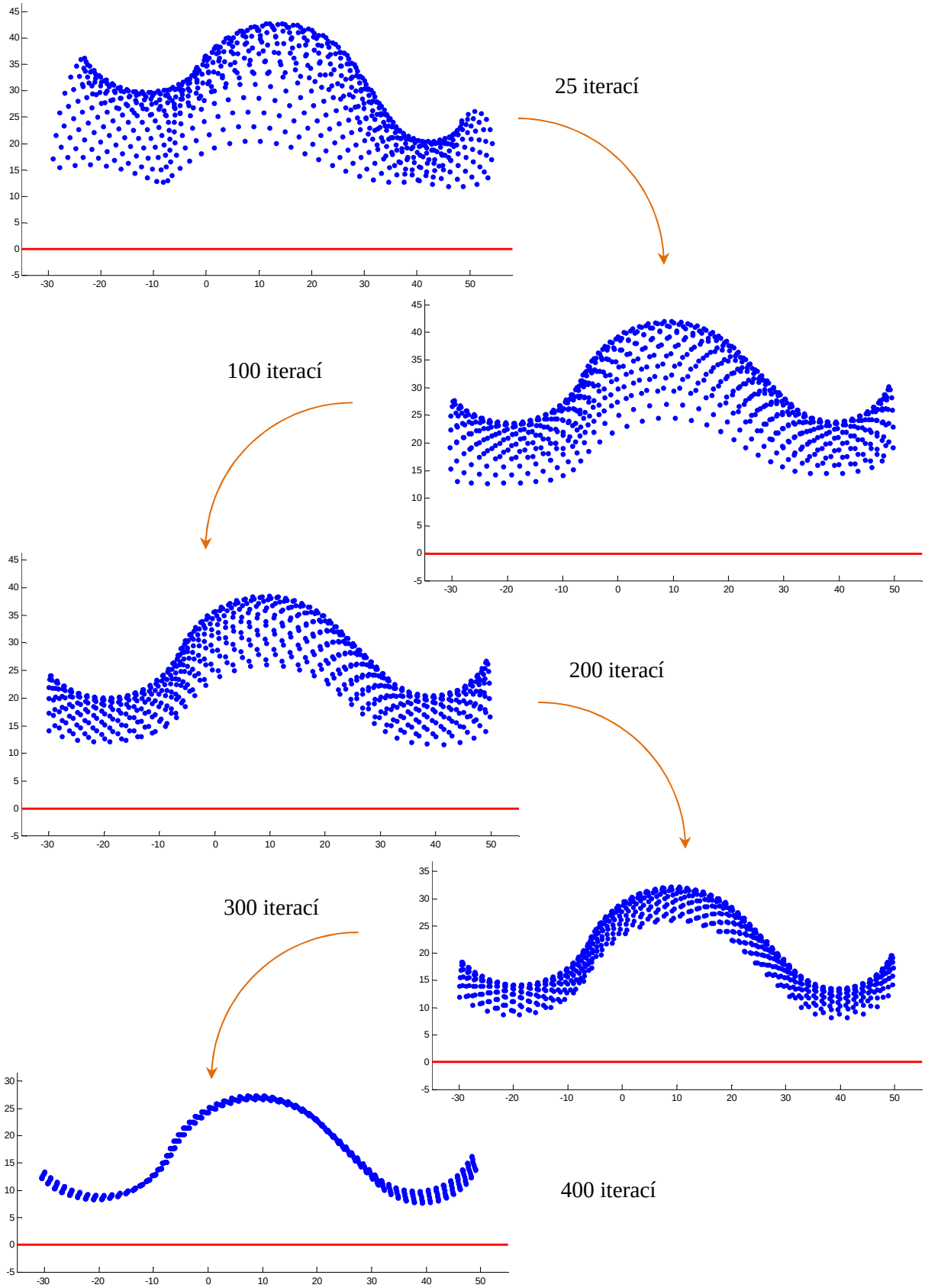
Obrázek 4.71: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – modifikovaná metoda největšího spádu

Další bodová množina je rovněž navzorkována na části obecnější rotační plochy, kterou jsme určili rotací polomeridiánu, jenž je složen z oblouků kružnic. Osu rotační plochy jsme zvolili opět v ose z. Rotace polomeridiánu kolem osy je provedena o úhel menší než $360°$. Body na rotační ploše tvoří pravidelnou síť, jak vidíme na obrázku 4.69. Na témže obrázku je zakreslena počáteční poloha přímky a zapsány jsou její parametrické rovnice. Pro počáteční polohu přímky můžeme na obrázku pozorovat situaci v soustavě souřadnic v rovině.

Postupný výpočet odhadu osy rotační plochy modifikovanou metodou největšího spádu je znázorněn sadou obrázků 4.70-4.73. Vykreslujeme také více kroků pro situaci v soustavě souřadnic v rovině. Do obrázků rovněž přidáváme půdorysy bodů vstupní množiny a půdorysy počáteční a výsledné polohy optimalizované přímky.

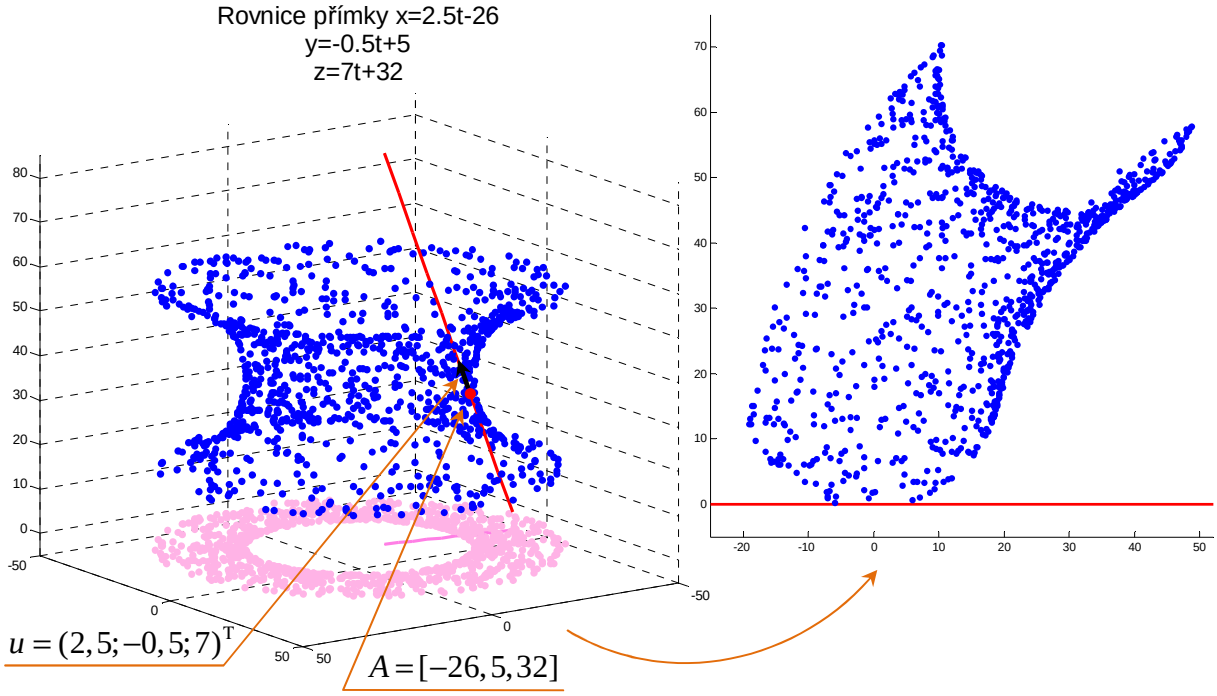


Obrázek 4.72: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

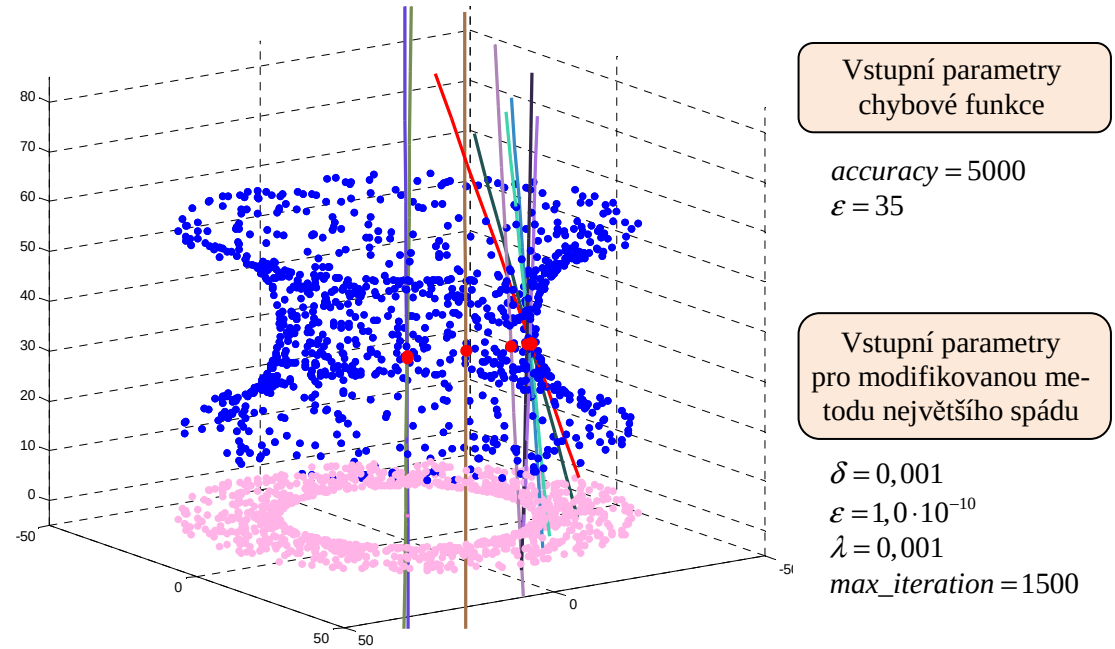


Obrázek 4.73: Postupná modifikace situace v soustavě souřadnic v rovině – zmenšování obsahu oblasti, kterou body vyplňují

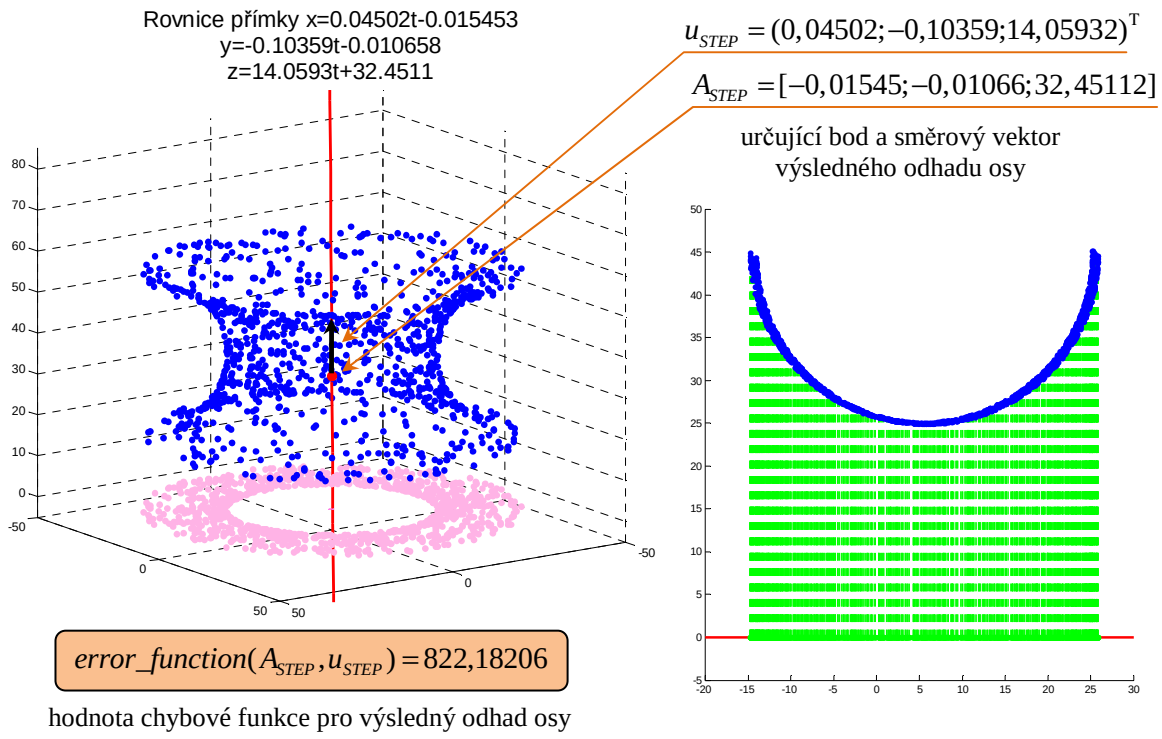
Náhodná bodová množina



Obrázek 4.74: Body náhodně rozmístěné na části anuloidu, počáteční poloha přímky a zakreslení vzdáleností od přímky do nové soustavy souřadnic v rovině



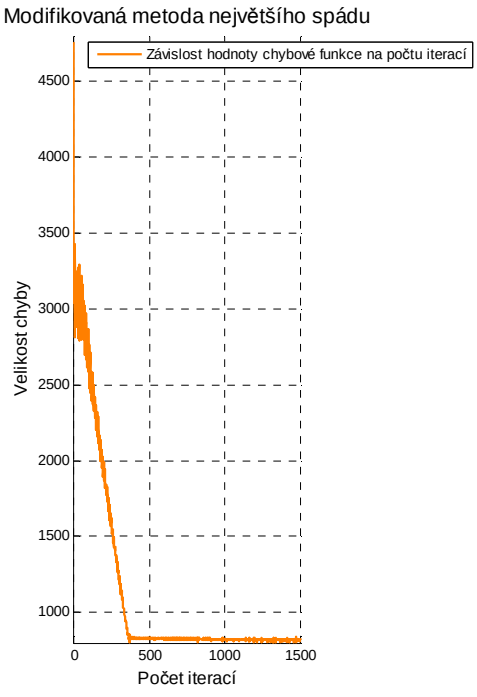
Obrázek 4.75: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



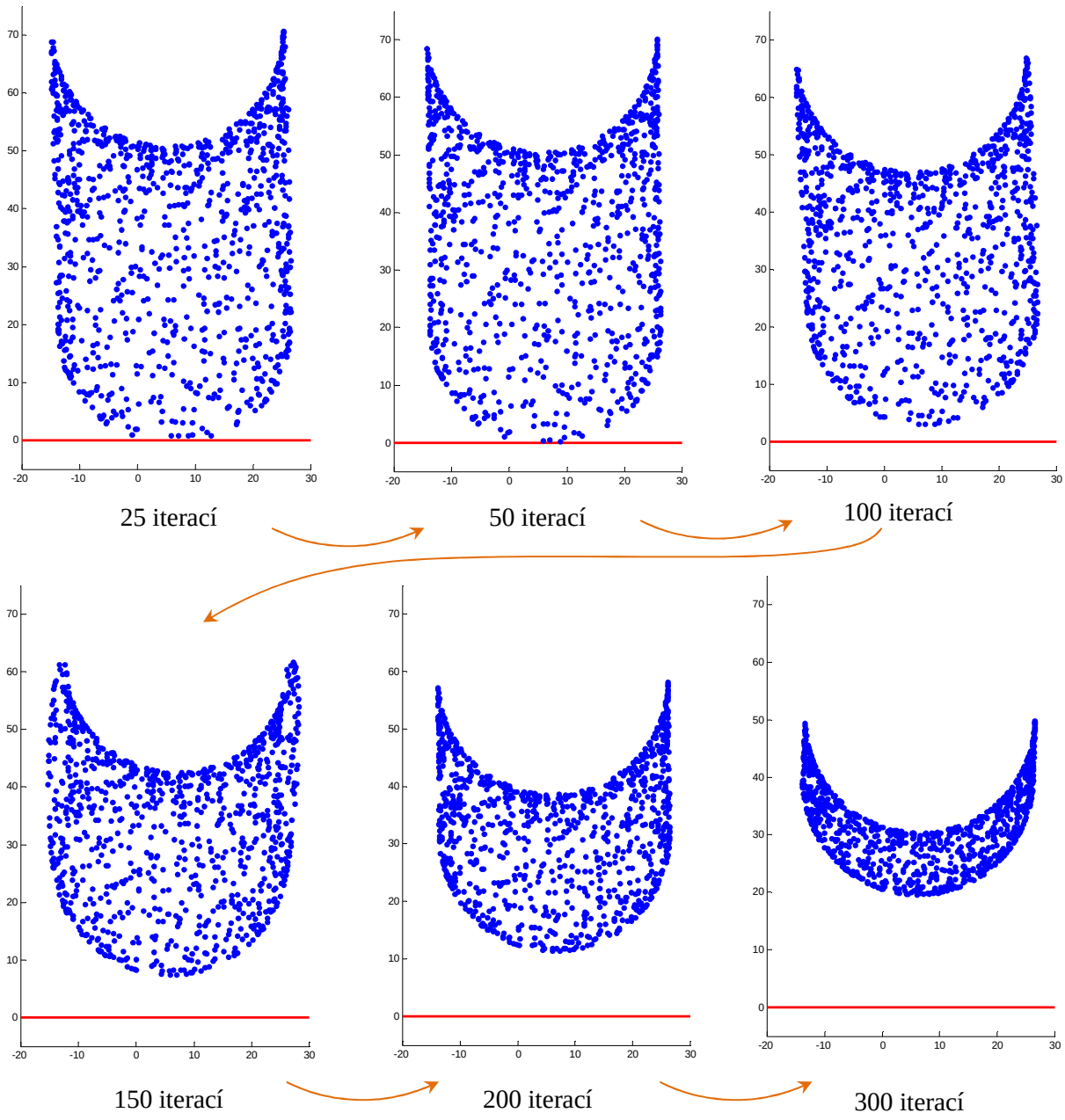
Obrázek 4.76: Výsledný odhad osy a situace v pomocné soustavě souřadnic v rovině pro výsledný odhad osy – modifikovaná metoda největšího spádu

Poslední bodová množina, kterou budeme v této případové studii zkoumat, je navzorkována náhodně na části *anuloidu* (Surynková, 2008a). Polomeridiánem anuloidu je kružnice, jejíž střed neleží na ose rotační plochy. V tomto případě jsme uvažovali půlkružnici, jejíž rotací vzniká pouze část anuloidu. Osu rotační plochy jsme zvolili opět v ose z. Vstupní množinu bodů a počáteční polohu přímky, kterou optimalizujeme, vidíme na obrázku 4.74. Předloženy jsou parametrické rovnice počáteční polohy přímky a situace v soustavě souřadnic v rovině.

Postupný výpočet odhadu osy rotační plochy modifikovanou metodou největšího spádu znázorňují obrázky 4.75-4.78. Vykreslujeme také situaci v soustavě souřadnic v rovině pro několik kroků algoritmu. Do obrázků rovněž přidáváme půdorysy bodů vstupní množiny a půdorysy počáteční a výsledné polohy optimalizované přímky.



Obrázek 4.77: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací



Obrázek 4.78: Postupná modifikace situace v soustavě souřadnic v rovině – zmenšování obsahu oblasti, kterou body vyplňují

Při zkoumání bodových množin jsme někdy narazili na problém jak správně určit vhodnou počáteční polohu přímky, která se bude modifikovat. Na volbě počáteční polohy závisí dobrý průběh minimalizační metody. Při použití našeho postupu pro hledání osy rotační plochy modifikovanou metodou největšího spádu proto doporučujeme volit počáteční polohu přímky v blízkosti očekávaného výsledku. Takovou volbu lze snadno provést na základě vizualizace dat.

V případě větší nestability průběhu minimalizace modifikovanou metodou největšího spádu navrhujeme ještě alternativní postup. Na základě mnoha experimentů, z nichž jen něko-

lik předkládáme v této případové studii, jsme se rozhodli zavést další minimalizační postup pro chybovou funkci popsanou v algoritmu 4.5. Uvědomme si, že metoda největšího spádu je založena na diferencování chybové funkce z algoritmu 4.5, ovšem při výpočtu chyby metodou Monte Carlo, kterou v algoritmu 4.5 používáme, lze chybovou funkci diferencovat pouze přibližně. Lze proto očekávat, že konvergence optimalizační metody, nemusí být vždy uspokojivá. Tato hypotéza se skutečně potvrdila testováním výpočtů na mnoha bodových množinách. Proto navrhujeme další metodu, která se neopírá o diferencování chybové funkce, nýbrž o její hodnoty. Navržená metoda je odvozená ze známé metody *hill climbing* – metoda stoupání do kopce (Rusell a Norvig, 2003).

Vysvětleme stručně princip tohoto postupu minimalizace chybové funkce, který původně vznikl v informatice. Opět se jedná o iterační algoritmus, kdy v prvním kroku volíme stejně jako u diferenciálních metod největšího spádu libovolně prvotní polohu přímky. V každém dalším kroku polohu přímky optimalizujeme tak, abychom se přiblížili minimu chybové funkce. Pro aktuální přímku vždy spočítáme několik jejích blízkých poloh a určíme, pro kterou z nich nejvíce klesne hodnota chybové funkce. Tuto přímku potom označíme za další aktuální polohu. Postup opakujeme, dokud není splněna předem daná cílová podmínka. V tomto případě, buď omezujeme maximální počet iterací, nebo výpočet ukončujeme, je-li rozdíl hodnot chybové funkce pro dvě sousední polohy přímky menší než vhodně zvolené malé $\varepsilon > 0$, $\varepsilon \in \mathbb{R}$.

Blízké polohy aktuální přímky určujeme tak, že k souřadnicím určujícího bodu a směrového vektoru aktuální přímky přičítáme náhodně generované malé hodnoty. Počet blízkých poloh pro aktuální přímku určujeme opět experimentálně, vhodná je například volba 100.

Výsledky použití tohoto algoritmu již z kapacitních důvodů neuvádíme, úspěšně jsme jej ale testovali na počítačově generovaných množinách.

Na přiloženém výměnném médiu jsou k dispozici další počítačově generované bodové množiny k testování algoritmů. Jedná se o již předvedené typy rotačních ploch s osami v jiných polohách i jiné typy rotačních ploch. Připojujeme také bodové množiny, které se více blíží reálným příkladům, připouštíme proto nepřesnosti v navzorkování rotačních ploch.

Na závěr této případové studie poznamenejme, že chybová funkce popsaná v algoritmu 4.5, je použitelná i pro hledání osy rotační válcové plochy. Jednoduše bychom také mohli chybovou funkci z algoritmu 4.5 upravit pro rovinné případy. Mohli bychom tak hledat osy speciálních bodových množin v rovině, ve kterých jsou body uvažovány na osově symetrické křivce či na křivkách, které jsou osově symetrické. Tyto speciální případy už neuvádíme, neboť je v dalších postupech nebudeme využívat.

4.4 Hledání rovinových symetrií

V tomto oddíle se budeme věnovat dalšímu typu analýzy bodové množiny v eukleidovském prostoru E_3 a to hledání rovinových symetrií. Zaveďme rovinovou symetrii formálně, i když intuitivně je tento pojem zřejmý.

Definice 4.5 (*Rovinová symetrie s rovinou* ω). Nechť je dána rovina ω v eukleidovském prostoru E_3. Potom zobrazení $f_\omega : E_3 \to E_3$, které každému bodu $A \in E_3$, $A \notin \omega$ přiřadí bod $f_\omega(A) \in E_3$ tak, že přímka $Af_\omega(A)$ je kolmá k rovině ω a střed úsečky $Af_\omega(A)$ leží v rovině ω a každému bodu $A \in E_3$, $A \in \omega$ přiřadí bod $f_\omega(A) = A$, nazveme *rovinovou symetrií s rovinou* ω. ■

Definice 4.6 (*Rovina symetrie*). Nechť jsou dány rovina ω a neprázdná množina $\mathcal{X}$ $n+1$ bodů $\{[x_i, y_i, z_i]\}_{i=0}^{n}$ v eukleidovském prostoru E_3. Řekneme, že množina bodů $\mathcal{X}$ je symetrická *podle roviny* ω, jestliže $f_\omega(\mathcal{X}) = \mathcal{X}$ a rovinu ω označíme jako *rovinu symetrie* bodové množiny $\mathcal{X}$. ■

Bodové množiny, které chceme rekonstruovat, jsou často rovinově symetrické podle jedné, dvou nebo tří navzájem kolmých rovin nebo nekonečně mnoha rovin se společnou průsečnicí (v případě bodových množin reprezentujících povrch rotačních ploch). V tomto oddíle se zaměříme na případy rovinových symetrií podle jedné, dvou a tří navzájem kolmých rovin.

Pro hledání rovin symetrií speciálních bodových množin navrhujeme iterační algoritmus vycházející z definic 4.5 a 4.6. Nalezené rovinové symetrie potom využíváme v dalších fázích rekonstrukce povrchů. Abychom mohli algoritmus pro hledání rovinových symetrií použít, musíme fakt, že je bodová množina symetrická podle nějaké roviny nebo rovin, znát. U analýzy bodových množin odpovídající reálným povrchům to předpokládáme na základě vlastností naskenovaného povrchu či z vizuálního posouzení bodové množiny.

4.4.1 Rovinová symetrie s jednou rovinou

Začněme nejjednodušším případem, kdy předpokládáme, že je bodová množina symetrická podle jedné roviny. Představme navržený postup hledání roviny symetrie této bodové množiny.

Mějme dánu neprázdnou množinu $\mathcal{X}$ n bodů v eukleidovském prostoru E_3, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Víme, že je tato vstupní bodová množina přesně symetrická podle jedné roviny.

Rovinu symetrie hledáme iteračně s použitím odvozené modifikované minimalizační techniky metody největšího spádu. V prvním kroku algoritmu určíme těžiště T vstupní bodové množiny $\mathcal{X}$, které spočítáme jako aritmetický průměr bodů $\{X_i\}_{i=1}^{n}$, tj.

$$T = \frac{1}{n}\sum_{i=1}^{n} X_i , \tag{4.11}$$

rozepsáno po souřadnicích

(4.12) $$T^x = \frac{1}{n}\sum_{i=1}^{n} X_i^x,\ T^y = \frac{1}{n}\sum_{i=1}^{n} X_i^y,\ T^z = \frac{1}{n}\sum_{i=1}^{n} X_i^z\ .$$

Do tohoto bodu umístíme rovinu ρ, kterou budeme následně optimalizovat, přičemž její počáteční polohu určenou jednotkovým normálovým vektorem n_ρ volíme libovolně.

Tvrzení 4.1 (*O těžišti*). Rovina symetrie ω bodové množiny prochází těžištěm této množiny. ■

Důkaz: Buď množina $\mathcal{X}$ *n* bodů $\{X_i\}_{i=1}^n$ v eukleidovském prostoru E_3. Nechť rovina symetrie ω dělí $\mathcal{X}$ na $\mathcal{X}_R$, $\mathcal{X}_L$ a $\mathcal{X}_S$ tak, že platí $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_R \cup \mathcal{X}_S$, $\mathcal{X}_L \cap \mathcal{X}_R = \emptyset$, $\mathcal{X}_L \cap \mathcal{X}_S = \emptyset$, $\mathcal{X}_S \cap \mathcal{X}_R = \emptyset$ a $|\mathcal{X}_L| = |\mathcal{X}_R|$ (počet bodů množin $\mathcal{X}_L$ a $\mathcal{X}_R$ je stejný). To znamená, body $\mathcal{X}_R$ leží v jednom poloprostoru určeného rovinou symetrie ω, body $\mathcal{X}_R$ v opačném poloprostoru a body $\mathcal{X}_S$ leží v rovině symetrie ω. Bez újmy na obecnosti dále předpokládejme, že je rovina symetrie ω umístěna do souřadnicové roviny (y,z). Do této polohy lze rovinu symetrie vždy přetransformovat. Potom platí, že

$$\mathcal{X}_R = \{X_i; X_i^x > 0, i = 1,2,...,n\},$$
$$\mathcal{X}_L = \{X_i; X_i^x < 0, i = 1,2,...,n\},$$
$$\mathcal{X}_S = \{X_i; X_i^x = 0, i = 1,2,...,n\}.$$

V rovinové symetrii s rovinou ω pro každý bod $X_i = [X_i^x, X_i^y, X_i^z]$ množiny $\mathcal{X}$ platí, že $f_\omega(X_i) = [-X_i^x, X_i^y, X_i^z]$. Potom $\forall X_i \in \mathcal{X}_L\ \exists X_j \in \mathcal{X}_R$ tak, že $f_\omega(X_i) = X_j$ a $\forall X_i \in \mathcal{X}_R$ $\exists X_j \in \mathcal{X}_L$ tak, že $f_\omega(X_i) = X_j$. Pro $\forall X_i \in \mathcal{X}_S$ platí $f_\omega(X_i) = X_i$.

Nyní chceme ukázat, že $T^x = 0$, tj. těžiště, jehož souřadnice určíme jako aritmetický průměr bodů množiny $\mathcal{X}$ podle (4.12), leží v rovině ω. Souřadnici T^x těžiště T spočítáme následovně

$$T^x = \frac{1}{n}\sum_{i=1}^{n} X_i^x = \frac{1}{n}\left(\sum_{X_i \in \mathcal{X}_L} X_i^x + \sum_{X_i \in \mathcal{X}_R} X_i^x + \sum_{X_i \in \mathcal{X}_S} X_i^x \right).$$

Víme, že $X_i^x = 0$ pro $X_i \in \mathcal{X}_S$, proto

$$T^x = \frac{1}{n}\left(\sum_{X_i \in \mathcal{X}_L} X_i^x + \sum_{X_i \in \mathcal{X}_L} \left(f_\omega(X_i)\right)^x \right) = \frac{1}{n}\left(\sum_{X_i \in \mathcal{X}_L} X_i^x + \sum_{X_i \in \mathcal{X}_L} (-X_i^x) \right) =$$

$$= \frac{1}{n} \cdot 0 = 0.$$

Těžiště T má souřadnice

$$T = \left[0, \frac{1}{n}\sum_{i=1}^{n} X_i^y, \frac{1}{n}\sum_{i=1}^{n} X_i^z \right],$$

tím je tvrzení dokázáno. ■

Důsledek 4.1. Těžiště T_R a T_L bodových množin $\mathcal{X}_R$ a $\mathcal{X}_L$ z důkazu tvrzení 4.1 si odpovídají v rovinové symetrii s rovinou ω. ■

Důkaz: Těžiště T_R a T_L spočítáme podle (4.12) jako aritmetické průměry bodů příslušných bodových množin $\mathcal{X}_R$ a $\mathcal{X}_L$, tj.

$$T_R = \left[\frac{1}{p} \sum_{X_i \in \mathcal{X}_R} X_i^x, \frac{1}{p} \sum_{X_i \in \mathcal{X}_R} X_i^y, \frac{1}{p} \sum_{X_i \in \mathcal{X}_R} X_i^z \right],$$

$$T_L = \left[\frac{1}{p} \sum_{X_i \in \mathcal{X}_L} X_i^x, \frac{1}{p} \sum_{X_i \in \mathcal{X}_L} X_i^y, \frac{1}{p} \sum_{X_i \in \mathcal{X}_L} X_i^z \right],$$

kde $p = |\mathcal{X}_L| = |\mathcal{X}_R|$. Z předchozích úvah plyne, že

$$T_R = \left[\frac{1}{p} \sum_{X_i \in \mathcal{X}_L} (-X_i^x), \frac{1}{p} \sum_{X_i \in \mathcal{X}_L} X_i^y, \frac{1}{p} \sum_{X_i \in \mathcal{X}_L} X_i^z \right].$$

To znamená, že

$$f_\omega(T_R) = T_L \text{ a } f_\omega(T_L) = T_R . \blacksquare$$

Jelikož rovina symetrie bodové množiny prochází jejím těžištěm, v dalších krocích algoritmu aktualizujeme pouze normálový vektor n_ρ roviny ρ, dokud se rovina dostatečně nepřiblíží rovině symetrie ω z definice 4.6.

V každém kroku algoritmu rozdělí rovina ρ v příslušné aktuální poloze eukleidovský prostor E_3 na dva poloprostory. Určíme, které body vstupní množiny leží v jednom a v druhém poloprostoru. Podle orientace normálového vektoru n_ρ roviny ρ prohlásíme jeden poloprostor za kladný a druhý za záporný. Značme množiny bodů v kladném poloprostoru ko $\mathcal{M}_1$ a v záporném poloprostoru jako $\mathcal{M}_2$. Je-li rovina ρ dána obecnou rovnicí ve tvaru

$$n_\rho \cdot (X - T) = 0 \tag{4.13}$$

kde T je těžiště vstupní bodové množiny a n_ρ je jednotkový normálový vektor roviny ρ, tj. $\|n_\rho\| = 1$, určujeme body množin $\mathcal{M}_1$ a $\mathcal{M}_2$ podle jednoduchého pravidla. Bod $X_i = [X_i^x, X_i^y, X_i^z]$ vstupní množiny $\mathcal{X}$ přidáme do množiny $\mathcal{M}_1$ (nebo $\mathcal{M}_2$), leží-li v poloprostoru (nebo v opačném poloprostoru) určeného rovinou ρ, tj. jestliže

$$n_\rho \cdot (X_i - T) > 0 \text{ (nebo } n_\rho \cdot (X_i - T) < 0), \tag{4.14}$$

Bodové množiny $\mathcal{M}_1$ a $\mathcal{M}_2$ jsou tedy definované takto

$$\mathcal{M}_1 = \{ X_i ; n_\rho \cdot (X_i - T) > 0, i = 1, 2, ..., n \}, \tag{4.15}$$

$$\mathcal{M}_2 = \{ X_i ; n_\rho \cdot (X_i - T) < 0, i = 1, 2, ..., n \}.$$

Vše určujeme v aktuální poloze roviny ρ. Body, které leží v rovině ρ v příslušné aktuální poloze není třeba uvažovat, protože nezmění hodnotu chybové funkce.

Nyní v obou poloprostorech spočítáme těžiště bodových množin $\mathcal{M}_1$ a $\mathcal{M}_2$. Těžiště bodové množiny $\mathcal{M}_1$ značme T_1, těžiště bodové množiny $\mathcal{M}_2$ značme T_2. Těžiště T_1 a T_2 zobrazíme v rovinové symetrii s rovinou ρ, tj. $f_\rho(T_1) = T_1'$ a $f_\rho(T_2) = T_2'$. Podle důsledku 4.1 víme, že v případě, kdy se aktuální poloha roviny ρ shoduje s rovinou symetrie, odpovídají si těžiště T_1 a T_2 v rovinové symetrii s rovinou ρ, tj. $T_1' = T_2$ a $T_2' = T_1$. Tohoto stavu chceme docílit minimalizací chybové funkce. Určujeme proto vzdálenosti těžiště a zobrazeného těžiště z opačného poloprostoru v rovinové symetrii s rovinou ρ v aktuální poloze, tj. $distance_1 = |T_1 T_2'|$ a $distance_2 = |T_2 T_1'|$.

Dále zavádíme chybovou funkci, značme ji δ^2, kterou definuje následovně

$$\delta^2 = (distance_1^2 + distance_2^2) \cdot \left(1 + |p_1 - p_2|\right), \tag{4.16}$$

kde $p_1 = |\mathcal{M}_1|$ a $p_2 = |\mathcal{M}_2|$.

Součet druhých mocnin vzdáleností $distance_1$ a $distance_2$ násobíme konstantou $1 + |p_1 - p_2|$, která je tím větší, čím méně souhlasí počet bodů v poloprostorech určených rovinou ρ v příslušné aktuální poloze. Tím zajistíme lepší průběh minimalizace chybové funkce. Hodnotu 1 přičítáme proto, aby byl výraz v závorce nenulový. Odpovídá-li si totiž počet bodů v poloprostorech, ještě zbývá ustavit korespondenci těžišť.

Minimalizaci chybové funkce z (4.16) provádíme pomocí diferenciální numerické metody největšího spádu a to její modifikovanou verzí s konstantní délkou kroku.

Chybová funkce má celkem tři parametry – souřadnice normálového vektoru n_ρ roviny ρ, které v každém kroku algoritmu aktualizujeme. Jedná se tedy o reálnou funkci tří reálných proměnných.

Výpočet chybové funkce předkládáme v následujícím symbolickém kódu jako algoritmus 4.7. Chybovou funkci značíme v pseudokódu *error_function_one_plane* a její hodnotu pro konkrétní tři parametry jako *error*.

Algoritmus 4.7: *VÝPOČET CHYBOVÉ FUNKCE ERROR_FUNCTION_ONE_-PLANE PRO ROVINOVOU SYMETRII*. Vstupním parametrem chybové funkce je jednotkový normálový vektor n_ρ roviny ρ, výstupem je reálná hodnota *error*.

Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, která je symetrická podle nějaké roviny symetrie. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus využívá tři externí funkce *subspace_one_plane*, *centre* a *mirror*. Funkce *subspace_one_plane* slouží k rozdělení vstupních bodů na dvě množiny v jednom a druhém poloprostoru podle (4.15). Poloprostory jsou určené rovinou ρ. Funkce *centre* počítá těžiště množiny bodů dané maticí *points* podle vzorce (4.11). Funkce *mirror* určuje souměrný bod k bodu A podle roviny ρ.

V symbolickém kódu je použito několik proměnných. Proměnné $\mathcal{M}_1$ a $\mathcal{M}_2$ jsou množiny bodů v kladném a záporném poloprostoru. Proměnné T_1 a T_2 jsou těžiště bodových množin $\mathcal{M}_1$ a $\mathcal{M}_2$ a proměnné T_1' a T_2' jsou jejich obrazy v rovinové symetrii s rovinou ρ. Do proměnných p_1 a p_2 ukládáme počet bodů množin $\mathcal{M}_1$ a $\mathcal{M}_2$ a do proměnných $distance_1$ a $distance_2$ vzdálenosti těžiště a zobrazeného těžiště.

---

function *error_function_one_plane*(n_ρ): **real**
1: $(\mathcal{M}_1, \mathcal{M}_2) \leftarrow subspace\_one\_plane(T, n_\rho)$
2: $T_1 \leftarrow centre\ (\mathcal{M}_1)$
3: $T_2 \leftarrow centre\ (\mathcal{M}_2)$
4: $T_1' \leftarrow mirror(T_1, T, n_\rho)$
5: $T_2' \leftarrow mirror(T_2, T, n_\rho)$
6: $distance_1 \leftarrow \|T_1 - T_2'\|$
7: $distance_2 \leftarrow \|T_2 - T_1'\|$
8: $p_1 \leftarrow |\mathcal{M}_1|$
9: $p_2 \leftarrow |\mathcal{M}_2|$
10: $error \leftarrow (distance_1^2 + distance_2^2) \cdot (1 + |p_1 - p_2|)$
11: **return** *error*

function *subspace_one_plane*(T, n_ρ): **pair of sets**
1: $\mathcal{M}_1 \leftarrow \emptyset$
2: $\mathcal{M}_2 \leftarrow \emptyset$
3: **for** $i = 1, 2, ..., n$ **do**
4: | **if** $n_\rho \cdot (X[i] - T) > 0$ **then**
5: | | $\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \{X[i]\}$
6: | **else**
7: | | $\mathcal{M}_2 \leftarrow \mathcal{M}_2 \cup \{X[i]\}$
8: **return** $(\mathcal{M}_1, \mathcal{M}_2)$

function *centre*(*points*): **point**
1: $p \leftarrow |points|$
2: **if** $p \neq 0$ **then**
3: | $centre \leftarrow \frac{1}{p} \sum_{i=1}^{n} X[i]$
4: **else**
5: | **return** {*failure*}
6: **return** *centre*

function *mirror*(A, T, n_ρ): **point**
1: $w \leftarrow A - T$
2: $\overline{A} \leftarrow A - (n_\rho \cdot w) n_\rho$
%{pravoúhlý průmět bodu A do roviny určené bodem T a jednotkovým normálovým vektorem n_ρ, odvozeno v oddíle o ortogonálním prokládání rovinou}%

3: $A' \leftarrow \overline{A} + (\overline{A} - A)$
4: **return** A'

Chybovou funkci z algoritmu 4.7 nyní minimalizujeme pomocí modifikované metody největšího spádu s konstantní délkou kroku. Proces hledání roviny symetrie bodové množiny zapišme opět pomocí pseudokódu jako algoritmus 4.8.

Algoritmus 4.8: *HLEDÁNÍ ROVINY SYMETRIE BODOVÉ MNOŽINY*. Algoritmus hledání roviny symetrie je založen na modifikované metodě největšího spádu a je symbolicky popsán jako funkce. Vstupním parametrem funkce je jednotkový normálový vektor n_ρ roviny ρ, volíme jej libovolně. Výstupem funkce je minimalizující posloupnost normálových vektorů – *sequence_vectors*. Poslední vektor této posloupnosti je normálový vektor roviny, která tvoří odhad hledané roviny symetrie.

Předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, která je symetrická podle nějaké roviny symetrie. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus využívá externí funkci *error_function_one_plane* popsanou v algoritmu 4.7 a externí funkci *centre* pro výpočet těžiště vstupní množiny bodů uvedenou v předchozím symbolickém kódu.

Další použité proměnné jsou následující. Proměnné Δn_ρ (Δn_ρ*_x,y,z*) slouží k výpočtu numerické parciální derivace, n_ρ^{STEP} je aktuální normálový vektor v jednom kroku iterace. Proměnné ∂n_ρ^{STEP} je numericky vypočítaná parciální derivace funkce *error_function_one_plane* podle normálového vektoru n_ρ (myšleno po souřadnicích) v n_ρ^{STEP}, ∇ je gradient funkce *error_function_one_plane* pro hodnoty n_ρ^{STEP}. Proměnná new_n_ρ je nový vektor minimalizující posloupnosti získané modifikovanou metodou největšího spádu v jednom kroku minimalizace. Iterační proces je ukončen, pokud je splněna jedna z podmínek.

V pseudokódu používáme ještě globální proměnné δ, ε, λ a *max_iteration*, které určují nastavení metody minimalizace. Vhodná je například volba

$\delta = 0,001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,0001$
$max\_iteration = 100$.

---

function *search_symmetry_one_plane*(n_ρ): **sequence of vectors**
1: $T \leftarrow centre(X)$
2: $plot(plane, T, n_\rho)$
%{vykreslení počáteční polohy roviny s určujícím bodem T a normálovým vektorem n_ρ}%
3: $\Delta n_\rho \leftarrow n_\rho$
4: $n_\rho^{STEP} \leftarrow n_\rho$
5: $iteration \leftarrow 0$

6: $sequence\_vectors[1] \leftarrow n_\rho$
7: **while** *True* **do** %{ukončení cyklu řízeno uvnitř}
8: $iteration \leftarrow iteration + 1$
9: $\Delta n_\rho\_x \leftarrow \Delta n_\rho + (\delta, 0, 0)$
10: $\Delta n_\rho\_y \leftarrow \Delta n_\rho + (0, \delta, 0)$
11: $\Delta n_\rho\_z \leftarrow \Delta n_\rho + (0, 0, \delta)$
12: $error_\Delta \leftarrow error\_function\_one\_plane(\Delta n_\rho)$
13: $error_{STEP} \leftarrow error\_function\_one\_plane(n_\rho^{STEP})$
14: $\partial n_\rho^{STEP} \leftarrow (error_\Delta - error_{STEP}) / \delta$
%{parciální derivace funkce *error_function_one_plane* podle vektoru n_ρ v n_ρ^{STEP} }%
15: $\nabla \leftarrow (\partial n_\rho^{STEP})$
%{gradient funkce *error_function_one_plane* }%
16: $accuracy \leftarrow norm(\nabla)$
17: **if** $accuracy \le \varepsilon$ **or** $iteration \ge max\_iteration$ **then**
%{podmínka ukončení iterace}%
18: **break**
19: $new\_n_\rho \leftarrow n_\rho^{STEP} - \lambda \cdot \nabla$
20: $new\_n_\rho \leftarrow new\_n_\rho / \left\| new\_n_\rho \right\|$
21: $sequence\_vectors[iteration] \leftarrow new\_n_\rho$
22: $\Delta n_\rho \leftarrow new\_n_\rho$
23: $n_\rho^{STEP} \leftarrow new\_n_\rho$
24: $plot(plane, T, n_\rho^{STEP})$
%{vykreslení aktuální polohy roviny s určujícím bodem T a normálovým vektorem n_ρ^{STEP}, animace minimalizační metody}%
25: $plot(plane, T, n_\rho^{STEP})$
%{vykreslení odhadu polohy hledané roviny s určujícím bodem T a normálovým vektorem n_ρ^{STEP} }%
26: $plot(X)$
%{vykreslení bodů vstupní množiny}%
27: **return** (*sequence_vectors*)

Funkčnost popsaného algoritmu si ukážeme na příkladě počítačově generované množiny bodů, o které předpokládáme, že je symetrická podle roviny.

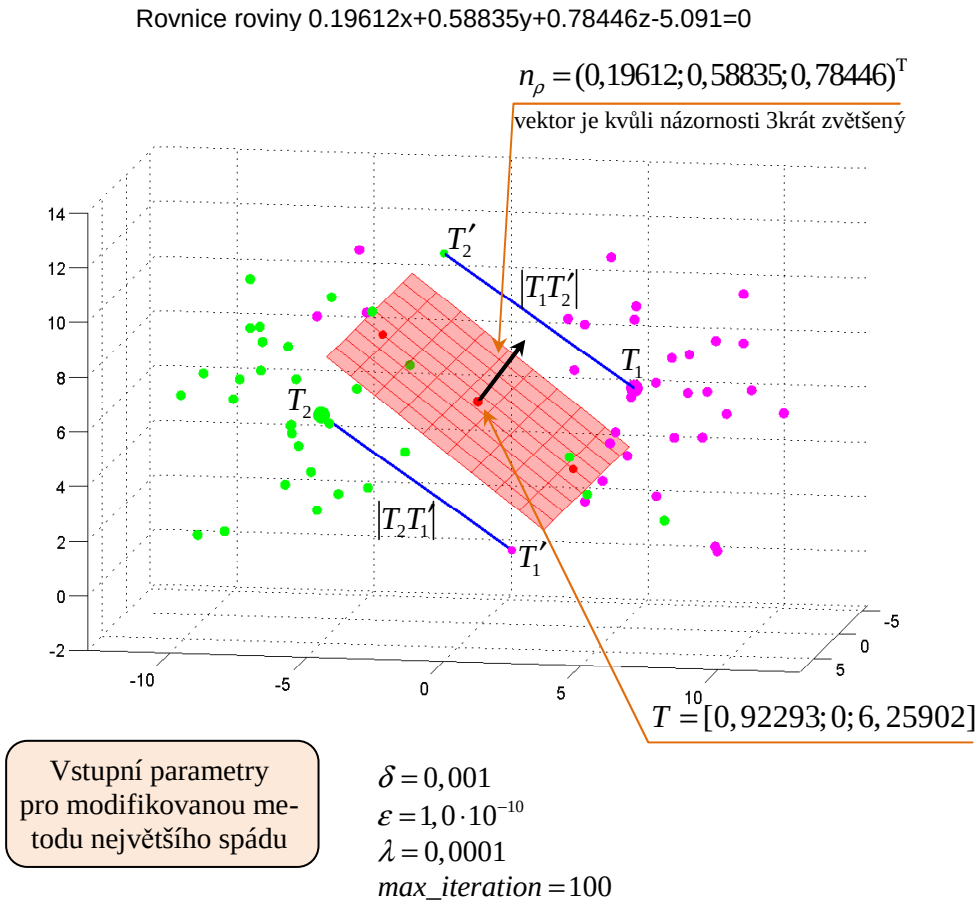
Příklad 4.5: *HLEDÁNÍ ROVINY SYMETRIE BODOVÉ MNOŽINY*. Navržený algoritmus předvedeme na příkladě experimentální bodové množiny. Předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, které jsou symetrické podle nějaké roviny. Pro lepší názornost volíme řídkou množinu bodů.

Na obrázku 4.79 je zakreslena vstupní množina bodů, počáteční poloha roviny, která se postupně optimalizuje, a předložena je její obecná rovnice. Body kladného

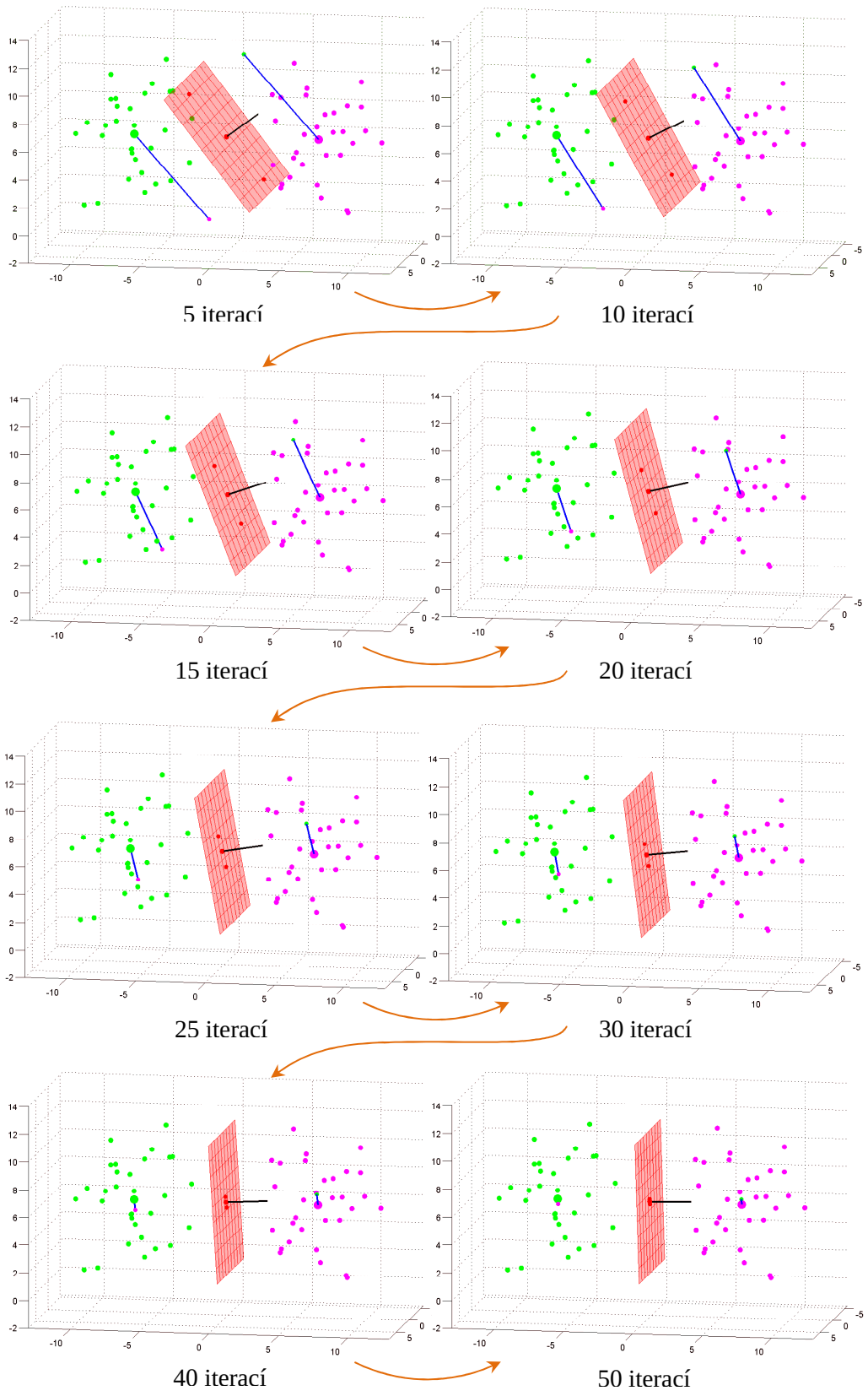
a záporného poloprostoru jsou barevně odlišeny. V obou poloprostorech jsou vykreslena těžiště T_1 a T_2 a jejich obrazy T_1' a T_2' v rovinové symetrii se zadanou rovinou. Modře jsou vyznačeny vzdálenosti těžišť a obrazů těžišť z opačného poloprostoru. Obrázek 4.80 ilustruje postupné vylepšování polohy roviny minimalizací chybové funkce, přičemž se rovina vykresluje pouze v několika pozicích. Zvlášť je vykreslena výsledná poloha roviny a vypsána je její obecná rovnice, viz obrázek 4.81. Dostatečně dobrý odhad roviny symetrie dostáváme pro uvedenou volbu vstupních parametrů metody (100 iterací). Pokud bychom vyžadovali přesnější výsledek, je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroku λ. Pro minimalizaci chybové funkce je vykreslena také závislost hodnoty chybové funkce na počtu iterací, viz obrázek 4.82. Všechny vypisované číselné hodnoty v textu vždy zaokrouhlujeme na pět desetinných míst.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

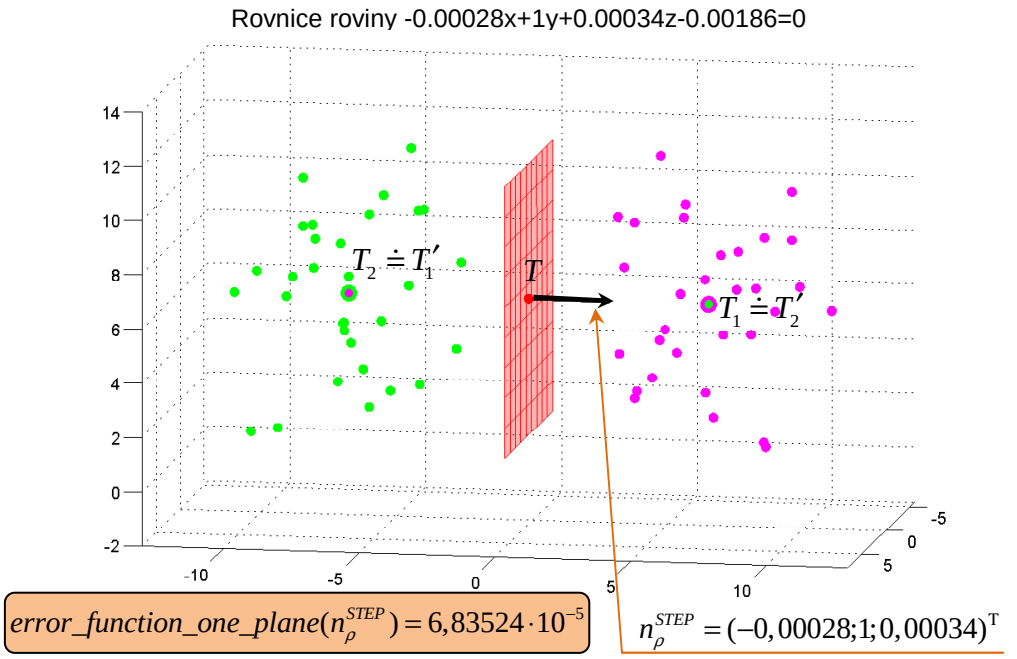
`symetrie_jedna_rovina.m` (programy/symetrie).



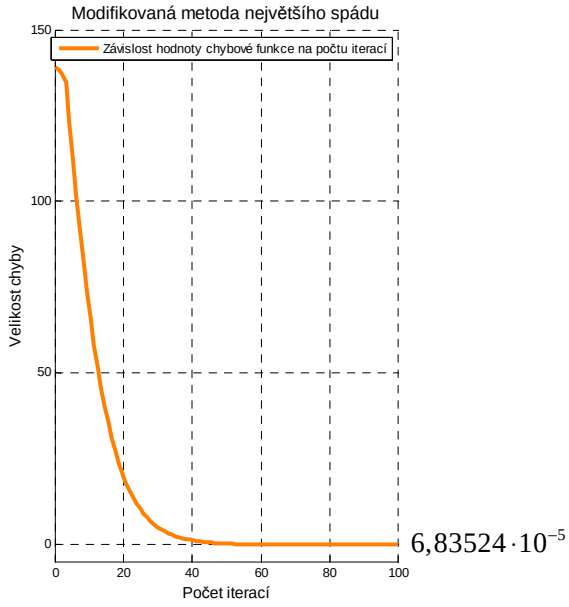
Obrázek 4.79: Symetrická bodová množina a počáteční poloha roviny – znázornění principu navržené metody a vstupní parametry minimalizace



Obrázek 4.80: Postupná optimalizace roviny minimalizací chybové funkce modifikovanou metodou největšího spádu



Obrázek 4.81: Výsledný odhad roviny symetrie



Obrázek 4.82: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací

Další příklady na hledání rovinové symetrie si uvedeme později pro reálné bodové množiny, které obsahují velké množství bodů.

4.4.2 Rovinové symetrie se dvěma navzájem kolmými rovinami

Nyní předpokládáme, že je bodová množina symetrická podle dvou navzájem kolmých rovin. Představme navržený postup hledání rovinových symetrií s těmito dvěma rovinami.

Mějme dánu neprázdnou množinu $\mathcal{X}$ n bodů v eukleidovském prostoru E_3, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Víme, že je tato vstupní bodová množina přesně symetrická podle dvou navzájem kolmých rovin.

Roviny symetrie hledáme opět iteračně s použitím odvozené minimalizace modifikovanou metodou největšího spádu. V prvním kroku algoritmu určíme těžiště T vstupní bodové množiny $\mathcal{X}$ podle (4.11). Do tohoto bodu umístíme dvě navzájem kolmé roviny ρ a σ, které budeme v každém kroku algoritmu optimalizovat, přičemž jejich počáteční polohy určené jednotkovými normálovými vektory n_ρ a n_σ volíme libovolně. Těžiště T tedy leží na průsečnici těchto rovin. Obě hledané roviny symetrií procházejí těžištěm (podle tvrzení 4.1), proto budeme v dalších krocích algoritmu aktualizovat pouze normálové vektory n_ρ a n_σ rovin ρ a σ, dokud se roviny dostatečně nepřiblíží rovinám symetrií podle definice 4.6.

Vysvětleme podrobněji počáteční volbu normálových vektorů n_ρ a n_σ. Směr jednotkového normálového vektoru n_ρ určujeme zcela libovolně, druhý jednotkový normálový vektor n_σ volíme tak, aby byl kolmý k vektoru n_ρ. Ke zvolenému vektoru n_ρ určíme dva navzájem kolmé vektory, značme je n_σ a a, tak, že vektory n_ρ, n_σ, a tvoří ortonormální bázi. Všechny tři vektory jsou tedy navzájem kolmé a navíc jednotkové. Z množiny vektorů kolmých k vektoru n_ρ vybíráme navzájem kolmé vektory n_σ a a následovně.

Mějme jednotkový normálový vektor $n_\rho = (n_\rho^x, n_\rho^y, n_\rho^z)^{\mathrm{T}}$ a určeme $\bar{a}$ jako

$$
\begin{aligned}
&\bar{a} = \left((-n_\rho^y - n_\rho^z)/n_\rho^x, 1, 1\right)^{\mathrm{T}}, \text{ je-li } n_\rho^x \neq 0, \text{ nebo} \\
&\bar{a} = \left(1, (-n_\rho^x - n_\rho^z)/n_\rho^y, 1\right)^{\mathrm{T}}, \text{ je-li } n_\rho^y \neq 0, \text{ nebo} \\
&\bar{a} = \left(1, 1, (-n_\rho^x - n_\rho^y)/n_\rho^z\right)^{\mathrm{T}}, \text{ je-li } n_\rho^z \neq 0.
\end{aligned}
\tag{4.17}
$$

Potom vektory a a n_σ mají tvar

$$
a = \frac{\bar{a}}{\|\bar{a}\|}, \; n_\sigma = \frac{n_\rho \times a}{\|n_\rho \times a\|}. \tag{4.18}
$$

Kvůli dalšímu postupu navíc vyžadujeme, aby vektory a, n_σ a n_ρ tvořili pravotočivou bázi. Proto je-li $(a \times n_\sigma) \cdot n_\rho < 0$, potom místo vektoru a uvažujeme opačný vektor $-a$.

V každém kroku algoritmu rozdělí roviny ρ a σ v příslušné aktuální poloze eukleidovský prostor E_3 na čtyři kvadranty. Pro každý kvadrant určíme, které body ze vstupní množiny v něm leží. Značme množiny bodů v kvadrantech $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ a $\mathcal{M}_4$. Jsou-li roviny ρ a σ dány obecnými rovnicemi ve tvaru

(4.19) $$n_\rho \cdot (X - T) = 0 \text{ a } n_\sigma \cdot (X - T) = 0$$

kde T je těžiště vstupní bodové množiny, n_ρ je jednotkový normálový vektor roviny ρ, tj. $\|n_\rho\| = 1$, a n_σ je jednotkový normálový vektor roviny σ, tj. $\|n_\sigma\| = 1$. Bod $X_i = [X_i^x, X_i^y, X_i^z]$ vstupní množiny $\mathcal{X}$ přidáme do množiny $\mathcal{M}_1$ ($\mathcal{M}_2$, $\mathcal{M}_3$ nebo $\mathcal{M}_4$), leží-li v 1. (2., 3. nebo 4.) kvadrantu určených rovinami ρ a σ, tj. bodové množiny $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ a $\mathcal{M}_4$ jsou definované

(4.20)
$$\begin{aligned}
\mathcal{M}_1 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_2 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) < 0, i = 1,2,...,n\},\\
\mathcal{M}_3 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_4 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) < 0, i = 1,2,...,n\}.
\end{aligned}$$

Opět není třeba uvažovat body vstupní množiny, které leží v rovinách ρ a σ.

Dále spočítáme těžiště bodových množin $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ a $\mathcal{M}_4$, značme je po řadě T_1, T_2, T_3 a T_4, a zobrazíme je v rovinových symetriích s rovinami ρ a σ, tj. $f_\rho(T_1) = T_{11}$, $f_\sigma(T_1) = T_{12}$, $f_\rho(T_2) = T_{21}$, $f_\sigma(T_2) = T_{22}$, $f_\rho(T_3) = T_{31}$, $f_\sigma(T_3) = T_{32}$ a $f_\rho(T_4) = T_{41}$, $f_\sigma(T_4) = T_{42}$. V případě, rovnají-li se roviny ρ a σ v aktuální poloze rovinám symetrií, odpovídají si těžiště T_1, T_2, T_3 a T_4 v rovinových symetriích s rovinami ρ a σ, tj. $T_1 = T_{31} = T_{22}$, $T_2 = T_{41} = T_{12}$, $T_3 = T_{11} = T_{42}$ a $T_4 = T_{21} = T_{32}$. Tohoto stavu chceme docílit minimalizací chybové funkce, k jejíž definici určujeme vzdálenosti těžiště a dvojice zobrazených těžišť ze sousedních kvadrantů v rovinových symetriích s rovinami ρ a σ. Vzdálenosti jsou $distance_1 = |T_1T_{22}|$, $distance_2 = |T_1T_{31}|$, $distance_3 = |T_2T_{12}|$, $distance_4 = |T_2T_{41}|$, $distance_5 = |T_3T_{42}|$, $distance_6 = |T_3T_{11}|$, $distance_7 = |T_4T_{32}|$ a $distance_8 = |T_4T_{21}|$.

Zavádíme chybovou funkci, značme ji δ^2, a definujeme ji následovně

(4.21)
$$\begin{aligned}
\delta^2 = &(distance_1^2 + distance_3^2) \cdot (1 + |p_1 - p_2|) +\\
&+ (distance_2^2 + distance_6^2) \cdot (1 + |p_1 - p_3|) +\\
&+ (distance_4^2 + distance_8^2) \cdot (1 + |p_4 - p_2|) +\\
&+ (distance_5^2 + distance_7^2) \cdot (1 + |p_4 - p_3|),
\end{aligned}$$

kde $p_j = |\mathcal{M}_j|$.

Součet druhých mocnin dvojic vzdáleností vždy násobíme konstantou $1 + |p_{1,4} - p_{2,3}|$, čímž zajistíme lepší průběh minimalizace chybové funkce. Tato konstanta je tím větší, čím méně si odpovídá počet bodů ve dvou sousedních kvadrantech určených rovinami ρ a σ v příslušné aktuální poloze. Poté co si odpovídají počty bodů v sousedních kvadrantech, ustanovujeme rovněž korespondenci těžišť. Pro minimalizaci chybové funkce z (4.21) volíme opět modifikovanou metodu největšího spádu s konstantní délkou kroku.

Chybová funkce má celkem šest parametrů – souřadnice normálového vektoru n_ρ roviny ρ a souřadnice normálového vektoru n_σ roviny σ, jedná se tedy o reálnou funkci šesti reálných proměnných. Ovšem v daném kroku minimalizace vždy aktualizujeme střídavě pouze tři parametry, tj. souřadnice jednoho z normálových vektorů. Zmiňovaných šest parametrů totiž není nezávislých. To je zřejmé, neboť jsou na sebe vektory n_ρ a n_σ kolmé, nelze je proto modifikovat najednou.

Popišme, jak v daném kroku minimalizace postupujeme. Při aktualizaci normálového vektoru n_ρ určíme minimalizací chybové funkce nový jednotkový normálový vektor, značme jej new_n_ρ. Tři navzájem kolmé jednotkové vektory n_ρ, n_σ, a přetransformujeme do nové polohy tak, že jednotkový normálový vektor n_ρ přejde do vektoru new_n_ρ.

Transformace probíhá následovně. Vektor new_n_ρ pravoúhle promítneme do roviny určené vektory n_ρ a a do vektoru p, tedy $p = (new\_n_\rho \cdot n_\rho)\, n_\rho + (new\_n_\rho \cdot a)\, a$. To lze jednoduše odvodit. Pro libovolné dva nenulové vektory u, v totiž platí $u \cdot v = \|u\|\|v\|\cos\alpha$, kde $\alpha \in \langle 0, \pi \rangle$ je úhel, který vektory svírají. Délka pravoúhlého průmětu vektoru v do vektoru u je tedy $\|v\|\cos\alpha = v \cdot \frac{u}{\|u\|}$ a vektor $(v \cdot \frac{u}{\|u\|})\frac{u}{\|u\|}$ je pravoúhlý průmět vektoru v do vektoru u. Vektory n_ρ a a jsou oba jednotkové, proto se ve výpočtu vektoru p neobjevují jejich normy.

Vektory n_ρ a a otočíme kolem osy dané vektorem n_σ o úhel $\varphi \in \langle 0, \pi \rangle$, který svírají vektory n_ρ a p, do vektorů aux_n_ρ a new_a. Je nutné rozlišit případy, kdy otáčíme ve směru a proti směru hodinových ručiček. To učiníme na základě znamének koeficientů lineární kombinace vektorů n_ρ a a v předpisu vektoru p. Je-li $(new\_n_\rho \cdot n_\rho) > 0$, $(new\_n_\rho \cdot a) > 0$ nebo $(new\_n_\rho \cdot n_\rho) < 0$, $(new\_n_\rho \cdot a) > 0$, otáčíme o úhel φ, tedy $aux\_n_\rho = n_\rho \cos\varphi + a \sin\varphi$ a $new\_a = -n_\rho \sin\varphi + a\cos\varphi$. Je-li $(new\_n_\rho \cdot n_\rho) > 0$, $(new\_n_\rho \cdot a) < 0$ nebo $(new\_n_\rho \cdot n_\rho) < 0$, $(new\_n_\rho \cdot a) < 0$, otáčíme o úhel $-\varphi$, tedy $aux\_n_\rho = n_\rho \cos(-\varphi) + a \sin(-\varphi)$ a $new\_a = -n_\rho \sin(-\varphi) + a\cos(-\varphi)$.

Dále otočíme vektory n_σ a aux_n_ρ kolem osy dané vektorem new_a o úhel $\omega \in \langle 0, \frac{\pi}{2} \rangle$, který svírají vektory new_n_ρ a aux_n_ρ, do vektorů new_n_σ a new_n_ρ. Zde směr otáčení rozlišíme na základě velikosti úhlu $\beta \in \langle 0, \pi \rangle$, který svírají vektory n_σ a new_n_ρ. Je-li úhel β větší než pravý, otáčíme o úhel ω, tedy $new\_n_\sigma = n_\sigma \cos\omega + aux\_n_\rho \sin\omega$ a $new\_n_\rho = -n_\sigma \sin\omega + aux\_n_\rho \cos\omega$. Je-li úhel β menší než pravý, otáčíme o úhel $-\omega$, tedy $new\_n_\sigma = n_\sigma \cos(-\omega) + aux\_n_\rho \sin(-\omega)$ a $new\_n_\rho = -n_\sigma \sin(-\omega) + aux\_n_\rho \cos(-\omega)$. Popsaná transformace je znázorněna na obrázku obrázku 4.83.

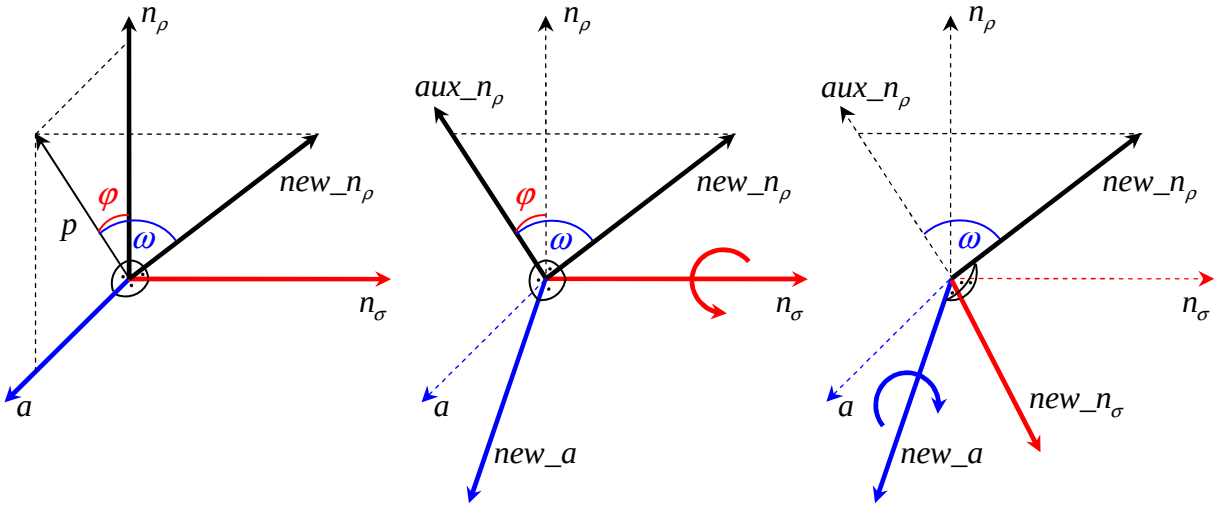
Při aktualizaci normálového vektoru n_σ určíme minimalizací chybové funkce nový jednotkový normálový vektor, značme jej new_n_σ. A dále postupujeme obdobně, tj. tři navzájem kolmé jednotkové vektory n_ρ, n_σ, a přetransformujeme do nové polohy tak, že jednotkový normálový vektor n_σ přejde do vektoru new_n_σ.

Průběh této transformace je analogický. Vektor new_n_σ pravoúhle promítneme do roviny určené vektory a a n_σ do vektoru p, tedy $p = (new\_n_\sigma \cdot a)\, a + (new\_n_\sigma \cdot n_\sigma)\, n_\sigma$. Vektory a a n_σ otočíme kolem osy dané vektorem n_ρ o úhel $\varphi \in \langle 0, \pi \rangle$, který svírají vektory p a n_σ, do vektorů new_a a aux_n_σ. Opět je nutné rozlišit případy, kdy otáčíme ve směru a proti směru hodinových ručiček. To učiníme na základě znamének koeficientů lineární kombinace vektorů a a n_σ v předpisu vektoru p. Je-li $(new\_n_\sigma \cdot a) < 0$, $(new\_n_\sigma \cdot n_\sigma) > 0$

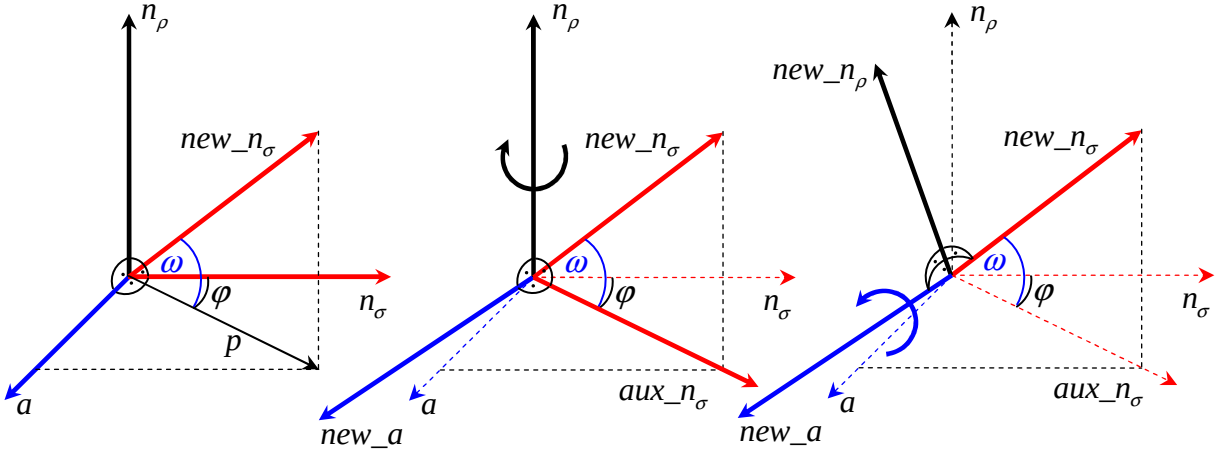
nebo $(new\_n_\sigma \cdot a) < 0$, $(new\_n_\sigma \cdot n_\sigma) < 0$, otáčíme o úhel φ, tedy $new\_a = a\cos\varphi + n_\sigma \sin\varphi$ a $aux\_n_\sigma = -a\sin\varphi + n_\sigma \cos\varphi$. Je-li $(new\_n_\sigma \cdot a) > 0$, $(new\_n_\sigma \cdot n_\sigma) > 0$ nebo $(new\_n_\sigma \cdot a) > 0$, $(new\_n_\sigma \cdot n_\sigma) < 0$, otáčíme o úhel $-\varphi$, tedy $new\_a = a\cos(-\varphi) + n_\sigma \sin(-\varphi)$ a $aux\_n_\sigma = -a\sin(-\varphi) + n_\sigma \cos(-\varphi)$.

Dále otočíme vektory aux_n_σ a n_ρ kolem osy dané vektorem new_a o úhel $\omega \in \langle 0, \frac{\pi}{2} \rangle$, který svírají vektory new_n_σ a aux_n_σ, do vektorů new_n_σ a new_n_ρ. Směr otáčení rozlišíme na základě velikosti úhlu $\beta \in \langle 0, \pi \rangle$, který svírají vektory new_n_σ a n_ρ. Je-li úhel β menší než pravý, otáčíme o úhel ω, tedy $new\_n_\sigma = aux\_n_\sigma \cos\omega + n_\rho \sin\omega$ a $new\_n_\rho = -aux\_n_\sigma \sin\omega + n_\rho \cos\omega$. Je-li úhel β větší než pravý, otáčíme o úhel $-\omega$, tedy $new\_n_\sigma = aux\_n_\sigma \cos(-\omega) + n_\rho \sin(-\omega)$ a $new\_n_\rho = -aux\_n_\sigma \sin(-\omega) + n_\rho \cos(-\omega)$. Popsaná transformace je znázorněna na obrázku obrázku 4.84.

Obě transformace budou ještě podrobně uvedeny v symbolickém kódu.



Obrázek 4.83: Transformace tří navzájem kolmých vektorů – aktualizace vektoru n_ρ



Obrázek 4.84: Transformace tří navzájem kolmých vektorů – aktualizace vektoru n_σ

Výpočet chybové funkce předkládáme v následujícím pseudokódu jako algoritmus 4.9. Chybová funkce je označena jako *error_function_two_planes* a její hodnota pro konkrétních šest parametrů jako *error* .

Algoritmus 4.9: *VÝPOČET CHYBOVÉ FUNKCE ERROR_FUNCTION_TWO_-PLANES PRO ROVINOVÉ SYMETRIE SE DVĚMA NAVZÁJEM KOLMÝMI ROVINAMI*. Vstupními parametry chybové funkce jsou jednotkový normálový vektor n_ρ roviny ρ a jednotkový normálový vektor n_σ roviny σ, výstupem je reálná hodnota *error* .

Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, která je symetrická podle dvou navzájem kolmých rovin. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X . Algoritmus využívá tři externí funkce *subspace_two_planes* , *centre* a *mirror* . Funkce *subspace_two_planes* rozděluje vstupní body na čtyři množiny bodů v kvadrantech podle (4.20), přičemž kvadranty jsou určené rovinami ρ a σ. Funkce *centre* a *mirror* jsou popsané v algoritmu 4.7.

V symbolickém kódu je použito několik proměnných. Proměnné $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ a $\mathcal{M}_4$ jsou množiny bodů v kvadrantech. Proměnné T_1 , T_2 , T_3 a T_4 jsou těžiště těchto bodových množin a proměnné T_{11} , T_{12} , T_{21} , T_{22} , T_{31} , T_{32} , T_{41} a T_{42} jsou jejich obrazy v rovinových symetriích s rovinami ρ a σ. Do proměnných p_j ukládáme počty bodů množin $\mathcal{M}_j$ a do proměnné d vzdálenosti těžiště a zobrazených těžišť v rovinových symetriích s rovinami ρ a σ.

---

function *error_function_two_planes*(n_ρ, n_σ): **real**

1: $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4) \leftarrow$ *subspace_two_planes*(T, n_ρ, n_σ)
2: $T_1 \leftarrow$ *centre* $(\mathcal{M}_1)$
3: $T_2 \leftarrow$ *centre* $(\mathcal{M}_2)$
4: $T_3 \leftarrow$ *centre* $(\mathcal{M}_3)$
5: $T_4 \leftarrow$ *centre* $(\mathcal{M}_4)$
6: $T_{11} \leftarrow mirror(T_1, T, n_\rho)$; $T_{12} \leftarrow mirror(T_1, T, n_\sigma)$
7: $T_{21} \leftarrow mirror(T_2, T, n_\rho)$; $T_{22} \leftarrow mirror(T_2, T, n_\sigma)$
8: $T_{31} \leftarrow mirror(T_3, T, n_\rho)$; $T_{32} \leftarrow mirror(T_3, T, n_\sigma)$
9: $T_{41} \leftarrow mirror(T_4, T, n_\rho)$; $T_{42} \leftarrow mirror(T_4, T, n_\sigma)$
10: $d \leftarrow \left[\|T_1 - T_{22}\|, \|T_1 - T_{31}\|, \|T_2 - T_{12}\|, \|T_2 - T_{41}\|, \|T_3 - T_{42}\|, \|T_3 - T_{11}\|, \|T_4 - T_{32}\|, \|T_4 - T_{21}\|\right]$
11: $p_1 \leftarrow |\mathcal{M}_1|$
12: $p_2 \leftarrow |\mathcal{M}_2|$
13: $p_3 \leftarrow |\mathcal{M}_3|$
14: $p_4 \leftarrow |\mathcal{M}_4|$
15: $error \leftarrow \left[(d[1])^2 + d[3])^2\right] \cdot \left(1 + |p_1 - p_2|\right) + \left[(d[2])^2 + (d[6])^2\right] \cdot \left(1 + |p_1 - p_3|\right) +$
$+ \left[(d[4])^2 + (d[8])^2\right] \cdot \left(1 + |p_4 - p_2|\right) + \left[(d[5])^2 + d[7])^2\right] \cdot \left(1 + |p_4 - p_3|\right)$
16: **return** *error*

```
function subspace_two_planes(T, n_ρ, n_σ): four-tuple of sets
1:  M_1 ← ∅
2:  M_2 ← ∅
3:  M_3 ← ∅
4:  M_4 ← ∅
5:  for i = 1,2,...,n do
6:      if n_ρ · (X[i] − T) > 0 then
7:          if n_σ · (X[i] − T) > 0 then
8:              M_1 ← M_1 ∪ {X[i]}
9:          else
10:             M_2 ← M_2 ∪ {X[i]}
11:     else
12:         if n_σ · (X[i] − T) > 0 then
13:             M_3 ← M_3 ∪ {X[i]}
14:         else
15:             M_4 ← M_4 ∪ {X[i]}
16: return (M_1, M_2, M_3, M_4)
```

Chybovou funkci z algoritmu 4.9 následně minimalizujeme modifikovanou metodou největšího spádu s konstantní délkou kroku. Získáme tak odhad dvou navzájem kolmých rovin symetrií. Proces minimalizace zapišme symbolickým kódem jako algoritmus 4.10.

V pseudokódu je zvýrazněna část, kdy aktualizujeme normálový vektor n_ρ roviny ρ (řádky 15 až 49), a část, kdy aktualizujeme normálový vektor n_σ roviny σ (řádky 50 až 84).

Algoritmus 4.10: *HLEDÁNÍ DVOU NAVZÁJEM KOLMÝCH ROVIN SYMETRIÍ BODOVÉ MNOŽINY*. Algoritmus hledání dvou navzájem kolmých rovin symetrií je popsán jako funkce a je opět založen na modifikované metodě největšího spádu. Vstupním parametrem funkce je jednotkový normálový vektor n_ρ roviny ρ, volíme jej libovolně. Výstupem funkce jsou dvě minimalizující posloupnosti normálových vektorů – *sequence_vectors_*n_ρ a *sequence_vectors_*n_σ. Poslední vektory posloupností jsou normálové vektory rovin ρ a σ, které tvoří odhady hledaných rovin symetrií.

Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, o které předpokládáme, že je symetrická podle dvou vzájemně kolmých rovin. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus využívá externí funkci *error_function_two_planes* popsanou v algoritmu 4.9, externí funkci *centre* pro výpočet těžiště vstupní množiny bodů uvedenou v symbolickém kódu v algoritmu 4.7 a externí funkci *orthogonal_vectors* sloužící k výpočtu dvou kolmých vektorů k danému vektoru.

Další použité proměnné jsou analogické k proměnným v algoritmu 4.8. Proměnné a a n_σ jsou vektory kolmé k vektoru n_ρ. Proměnné Δn_ρ ($\Delta n_\rho\_x,y,z$) a Δn_σ ($\Delta n_\sigma\_x,y,z$) slouží k výpočtu numerické parciální derivace, n_ρ^{STEP}, n_σ^{STEP} a a_{STEP} jsou aktuální tři navzájem kolmé vektory v jednom kroku iterace. Proměnné *error*$_\Delta$

a $error_{STEP}$ jsou hodnoty chybové funkce. Proměnné ∂n_ρ^{STEP} a ∂n_σ^{STEP} jsou numericky spočítané parciální derivace funkce *error_function_two_planes* podle normálového vektoru n_ρ (myšleno po souřadnicích) v n_ρ^{STEP} a podle normálového vektoru n_σ (myšleno po souřadnicích) v n_σ^{STEP}, ∇ je gradient funkce *error_function_two_planes* pro hodnoty n_ρ^{STEP} nebo n_σ^{STEP}. Proměnné new_n_ρ a new_n_σ jsou nové vektory minimalizujících posloupností získané střídavě modifikovanou metodou největšího spádu v jednom kroku minimalizace. Vektor *new_a* je kolmý k vektorům new_n_ρ a new_n_σ. Význam dalších použitých proměnných (p, φ, ω, β, aux_n_ρ, aux_n_σ, $coeff_1$, $coeff_2$, ...) je zřejmý z popisu transformací v předchozím textu. Iterační proces je ukončen, pokud je splněna jedna z podmínek.

V pseudokódu používáme ještě globální proměnné δ, ε, λ a $max\_iteration$, které určují nastavení metody minimalizace. Vhodná je například volba

$\delta = 0,0001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,0001$
$max\_iteration = 100$.

---

function *search_symmetry_two_planes*(n_ρ): **pair of sequences**

1: $T \leftarrow centre(X)$
2: $(a, n_\sigma) \leftarrow orthogonal\_vectors(n_\rho)$
3: $plot(plane, T, n_\rho)$
%{vykreslení počáteční polohy roviny s určujícím bodem T a normálovým vektorem n_ρ}%
4: $plot(plane, T, n_\sigma)$
%{vykreslení počáteční polohy roviny s určujícím bodem T a normálovým vektorem n_σ}%
5: $\Delta n_\rho \leftarrow n_\rho$
6: $n_\rho^{STEP} \leftarrow n_\rho$
7: $\Delta n_\sigma \leftarrow n_\sigma$
8: $n_\sigma^{STEP} \leftarrow n_\sigma$
9: $a_{STEP} \leftarrow a$
10: $iteration \leftarrow 0$
11: $sequence\_vectors\_n_\rho[1] \leftarrow n_\rho$
12: $sequence\_vectors\_n_\sigma[1] \leftarrow n_\sigma$
13: **while** *True* **do** %{ukončení cyklu řízeno uvnitř}
14: $iteration \leftarrow iteration + 1$
15: $\Delta n_\rho\_x \leftarrow \Delta n_\rho + (\delta, 0, 0)$
16: $\Delta n_\rho\_y \leftarrow \Delta n_\rho + (0, \delta, 0)$
17: $\Delta n_\rho\_z \leftarrow \Delta n_\rho + (0, 0, \delta)$
18: $error_\Delta \leftarrow error\_function\_two\_planes(\Delta n_\rho, n_\sigma^{STEP})$
19: $error_{STEP} \leftarrow error\_function\_two\_planes(n_\rho^{STEP}, n_\sigma^{STEP})$
20: $\partial n_\rho^{STEP} \leftarrow (error_\Delta - error_{STEP}) / \delta$

%{parciální derivace funkce *error_function_two_planes* podle vektoru n_ρ v n_ρ^{STEP} }%

21: $\nabla \leftarrow (\partial n_\rho^{STEP})$

%{gradient funkce *error_function_two_planes* }%

22: $accuracy \leftarrow norm(\nabla)$

23: **if** $accuracy \le \varepsilon$ **or** $iteration \ge max\_iteration$ **then**

%{podmínka ukončení iterace}%

24: **break**

25: $new\_n_\rho \leftarrow n_\rho^{STEP} - \lambda \cdot \nabla$

26: $new\_n_\rho \leftarrow new\_n_\rho / \left\| new\_n_\rho \right\|$

27: $p \leftarrow (new\_n_\rho \cdot n_\rho^{STEP})\, n_\rho^{STEP} + (new\_n_\rho \cdot a_{STEP})\, a_{STEP}$

28: $\varphi \leftarrow \arccos \dfrac{p \cdot n_\rho^{STEP}}{\left\| p \right\| \left\| n_\rho^{STEP} \right\|}$

29: $coeff_1 \leftarrow new\_n_\rho \cdot n_\rho^{STEP}$

30: $coeff_2 \leftarrow new\_n_\rho \cdot a_{STEP}$

31: **if** $(coeff_1 > 0$ **and** $coeff_2 < 0)$ **or** $(coeff_1 < 0$ **and** $coeff_2 < 0)$ **then**

32: $\varphi \leftarrow -\varphi$

33: $aux\_n_\rho \leftarrow n_\rho^{STEP} \cos\varphi + a_{STEP} \sin\varphi$

34: $new\_a \leftarrow -n_\rho^{STEP} \sin\varphi + a_{STEP} \cos\varphi$

35: $\omega \leftarrow \arccos \dfrac{new\_n_\rho \cdot aux\_n_\rho}{\left\| new\_n_\rho \right\| \left\| aux\_n_\rho \right\|}$

36: $\beta \leftarrow \arccos \dfrac{n_\sigma^{STEP} \cdot new\_n_\rho}{\left\| n_\sigma^{STEP} \right\| \left\| new\_n_\rho \right\|}$

37: **if** $\beta < \dfrac{\pi}{2}$ **then**

38: $\omega \leftarrow -\omega$

39: $new\_n_\sigma \leftarrow n_\sigma^{STEP} \cos\omega + aux\_n_\rho \sin\omega$

40: $new\_n_\rho \leftarrow -n_\sigma^{STEP} \sin\omega + aux\_n_\rho \cos\omega$

41: $sequence\_vectors\_n_\rho[iteration] \leftarrow new\_n_\rho$

42: $sequence\_vectors\_n_\sigma[iteration] \leftarrow new\_n_\sigma$

43: $\Delta n_\rho \leftarrow new\_n_\rho$

44: $n_\rho^{STEP} \leftarrow new\_n_\rho$

45: $\Delta n_\sigma \leftarrow new\_n_\sigma$

46: $n_\sigma^{STEP} \leftarrow new\_n_\sigma$

47: $a_{STEP} \leftarrow new\_a$

48: $plot(plane, T, n_\rho^{STEP})$

%{vykreslení aktuální polohy roviny s určujícím bodem T a normálovým vektorem n_ρ^{STEP}, animace minimalizační metody}%

49: $plot(plane, T, n_\sigma^{STEP})$

%{vykreslení aktuální polohy roviny s určujícím bodem T a normálovým

vektorem n_σ^{STEP} , animace minimalizační metody}%

50: $\Delta n_\sigma\_x \leftarrow \Delta n_\sigma + (\delta, 0, 0)$

51: $\Delta n_\sigma\_y \leftarrow \Delta n_\sigma + (0, \delta, 0)$

52: $\Delta n_\sigma\_z \leftarrow \Delta n_\sigma + (0, 0, \delta)$

53: $error_\Delta \leftarrow error\_function\_two\_planes(n_\rho^{STEP}, \Delta n_\sigma)$

54: $error_{STEP} \leftarrow error\_function\_two\_planes(n_\rho^{STEP}, n_\sigma^{STEP})$

55: $\partial n_\sigma^{STEP} \leftarrow \left(error_\Delta - error_{STEP}\right) / \delta$

%{parciální derivace funkce *error_function_two_planes* podle vektoru n_σ v n_σ^{STEP} }%

56: $\nabla \leftarrow (\partial n_\sigma^{STEP})$

%{gradient funkce *error_function_two_planes* }%

57: $accuracy \leftarrow norm(\nabla)$

58: **if** $accuracy \le \varepsilon$ **or** $iteration \ge max\_iteration$ **then**

%{podmínka ukončení iterace}%

59: **break**

60: $new\_n_\sigma \leftarrow n_\sigma^{STEP} - \lambda \cdot \nabla$

61: $new\_n_\sigma \leftarrow new\_n_\sigma / \left\| new\_n_\sigma \right\|$

62: $p \leftarrow (new\_n_\sigma \cdot a_{STEP})\, a_{STEP} + (new\_n_\sigma \cdot n_\sigma^{STEP})\, n_\sigma^{STEP}$

63: $\varphi \leftarrow \arccos \dfrac{p \cdot n_\sigma^{STEP}}{\left\| p \right\| \left\| n_\sigma^{STEP} \right\|}$

64: $coeff_1 \leftarrow new\_n_\sigma \cdot a_{STEP}$

65: $coeff_2 \leftarrow new\_n_\sigma \cdot n_\sigma^{STEP}$

66: **if** $(coeff_1 > 0$ **and** $coeff_2 > 0)$ **or** $(coeff_1 > 0$ **and** $coeff_2 < 0)$ **then**

67: $\varphi \leftarrow -\varphi$

68: $new\_a \leftarrow a_{STEP} \cos\varphi + n_\sigma^{STEP} \sin\varphi$

69: $aux\_n_\sigma \leftarrow -a_{STEP} \sin\varphi + n_\sigma^{STEP} \cos\varphi$

70: $\omega \leftarrow \arccos \dfrac{new\_n_\sigma \cdot aux\_n_\sigma}{\left\| new\_n_\sigma \right\| \left\| aux\_n_\sigma \right\|}$

71: $\beta \leftarrow \arccos \dfrac{new\_n_\sigma \cdot n_\rho^{STEP}}{\left\| new\_n_\rho \right\| \left\| n_\sigma^{STEP} \right\|}$

72: **if** $\beta > \dfrac{\pi}{2}$ **then**

73: $\omega \leftarrow -\omega$

74: $new\_n_\sigma \leftarrow aux\_n_\sigma \cos\omega + n_\rho^{STEP} \sin\omega$

75: $new\_n_\rho \leftarrow -aux\_n_\sigma \sin\omega + n_\rho^{STEP} \cos\omega$

76: $sequence\_vectors\_n_\rho[iteration] \leftarrow new\_n_\rho$

77: $sequence\_vectors\_n_\sigma[iteration] \leftarrow new\_n_\sigma$

78: $\Delta n_\rho \leftarrow new\_n_\rho$

79: $n_\rho^{STEP} \leftarrow new\_n_\rho$

80: $\Delta n_\sigma \leftarrow new\_n_\sigma$

81: $n_\sigma^{STEP} \leftarrow new\_n_\sigma$

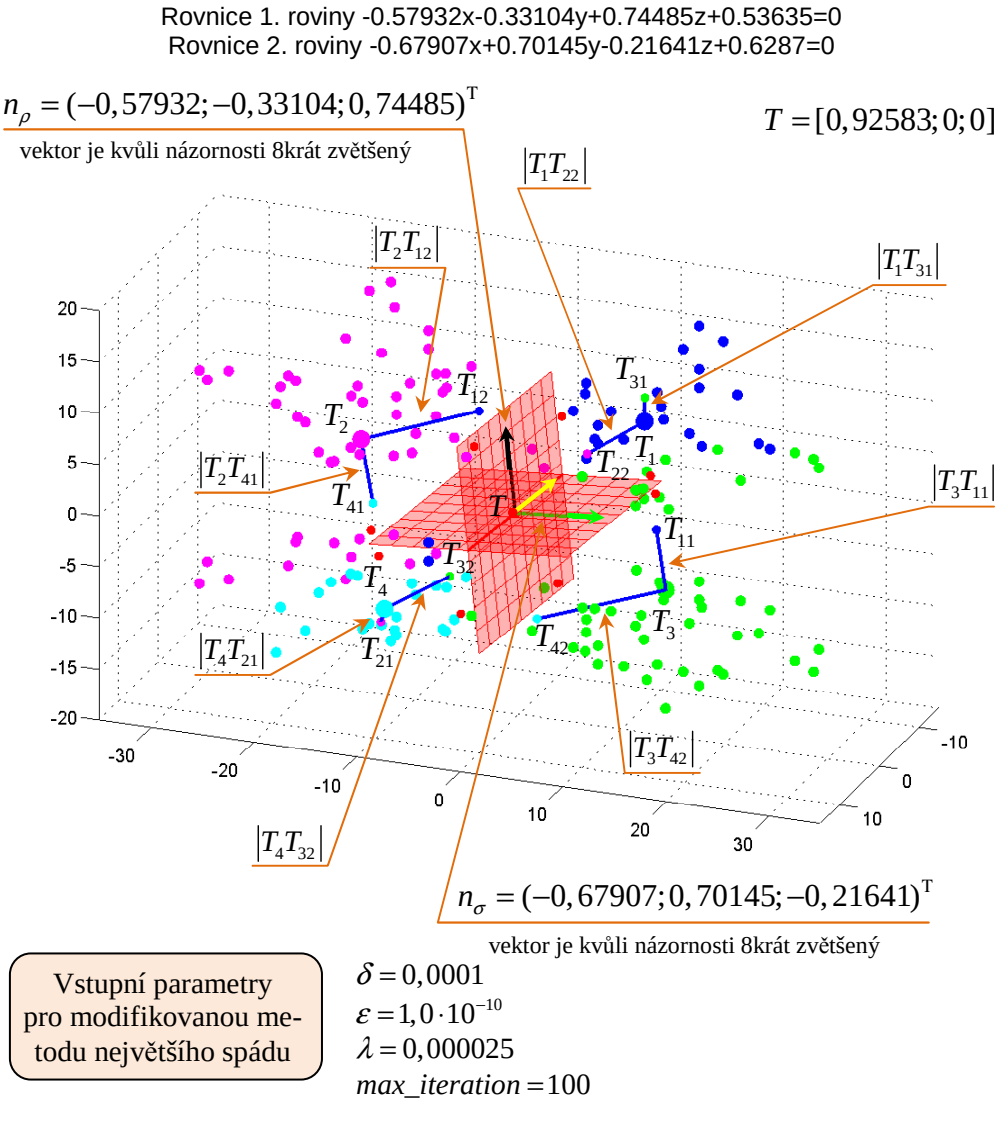
```
82:     a_STEP ← new_a
83:     plot(plane, T, n_ρ^STEP)
        %{vykreslení aktuální polohy roviny s určujícím bodem T a normálovým
        vektorem n_ρ^STEP, animace minimalizační metody}%
84:     plot(plane, T, n_σ^STEP)
        %{vykreslení aktuální polohy roviny s určujícím bodem T a normálovým
        vektorem n_σ^STEP, animace minimalizační metody}%
85: plot(plane, T, n_ρ^STEP)
    %{vykreslení odhadu polohy hledané roviny s určujícím bodem T a normálovým
    vektorem n_ρ^STEP }%
86: plot(plane, T, n_σ^STEP)
    %{vykreslení odhadu polohy hledané roviny s určujícím bodem T a normálovým
    vektorem n_σ^STEP }%
87: plot(X)
    %{vykreslení bodů vstupní množiny}%
88: return (sequence_vectors_n_ρ, sequence_vectors_n_σ)

function orthogonal_vectors(u): pair of vectors
1:  a ← (1,1,1)
2:  if u[1] ≠ 0 then
3:      a[1] ← (−u[2] − u[3]) / u[1]
4:  else
5:      if u[2] ≠ 0 then
6:          a[2] ← (−u[1] − u[3]) / u[2]
7:      else
8:          if u[3] ≠ 0 then
9:              a[3] ← (−u[1] − u[2]) / u[3]
10:         else
11:             return {failure}
12: a ← a / ||a||
13: b ← u × a
14: b ← b / ||b||
15: if (a × b) · u < 0 then
16:     a ← −a
        %{vektory a, b, u jsou jednotkové a tvoří pravotočivou bázi}%
17: return (a, b)
```

Hledání dvou rovinových symetrií s dvěma navzájem kolmými rovinami demonstrujme na příkladě počítačově generované množiny.

Příklad 4.6: *HLEDÁNÍ DVOU NAVZÁJEM KOLMÝCH ROVIN SYMETRIÍ BODOVÉ MNOŽINY*. Navržený algoritmus demonstrujme na příkladě experimentální bodové množiny. Předpokládejme, že je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$

v eukleidovském prostoru E_3, která je symetrická podle nějakých dvou vzájemně kolmých rovin. Pro lepší názornost volíme záměrně řídkou množinu bodů.



Vstupní parametry pro modifikovanou metodu největšího spádu

$\delta = 0,0001$
$\varepsilon = 1,0 \cdot 10^{-10}$
$\lambda = 0,000025$
$max\_iteration = 100$

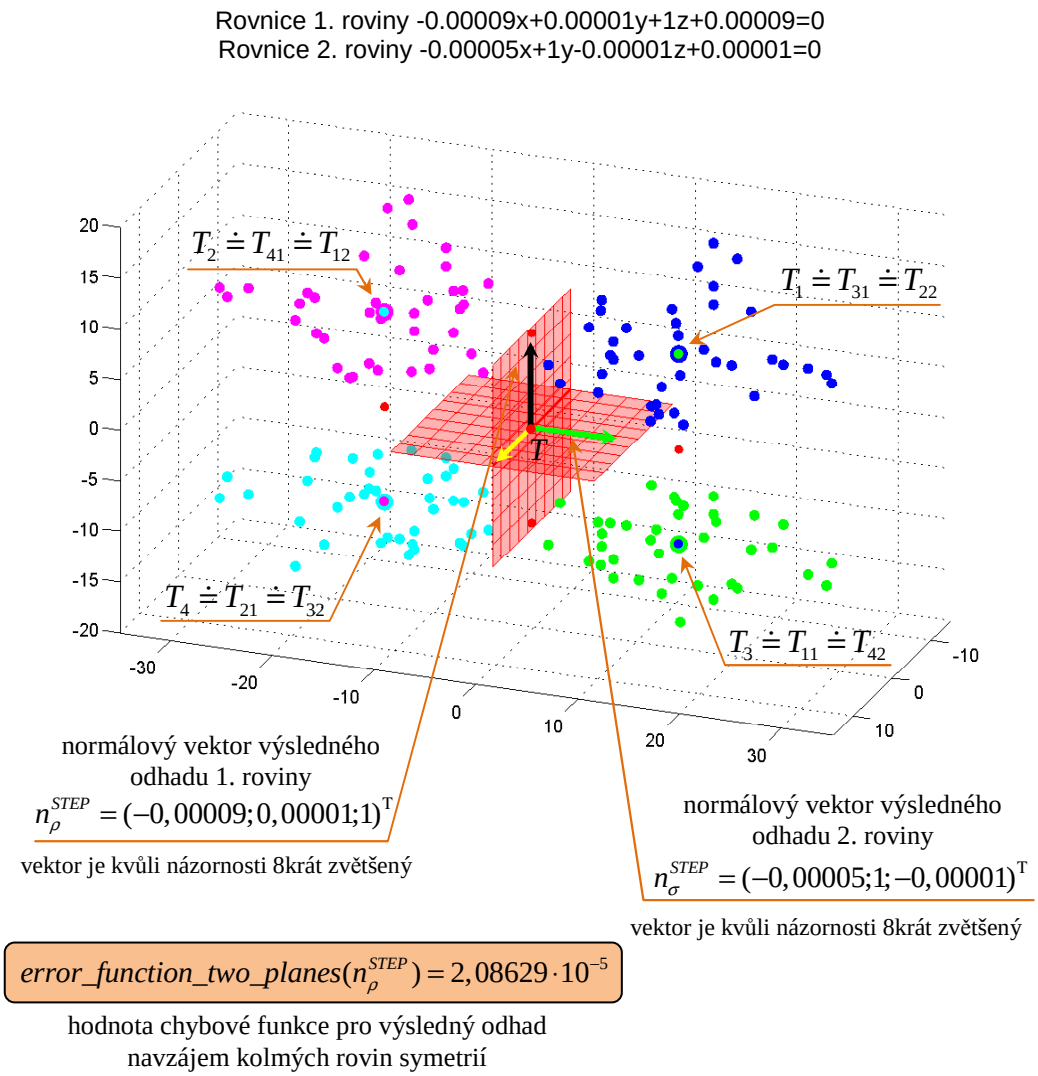
Obrázek 4.85: Bodová množina symetrická podle dvou vzájemně kolmých rovin a počáteční poloha rovin – znázornění principu navržené metody a vstupní parametry minimalizace

Na obrázku 4.85 je zakreslena vstupní množina bodů, počáteční poloha navzájem kolmých rovin, které se postupně optimalizují, a předloženy jsou jejich obecné rovnice. Body v jednotlivých kvadrantech jsou barevně odlišeny. Ve všech čtyřech kvadrantech jsou vyznačena těžiště T_1, T_2, T_3 a T_4 a jejich obrazy T_{11}, T_{12}, T_{21}, T_{22}, T_{31}, T_{32}, T_{41} a T_{42} v obou rovinových symetriích se zadanými rovinami. Modře jsou vyznačeny vzdálenosti těžišť a obrazů těžišť ze sousedních kvadrantů. Na obrázku 4.86 je vykreslena výsledná poloha rovin a vypsané jsou jejich obecné rovnice. Obrázek 4.87 ilustruje postupné vylepšování polohy navzájem kolmých rovin minimalizací chybové funkce, přičemž se roviny vykreslují pouze v několika pozicích. Dosta-

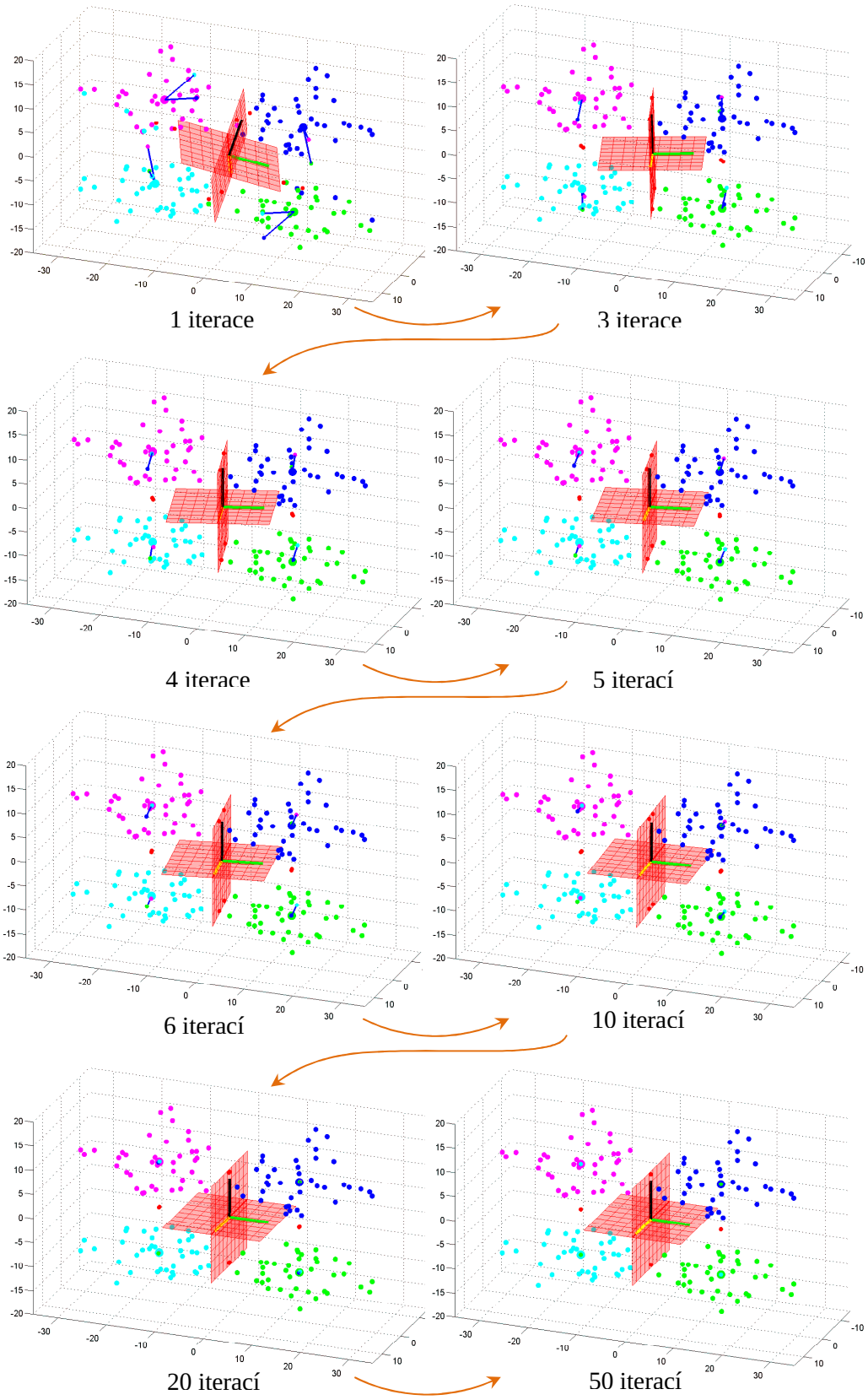
tečně dobrý odhad rovin symetrií dostáváme pro uvedenou volbu vstupních parametrů metody (100 iterací). Vyžadujeme-li přesnější výsledek, opět je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroku λ. Pro minimalizaci chybové funkce je vykreslena také závislost hodnoty chybové funkce na počtu iterací, viz obrázek 4.88. Při zobrazení jsme užili logaritmického měřítka na ose *y*, neboť v prvním kroku minimalizace vychází velká hodnota chybové funkce. Všechny vypisované číselné hodnoty v textu vždy zaokrouhlujeme na pět desetinných míst.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

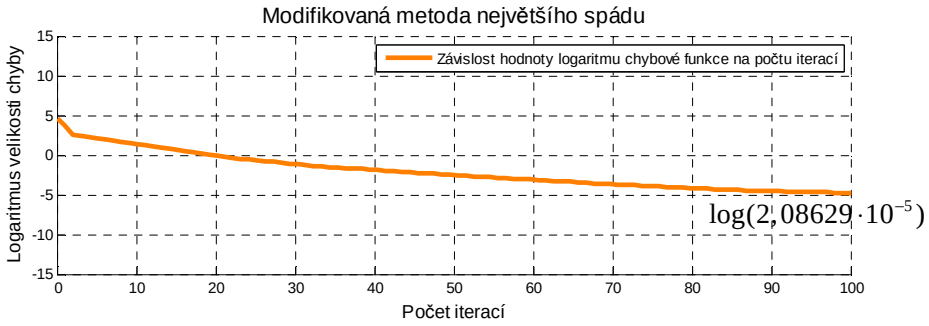
`symetrie_dve_roviny.m` (programy/symetrie).



Obrázek 4.86: Výsledný odhad navzájem kolmých rovin symetrií



Obrázek 4.87: Postupná optimalizace dvou navzájem kolmých rovin minimalizací chybové funkce modifikovanou metodou největšího spádu



Obrázek 4.88: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací (logaritmické měřítko na ose *y*)

Další příklady hledání dvou rovinových symetrií s dvěma navzájem kolmými rovinami budou uvedeny pro reálné množiny v závěrečné kapitole. Půjde o množiny obsahující již veliké množství bodů.

4.4.3 Rovinové symetrie se třemi navzájem kolmými rovinami

Poslední typ bodové množiny, který budeme uvažovat, je množina rovinově symetrická podle tří navzájem kolmých rovin. Představme navržený postup hledání těchto rovinových symetrií.

Mějme dánu neprázdnou množinu $\mathcal{X}$ n bodů v eukleidovském prostoru E_3, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Nechť je tato vstupní bodová množina přesně symetrická podle tří navzájem kolmých rovin.

Roviny symetrie hledáme iteračně s použitím odvozené minimalizace modifikovanou metodou největšího spádu. V prvním kroku algoritmu určíme opět těžiště T vstupní bodové množiny $\mathcal{X}$ podle (4.11). Do těžiště T umístíme tři navzájem kolmé roviny ρ, σ a τ, jejichž polohy budeme v každém kroku algoritmu aktualizovat, přičemž počáteční polohy určené jednotkovými normálovými vektory n_ρ, n_σ a n_τ volíme libovolně. Těžiště T je tedy společným bodem těchto tří rovin. Všechny tři hledané roviny symetrií těžištěm procházejí (podle tvrzení 4.1), proto budeme v dalších krocích algoritmu měnit pouze normálové vektory n_ρ, n_σ a n_τ rovin ρ, σ a τ, dokud se roviny dostatečně nepřiblíží rovinám symetrií podle definice 4.6.

Počáteční volba jednotkových normálových vektorů n_ρ, n_σ a n_τ je stejná jako v předchozím oddíle o rovinových symetriích se dvěma navzájem kolmými rovinami. Směr jednotkového normálového vektoru n_ρ tedy volíme zcela libovolně, normálové vektory n_σ a n_τ dourčíme tak, aby vektory n_ρ, n_σ a n_τ tvořily pravotočivou ortonormální bázi. Z množiny kolmých vektorů k vektoru n_ρ vybíráme vektory n_σ a n_τ stejně jako v (4.17) a (4.18), místo vektoru a máme nyní vektor n_τ.

Výpočet se bude lišit až v určování chybové funkce. V každém kroku rozdělí roviny ρ, σ a τ v příslušné aktuální poloze eukleidovský prostor E_3 na osm oktantů. Pro každý oktant určíme, které body ze vstupní množiny v něm leží. Množiny bodů v jednotlivých ok-

tantech značíme $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ a $\mathcal{M}_8$. Nechť jsou roviny ρ, σ a τ dány obecnými rovnicemi ve tvaru

$$n_\rho \cdot (X - T) = 0,\ n_\sigma \cdot (X - T) = 0 \text{ a } n_\tau \cdot (X - T) = 0 \tag{4.22}$$

kde T je těžiště vstupní bodové množiny, n_ρ je jednotkový normálový vektor roviny ρ, tj. $\|n_\rho\| = 1$, n_σ je jednotkový normálový vektor roviny σ, tj. $\|n_\sigma\| = 1$, a n_τ je jednotkový normálový vektor roviny τ, tj. $\|n_\tau\| = 1$. Potom bod $X_i = [X_i^x, X_i^y, X_i^z]$ vstupní množiny $\mathcal{X}$ přidáme do množiny $\mathcal{M}_1$ ($\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ nebo $\mathcal{M}_8$), leží-li v 1. (2., 3., 4., 5., 6., 7. nebo 8.) oktantu určených rovinami ρ, σ a τ. Bodové množiny $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ a $\mathcal{M}_8$ tedy definujeme následovně

$$\begin{aligned}
\mathcal{M}_1 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) > 0 \wedge n_\tau \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_2 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) > 0 \wedge n_\tau \cdot (X_i - T) < 0, i = 1,2,...,n\},\\
\mathcal{M}_3 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) < 0 \wedge n_\tau \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_4 &= \{X_i; n_\rho \cdot (X_i - T) > 0 \wedge n_\sigma \cdot (X_i - T) < 0 \wedge n_\tau \cdot (X_i - T) < 0, i = 1,2,...,n\},\\
\mathcal{M}_5 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) > 0 \wedge n_\tau \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_6 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) > 0 \wedge n_\tau \cdot (X_i - T) < 0, i = 1,2,...,n\},\\
\mathcal{M}_7 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) < 0 \wedge n_\tau \cdot (X_i - T) > 0, i = 1,2,...,n\},\\
\mathcal{M}_8 &= \{X_i; n_\rho \cdot (X_i - T) < 0 \wedge n_\sigma \cdot (X_i - T) < 0 \wedge n_\tau \cdot (X_i - T) < 0, i = 1,2,...,n\}.
\end{aligned} \tag{4.23}$$

Body vstupní množiny, které leží v rovinách ρ, σ a τ, nemusíme uvažovat, protože nezmění hodnotu chybové funkce.

Dále postupujeme zcela analogicky. Určíme těžiště bodových množin $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ a $\mathcal{M}_8$, které po řadě značíme T_1, T_2, T_3, T_4, T_5, T_6, T_7 a T_8 a zobrazíme je v rovinových symetriích s rovinami ρ, σ a τ, tj. $f_\rho(T_1) = T_{11}$, $f_\sigma(T_1) = T_{12}$, $f_\tau(T_1) = T_{13}$, $f_\rho(T_2) = T_{21}$, $f_\sigma(T_2) = T_{22}$, $f_\tau(T_2) = T_{23}$, $f_\rho(T_3) = T_{31}$, $f_\sigma(T_3) = T_{32}$, $f_\tau(T_3) = T_{33}$, $f_\rho(T_4) = T_{41}$, $f_\sigma(T_4) = T_{42}$, $f_\tau(T_4) = T_{43}$, $f_\rho(T_5) = T_{51}$, $f_\sigma(T_5) = T_{52}$, $f_\tau(T_5) = T_{53}$, $f_\rho(T_6) = T_{61}$, $f_\sigma(T_6) = T_{62}$, $f_\tau(T_6) = T_{63}$, $f_\rho(T_7) = T_{71}$, $f_\sigma(T_7) = T_{72}$, $f_\tau(T_7) = T_{73}$, $f_\rho(T_8) = T_{81}$, $f_\sigma(T_8) = T_{82}$, $f_\tau(T_8) = T_{83}$. Pokud se roviny ρ, σ a τ v aktuální poloze rovnají přímo rovinám symetrií, odpovídají si těžiště T_1, T_2, T_3, T_4, T_5, T_6, T_7 a T_8 v rovinových symetriích s rovinami ρ, σ a τ, tj. $T_1 = T_{51} = T_{32} = T_{23}$, $T_2 = T_{61} = T_{42} = T_{13}$, $T_3 = T_{71} = T_{12} = T_{43}$, $T_4 = T_{81} = T_{22} = T_{33}$, $T_5 = T_{11} = T_{72} = T_{63}$, $T_6 = T_{21} = T_{82} = T_{53}$, $T_7 = T_{31} = T_{52} = T_{83}$ a $T_8 = T_{41} = T_{62} = T_{73}$. Minimalizací chybové funkce chceme dosáhnout tohoto stavu. Určujeme proto vzdálenosti těžiště a trojice zobrazených těžišť ze sousedních oktantů v rovinových symetriích s rovinami ρ, σ a τ. Vzdálenosti jsou $distance_1 = |T_1T_{51}|$, $distance_2 = |T_1T_{32}|$, $distance_3 = |T_1T_{23}|$, $distance_4 = |T_2T_{61}|$, $distance_5 = |T_2T_{42}|$, $distance_6 = |T_2T_{13}|$, $distance_7 = |T_3T_{71}|$, $distance_8 = |T_3T_{12}|$, $distance_9 = |T_3T_{43}|$, $distance_{10} = |T_4T_{81}|$, $distance_{11} = |T_4T_{22}|$, $distance_{12} = |T_4T_{33}|$, $distance_{13} = |T_5T_{11}|$,

$distance_{14} = |T_5T_{72}|$, $distance_{15} = |T_5T_{63}|$, $distance_{16} = |T_6T_{21}|$, $distance_{17} = |T_6T_{82}|$, $distance_{18} = |T_2T_{53}|$, $distance_{19} = |T_7T_{31}|$, $distance_{20} = |T_7T_{52}|$, $distance_{21} = |T_7T_{83}|$, $distance_{22} = |T_8T_{41}|$, $distance_{23} = |T_8T_{62}|$ a $distance_{24} = |T_8T_{73}|$.

Obdobně jako v předchozích oddílech zavádíme chybovou funkci, značme ji δ^2, a definujeme ji následovně

$$\begin{aligned}\delta^2 = &(distance_1^2 + distance_{13}^2)\cdot(1+|p_1 - p_5|) + (distance_2^2 + distance_8^2)\cdot(1+|p_1 - p_3|) + \\ &+ (distance_3^2 + distance_6^2)\cdot(1+|p_1 - p_2|) + (distance_4^2 + distance_{16}^2)\cdot(1+|p_2 - p_6|) + \\ &+ (distance_5^2 + distance_{11}^2)\cdot(1+|p_2 - p_4|) + (distance_7^2 + distance_{19}^2)\cdot(1+|p_3 - p_7|) + \\ &+ (distance_9^2 + distance_{12}^2)\cdot(1+|p_3 - p_4|) + (distance_{10}^2 + distance_{22}^2)\cdot(1+|p_4 - p_8|) + \\ &+ (distance_{14}^2 + distance_{20}^2)\cdot(1+|p_5 - p_7|) + (distance_{15}^2 + distance_{18}^2)\cdot(1+|p_5 - p_6|) + \\ &+ (distance_{17}^2 + distance_{23}^2)\cdot(1+|p_6 - p_8|) + (distance_{21}^2 + distance_{24}^2)\cdot(1+|p_7 - p_8|)\end{aligned} \tag{4.24}$$

kde $p_j = |\mathcal{M}_j|$.

Součet druhých mocnin dvojic vzdáleností vždy násobíme konstantou $1+|p_{1,2,3,4,5,6,7} - p_{2,3,4,5,6,7,8}|$, čímž zajistíme lepší průběh minimalizace chybové funkce. Tato konstanta je tím větší, čím méně si odpovídá počet bodů ve dvou sousedních oktantech určených rovinami ρ, σ a τ v příslušné aktuální poloze. Tímto způsobem dospějeme k tomu, že si odpovídají počty bodů v sousedních oktantech, a navíc v sousedních oktantech korespondují také těžiště. Minimalizaci chybové funkce z (4.24) provádíme opět modifikovanou metodu největšího spádu s konstantní délkou kroku.

Chybová funkce má celkem devět parametrů – souřadnice normálového vektoru n_ρ roviny ρ, souřadnice normálového vektoru n_σ roviny σ a souřadnice normálového vektoru n_τ roviny τ, jedná se tedy o reálnou funkci devíti reálných proměnných. Ovšem v daném kroku minimalizace vždy aktualizujeme střídavě pouze tři parametry, tj. souřadnice jednoho z normálových vektorů. Stupňů volnosti je ve skutečnosti méně, neboť těchto devět parametrů totiž není nezávislých. To je zřejmé, protože vektory n_ρ, n_σ a n_τ jsou navzájem kolmé. Ke zvolenému vektoru n_ρ určujeme vektor n_σ jako kolmý a vektor n_τ je potom až na směr už jednoznačně daný.

Stejně jako v předchozím oddíle budeme v daném kroku minimalizace chybové funkce střídavě aktualizovat pouze souřadnice normálových vektorů n_ρ a n_σ a provádět příslušné transformace vzájemně kolmých vektorů n_ρ, n_σ a n_τ do nových poloh. Tato metoda dává možnost aktualizovat střídavě všechny tři vektory, ovšem při reálném testování této varianty jsme neobdrželi příliš rozdílné výsledky.

V následujícím symbolickém kódu je jako algoritmus 4.11 popsán výpočet chybové funkce. Chybovou funkci značíme jako *error_function_three_planes* a její hodnotu pro konkrétních devět parametrů jako *error* .

Algoritmus 4.11: *VÝPOČET CHYBOVÉ FUNKCE ERROR_FUNCTION_THREE_-PLANES PRO ROVINOVÉ SYMETRIE SE TŘEMI NAVZÁJEM KOLMÝMI ROVINAMI*. Vstupními parametry chybové funkce jsou jednotkový normálový vektor n_ρ roviny ρ, jednotkový normálový vektor n_σ roviny σ a jednotkový normálový vektor n_τ roviny τ. Výstupem funkce je reálná hodnota *error*.

Mějme dánu množin $\mathcal{X}$ n bodů $\left\{[x_i, y_i, z_i]\right\}_{i=1}^{n}$ v eukleidovském prostoru E_3, která je symetrická podle tří navzájem kolmých rovin. Matice vstupních bodů je v pseudokódu zadávána jako globální proměnná, značíme ji X. Algoritmus využívá tři externí funkce *subspace_three_planes*, *centre* a *mirror*. Funkce *subspace_three_planes* slouží k rozdělení vstupních bodů na osm množin bodů podle (4.23), přičemž oktanty jsou určené rovinami ρ, σ a τ. Funkce *centre* a *mirror* jsou popsané v algoritmu 4.7.

Pseudokód využívá několik proměnných. Proměnné $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ a $\mathcal{M}_8$ jsou množiny bodů v oktantech. Proměnné T_1, T_2, T_3, T_4, T_5, T_6, T_7 a T_8 jsou těžiště těchto bodových množin a proměnné T_{11}, T_{12}, T_{13}, T_{21}, T_{22}, T_{23}, T_{31}, T_{32}, T_{33}, T_{41}, T_{42}, T_{43}, T_{51}, T_{52}, T_{53}, T_{61}, T_{62}, T_{63}, T_{71}, T_{72}, T_{73}, T_{81}, T_{82} a T_{83} jsou jejich obrazy v rovinových symetriích s rovinami ρ, σ a τ. Do proměnných p_j ukládáme počty bodů množin $\mathcal{M}_j$ a do proměnné d vzdálenosti těžiště a zobrazených těžišť v rovinových symetriích s rovinami ρ, σ a τ.

---

function *error_function_three_planes*$(n_\rho, n_\sigma, n_\tau)$: **real**

1: $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6, \mathcal{M}_7, \mathcal{M}_8) \leftarrow$ *subspace_three_planes*$(T, n_\rho, n_\sigma, n_\tau)$

2: $T_1 \leftarrow$ *centre* $(\mathcal{M}_1)$; $T_2 \leftarrow$ *centre* $(\mathcal{M}_2)$

3: $T_3 \leftarrow$ *centre* $(\mathcal{M}_3)$; $T_4 \leftarrow$ *centre* $(\mathcal{M}_4)$

4: $T_5 \leftarrow$ *centre* $(\mathcal{M}_5)$; $T_6 \leftarrow$ *centre* $(\mathcal{M}_6)$

5: $T_7 \leftarrow$ *centre* $(\mathcal{M}_7)$; $T_8 \leftarrow$ *centre* $(\mathcal{M}_8)$

6: $T_{11} \leftarrow mirror(T_1, T, n_\rho)$; $T_{12} \leftarrow mirror(T_1, T, n_\sigma)$; $T_{13} \leftarrow mirror(T_1, T, n_\tau)$

7: $T_{21} \leftarrow mirror(T_2, T, n_\rho)$; $T_{22} \leftarrow mirror(T_2, T, n_\sigma)$; $T_{23} \leftarrow mirror(T_2, T, n_\tau)$

8: $T_{31} \leftarrow mirror(T_3, T, n_\rho)$; $T_{32} \leftarrow mirror(T_3, T, n_\sigma)$; $T_{33} \leftarrow mirror(T_3, T, n_\tau)$

9: $T_{41} \leftarrow mirror(T_4, T, n_\rho)$; $T_{42} \leftarrow mirror(T_4, T, n_\sigma)$; $T_{43} \leftarrow mirror(T_4, T, n_\tau)$

10: $T_{51} \leftarrow mirror(T_5, T, n_\rho)$; $T_{52} \leftarrow mirror(T_5, T, n_\sigma)$; $T_{53} \leftarrow mirror(T_5, T, n_\tau)$

11: $T_{61} \leftarrow mirror(T_6, T, n_\rho)$; $T_{62} \leftarrow mirror(T_6, T, n_\sigma)$; $T_{63} \leftarrow mirror(T_6, T, n_\tau)$

12: $T_{71} \leftarrow mirror(T_7, T, n_\rho)$; $T_{72} \leftarrow mirror(T_7, T, n_\sigma)$; $T_{73} \leftarrow mirror(T_7, T, n_\tau)$

13: $T_{81} \leftarrow mirror(T_8, T, n_\rho)$; $T_{82} \leftarrow mirror(T_8, T, n_\sigma)$; $T_{83} \leftarrow mirror(T_8, T, n_\tau)$

14: $d \leftarrow \Big[\|T_1 - T_{51}\|, \|T_1 - T_{32}\|, \|T_1 - T_{23}\|, \|T_2 - T_{61}\|, \|T_2 - T_{42}\|, \|T_2 - T_{13}\|, \|T_3 - T_{71}\|, \|T_3 - T_{12}\|,$
$\|T_3 - T_{43}\|, \|T_4 - T_{81}\|, \|T_4 - T_{22}\|, \|T_4 - T_{33}\|, \|T_5 - T_{11}\|, \|T_5 - T_{72}\|, \|T_5 - T_{63}\|, \|T_6 - T_{21}\|,$
$\|T_6 - T_{82}\|, \|T_6 - T_{53}\|, \|T_7 - T_{31}\|, \|T_7 - T_{52}\|, \|T_7 - T_{83}\|, \|T_8 - T_{41}\|, \|T_8 - T_{62}\|, \|T_8 - T_{73}\|\Big]$

15: $p_1 \leftarrow |\mathcal{M}_1|$; $p_2 \leftarrow |\mathcal{M}_2|$

16: $p_3 \leftarrow |\mathcal{M}_3|$; $p_4 \leftarrow |\mathcal{M}_4|$

17: $p_5 \leftarrow |\mathcal{M}_5|$; $p_6 \leftarrow |\mathcal{M}_6|$

18: $p_7 \leftarrow |\mathcal{M}_7|$; $p_8 \leftarrow |\mathcal{M}_8|$

19: $error \leftarrow \left[(d[1])^2 + d[13])^2\right] \cdot \left(1 + \left|p_1 - p_5\right|\right) + \left[(d[2])^2 + (d[8])^2\right] \cdot \left(1 + \left|p_1 - p_3\right|\right) +$
$+ \left[(d[3])^2 + (d[6])^2\right] \cdot \left(1 + \left|p_1 - p_2\right|\right) + \left[(d[4])^2 + d[16])^2\right] \cdot \left(1 + \left|p_2 - p_6\right|\right) +$
$+ \left[(d[5])^2 + (d[11])^2\right] \cdot \left(1 + \left|p_2 - p_4\right|\right) + \left[(d[7])^2 + d[19])^2\right] \cdot \left(1 + \left|p_3 - p_7\right|\right) +$
$+ \left[(d[9])^2 + (d[12])^2\right] \cdot \left(1 + \left|p_3 - p_4\right|\right) + \left[(d[10])^2 + d[22])^2\right] \cdot \left(1 + \left|p_4 - p_8\right|\right) +$
$+ \left[(d[14])^2 + (d[20])^2\right] \cdot \left(1 + \left|p_5 - p_7\right|\right) + \left[(d[15])^2 + d[18])^2\right] \cdot \left(1 + \left|p_5 - p_6\right|\right) +$
$+ \left[(d[17])^2 + (d[23])^2\right] \cdot \left(1 + \left|p_6 - p_8\right|\right) + \left[(d[21])^2 + d[24])^2\right] \cdot \left(1 + \left|p_7 - p_8\right|\right)$
20: **return** *error*

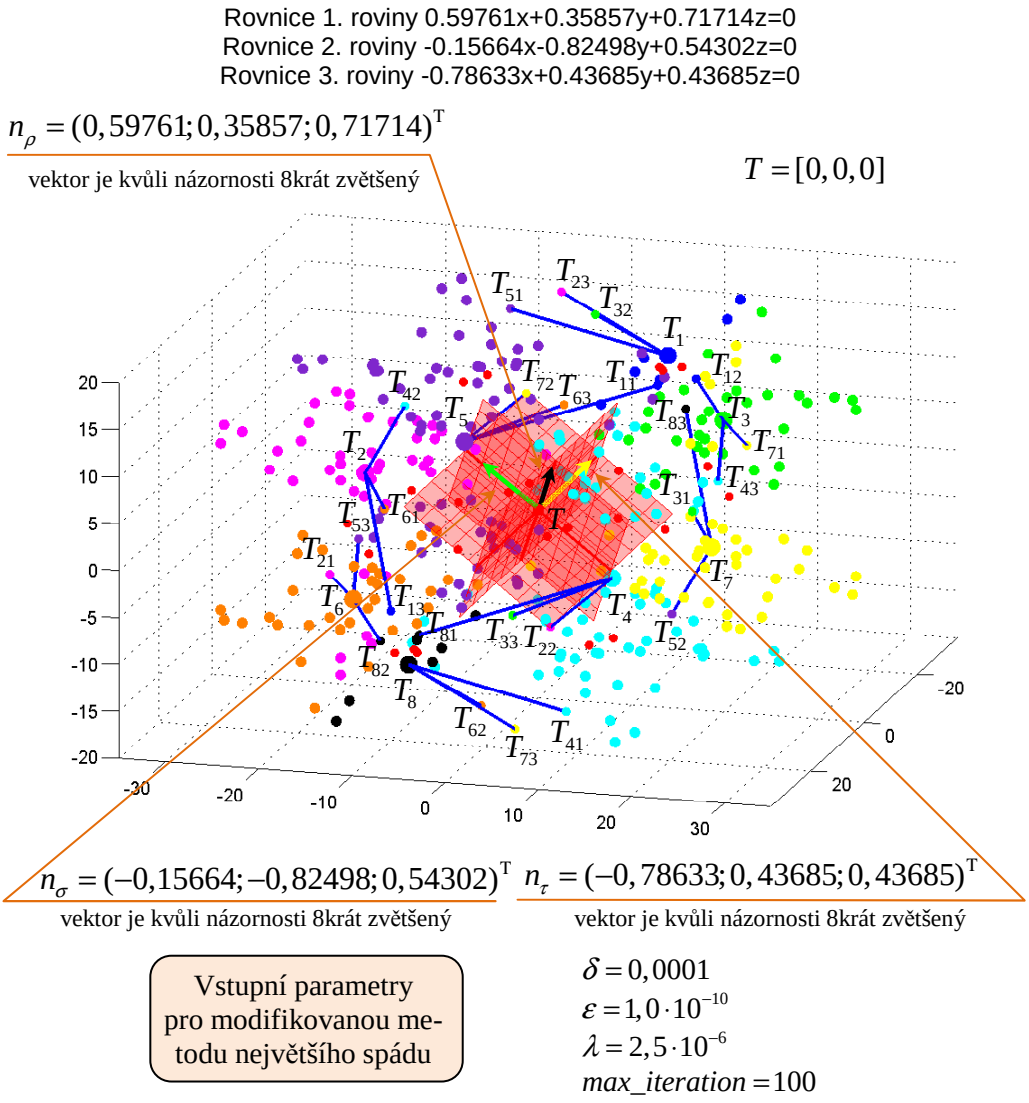
function *subspace_three_planes*$(T, n_\rho, n_\sigma, n_\tau)$: **eight-tuple of sets**
1: $\mathcal{M}_1 \leftarrow \emptyset$; $\mathcal{M}_2 \leftarrow \emptyset$
2: $\mathcal{M}_3 \leftarrow \emptyset$; $\mathcal{M}_4 \leftarrow \emptyset$
3: $\mathcal{M}_5 \leftarrow \emptyset$; $\mathcal{M}_6 \leftarrow \emptyset$
4: $\mathcal{M}_7 \leftarrow \emptyset$; $\mathcal{M}_8 \leftarrow \emptyset$
5: **for** $i = 1, 2, ..., n$ **do**
6: | **if** $n_\rho \cdot (X[i] - T) > 0$ **then**
7: | | **if** $n_\sigma \cdot (X[i] - T) > 0$ **then**
8: | | | **if** $n_\tau \cdot (X[i] - T) > 0$ **then**
9: | | | | $\mathcal{M}_1 \leftarrow \mathcal{M}_1 \cup \{X[i]\}$
10: | | | **else**
11: | | | | $\mathcal{M}_2 \leftarrow \mathcal{M}_2 \cup \{X[i]\}$
12: | | **else**
13: | | | **if** $n_\tau \cdot (X[i] - T) > 0$ **then**
14: | | | | $\mathcal{M}_3 \leftarrow \mathcal{M}_3 \cup \{X[i]\}$
15: | | | **else**
16: | | | | $\mathcal{M}_4 \leftarrow \mathcal{M}_4 \cup \{X[i]\}$
17: | **else**
18: | | **if** $n_\sigma \cdot (X[i] - T) > 0$ **then**
19: | | | **if** $n_\tau \cdot (X[i] - T) > 0$ **then**
20: | | | | $\mathcal{M}_5 \leftarrow \mathcal{M}_5 \cup \{X[i]\}$
21: | | | **else**
22: | | | | $\mathcal{M}_6 \leftarrow \mathcal{M}_6 \cup \{X[i]\}$
23: | | **else**
24: | | | **if** $n_\tau \cdot (X[i] - T) > 0$ **then**
25: | | | | $\mathcal{M}_7 \leftarrow \mathcal{M}_7 \cup \{X[i]\}$
26: | | | **else**
27: | | | | $\mathcal{M}_8 \leftarrow \mathcal{M}_8 \cup \{X[i]\}$
28: **return** $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6, \mathcal{M}_7, \mathcal{M}_8)$

Chybovou funkci popsanou v algoritmu 4.11 dále minimalizujeme modifikovanou metodou největšího spádu s konstantní délkou kroku. Výsledkem minimalizace je potom odhad tří navzájem kolmých rovin symetrií. Proces minimalizace již nebudeme zapisovat sym-

bolickým kódem, neboť je tento kód stejný jako v algoritmu 4.10, stačí zaměnit chybové funkce. To znamená, že místo chybové funkce *error_function_two_planes* použijeme v algoritmu 4.10 chybovou funkci *error_function_three_planes*.

V závěru tohoto oddílu představme proces hledání tří rovinových symetrií s navzájem kolmými rovinami na konkrétním příkladě.

Příklad 4.7: *HLEDÁNÍ TŘÍ NAVZÁJEM KOLMÝCH ROVIN SYMETRIÍ BODOVÉ MNOŽINY*. Funkčnost navrženého algoritmu ověřme na příkladě počítačově generované bodové množiny. Nechť je dána množina $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, která je symetrická podle nějakých tří vzájemně kolmých rovin. Opět volíme záměrně řídkou množinu bodů.

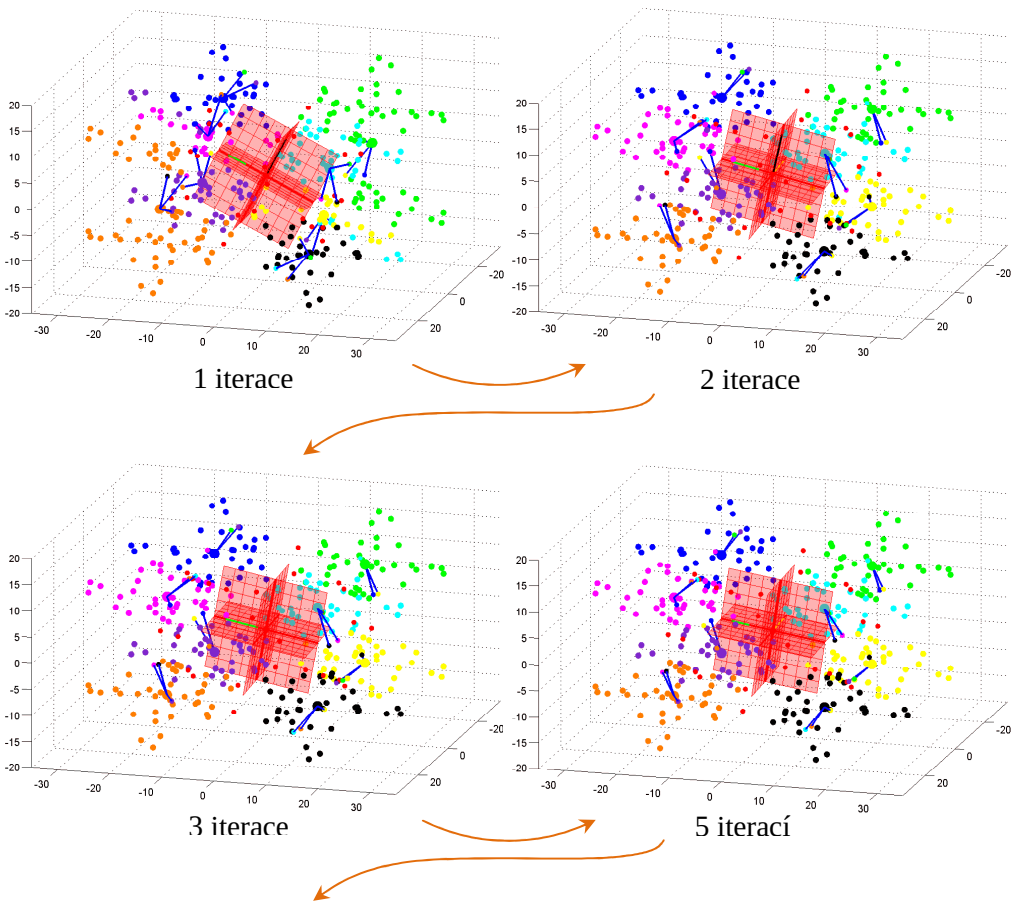


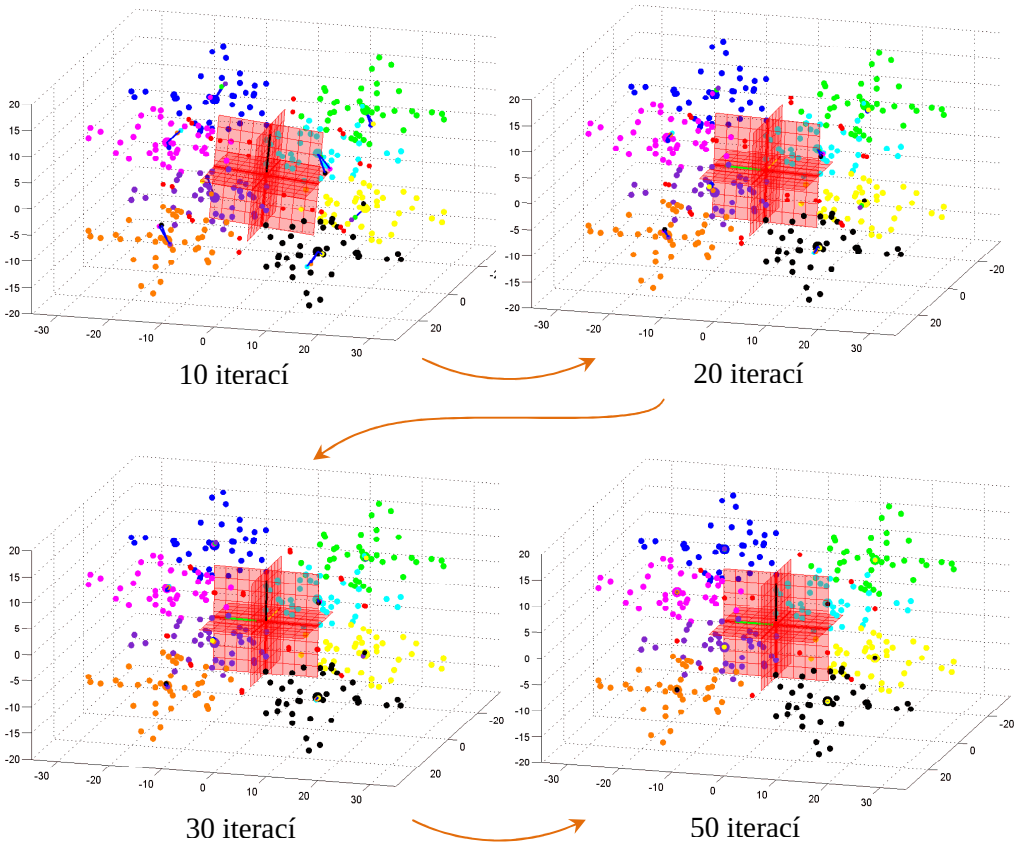
Obrázek 4.89: Bodová množina symetrická podle tří vzájemně kolmých rovin a počáteční poloha rovin – znázornění principu navržené metody a vstupní parametry minimalizace

Na obrázku 4.89 je znázorněna vstupní množina bodů, počáteční poloha navzájem kolmých rovin, které se postupně optimalizují, a předloženy jsou jejich obecné rovnice. Body v jednotlivých oktantech jsou barevně odlišeny. Ve všech oktantech jsou vyznačena těžiště T_1, T_2, T_3, T_4, T_5, T_6, T_7 a T_8 a jejich obrazy T_{11}, T_{12}, T_{13}, T_{21}, T_{22}, T_{23}, T_{31}, T_{32}, T_{33}, T_{41}, T_{42}, T_{43}, T_{51}, T_{52}, T_{53}, T_{61}, T_{62}, T_{63}, T_{71}, T_{72}, T_{73}, T_{81}, T_{82} a T_{83} ve třech rovinových symetriích se zadanými rovinami. Modře jsou zakresleny vzdálenosti těžišť a obrazů těžišť ze sousedních oktantů. Na obrázku 4.90 vidíme postupné vylepšování polohy navzájem kolmých rovin minimalizací chybové funkce, přičemž roviny znázorňujeme pouze v několika pozicích. Rovněž předkládáme funkční závislost hodnoty chybové funkce na počtu iterací, viz obrázek 4.91. Pro zobrazení jsme znovu užili logaritmického měřítka na ose y. Samostatně je vykreslena výsledná poloha rovin a vypsané jsou jejich obecné rovnice, viz obrázek 4.92. Kvalitní odhad rovin symetrií dostáváme pro uvedenou volbu vstupních parametrů metody (100 iterací). Vyžadujeme-li přesnější výsledek, můžeme zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroku λ. Všechny vypisované číselné hodnoty v textu vždy zaokrouhlujeme na pět desetinných míst.

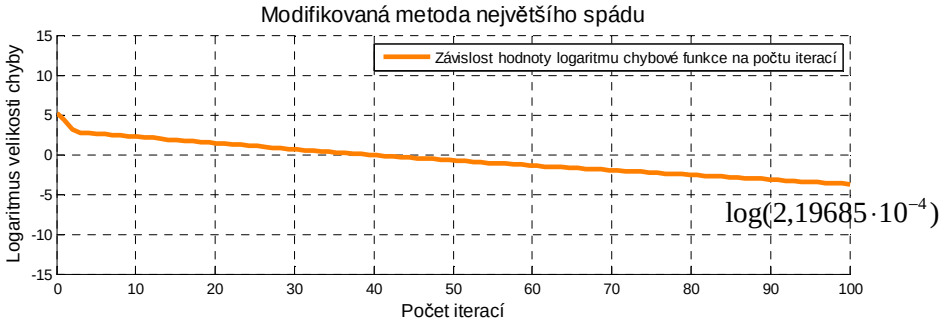
Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

`symetrie_tri_roviny.m` (programy/symetrie).

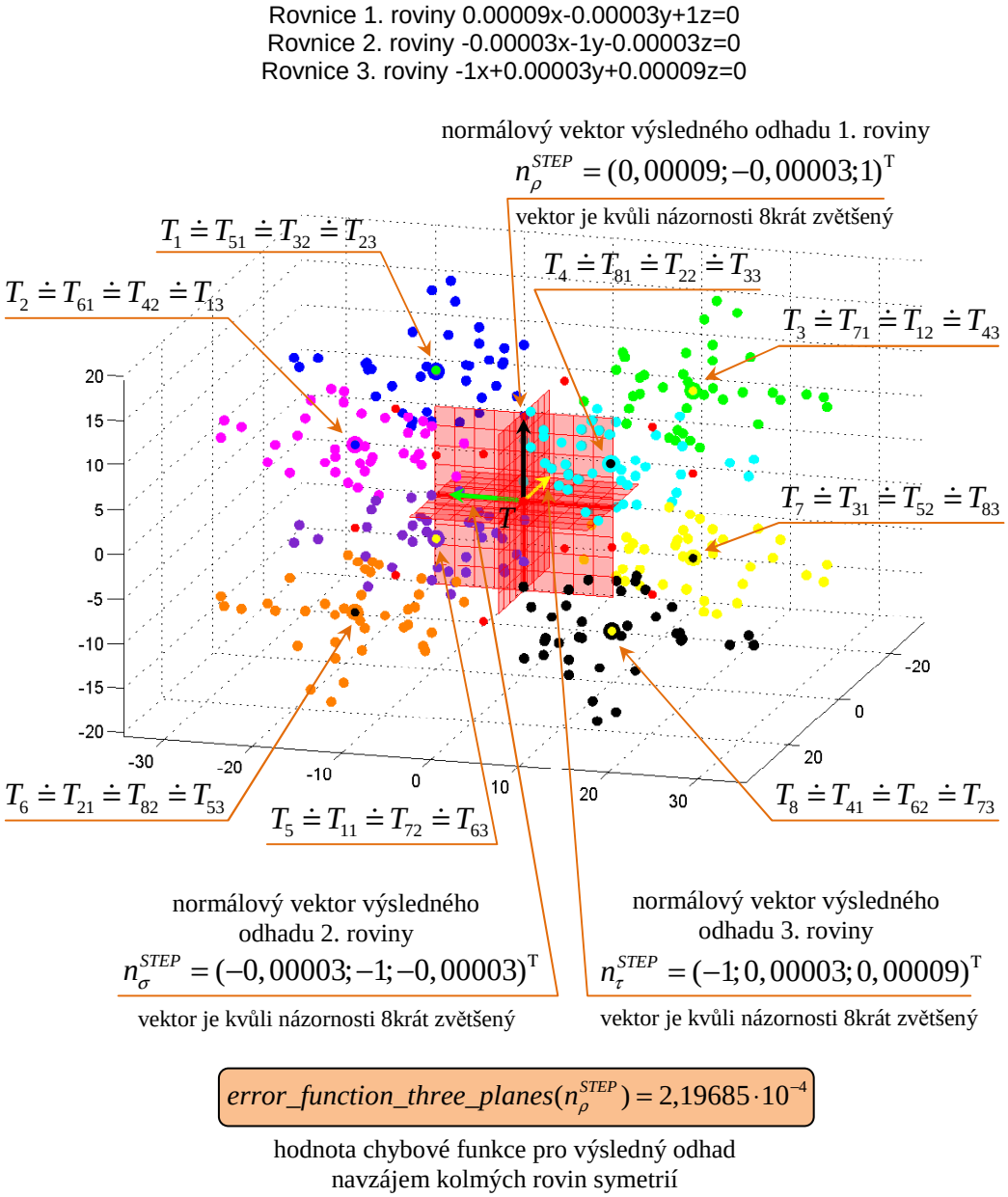




Obrázek 4.90: Postupná optimalizace tří navzájem kolmých rovin minimalizací chybové funkce modifikovanou metodou největšího spádu



Obrázek 4.91: Minimalizace chybové funkce modifikovanou metodou největšího spádu – závislost hodnoty chybové funkce na počtu iterací (logaritmické měřítko na ose *y*)



Obrázek 4.92: Výsledný odhad navzájem kolmých rovin symetrií

Zpracování dalších bodových množin z hlediska nalezení tří rovinových symetrií se třemi navzájem kolmými rovinami uvedeme pro reálné množiny v závěrečné kapitole. Přičemž jednat se bude již o rozsáhlé bodové množiny získané skenováním skutečných povrchů.

Navržené metody hledání rovinových symetrií s jednou rovinou nebo s dvěma a třemi navzájem kolmými rovinami lze obohatit o další možné přístupy. Vylepšení této metody pro jemné vyladění poloh rovin symetrie nebo symetrií je možné dosáhnout pomocí zjišťování korespondence jednotlivých bodů podle roviny nebo jednotlivých rovin symetrií. Současná

navržená metoda dává uspokojivé výsledky, tato modifikace proto představuje námět na budoucí výzkumnou práci.

Zajímavostí je, že toto vylepšení odpovídá informatickému problému hledání párování s minimální váhou v bipartitním grafu (*minimum weighted matching in bipartite graphs*), (Cheng *a kol.*, 1996). Je dokázáno, že nejlepší algoritmus na nalezení takového párování pracuje s kubickou asymptotickou složitostí.

Kapitola 5

Povrchové analýzy

Dalším cílem digitální rekonstrukce povrchu je vymezení hranice zkoumaného prostorového objektu. V následující kapitole se proto věnujeme povrchovým analýzám bodových mračen. Nejdříve se zaměříme na odhady tečných rovin a tedy i normál bodových množin popisujících zkoumaný objekt. Dále se budeme zabývat hraničními reprezentacemi objektu. Představíme možnosti tvorby polygonální sítě reprezentující povrch objektu a možný přechod k analytické deskripci objektu.

K určení odhadu tečné roviny a zároveň normály (oddíl *Odhad tečných rovin a normál bodové množiny*) objektu v daném bodě mračna využíváme odvozené metody ortogonálního prokládání dat rovinou případně analýzy hlavních komponent popsané v kapitole 3. V tomto oddíle představujeme známý algoritmus, který dále modifikujeme.

Polygonální síť reprezentující povrch objektu sestrojujeme celkem třemi možnými způsoby. Jednak volíme inkrementální tedy postupnou konstrukci sítě (oddíl *Inkrementální konstrukce polygonální sítě*), která je vhodná pro typy bodových množin, jež zpracováváme, a navíc je geometricky názorná. A jednak se při konstrukci polygonální sítě můžeme rozhodnout pro využití transformace souřadnic bodů do cylindrických souřadnic (oddíl *Triangulace povrchu pomocí transformace*), jelikož často zpracováváme bodové množiny odpovídající rotačním plochám. Při tomto postupu využíváme navržených metod hledání osy rotační plochy z předchozí kapitoly. Tyto dva existující přístupy doplňujeme o řadu vylepšení. Dále navrhujeme třetí možnost tvorby polygonální sítě vycházející z určení rovinových symetrií (oddíl *Konstrukce povrchové triangulace pomocí rovinové symetrie*). Nalezením orientace bodového mračna pomocí rovin symetrií lze v dalším postupu zpracovávat pouze část bodové množiny a využívat opět triangulační algoritmy v rovině. Případně lze rovinových symetrií využít také v dalším postupu u analytické deskripce povrchu.

Všechny navržené algoritmy implementujeme ve výpočetním prostředí MATLAB. Grafické výstupy doprovázející veškeré geometrické algoritmy získáváme rovněž z MATLABu nebo je vytváříme v modelovacím softwaru Rhinoceros. Algoritmy ověřujeme na bodových množinách, které generujeme buď z MATLABu nebo ze softwaru Rhinoceros stejně jako v předchozí kapitole. Předkládáme celou řadu příkladů s názornými ilustracemi demonstrujících odvozené postupy, přičemž v této kapitole spouštíme programy výhradně na počítačově generovaných bodových množinách. Opět je to záměr, neboť u počítačově generované bodové množiny lze názorněji demonstrovat fungování navržených metod. Testovací bodové množiny vytváříme tradičně v programu Rhinoceros (regulární množiny) nebo ve výpočetním prostředí MATLAB (regulární množiny, náhodně generované body na povrchu ploch nebo zašuměné body). Při generování zašuměných bodů v prostoru použí-

váme normální rozdělení s nulovou střední hodnotou se záměrem napodobit reálné situace nepřesného měření.

Naimplementované programy a konkrétní počítačově generované vstupní bodové množiny jsou k dispozici na přiloženém výměnném médiu.

Postupy uvedené v této kapitole plánujeme rovněž dále rozvíjet a vylepšovat v budoucí práci. Především analytický popis povrchu představuje námět pro pozdější výzkum.

5.1 Odhad tečných rovin a normál bodové množiny

V tomto oddíle využijeme již odvozené metody ortogonálního prokládání dat rovinou a analýzy hlavních komponent k určení odhadu tečné roviny a tedy i normály bodového mračna v daném bodě. Algoritmus, který zde předkládáme, je inspirován článkem (Hoppe *a kol.*, 1992). Dále jej upravujeme pro speciální případy bodových množin, jejichž zpracováním se zabýváme.

Nechť je dána neprázdná množina $\mathcal{X}$ *n* bodů v eukleidovském prostoru E_3 popisující nějaký prostorový objekt, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^n$. Při snímání bodů $\{X_i\}_{i=1}^n$ skenováním reálných povrchů dochází obvykle k nepřesnostem, proto i zde připouštíme, že body vstupní množiny $\mathcal{X}$ leží na nebo velmi blízko povrchu, který budeme značit $\mathcal{P}$. Předpokládáme tedy, že každý bod $X_i \in \mathcal{X}$ lze vyjádřit jako $X_i = P_i + e_i$, kde $P_i \in \mathcal{P}$ a $e_i \in \mathbb{R}$ je vektor chyby, pro $i = 1, 2, ..., n$. Pro všechna $i = 1, 2, ..., n$ platí $\|e_i\| \leq \delta$, přičemž hodnota δ je dána přesností daného typu skenovacího zařízení.

Tečnou rovinu ρ_i bodového mračna v daném bodě $X_i \in \mathcal{X}$ budeme odhadovat pomocí jednotkového normálového vektoru n_i, kde $i = 1, 2, ..., n$. Hledanou tečnou rovinu ρ_i v bodě X_i vyjádříme obecnou rovnicí

$$n_i \cdot (X - X_i) = 0, \tag{5.1}$$

kde X_i je bod tečné roviny a n_i je její jednotkový normálový vektor, tj. $\|n_i\| = 1$. Souřadnice bodů a vektorů budeme značit pomocí horního indexu, tj. $X_i = [X_i^x, X_i^y, X_i^z]$ a $n_i = (n_i^x, n_i^y, n_i^z)^{\mathrm{T}}$.

K nalezení jednotkového normálového vektoru n_i použijeme metodu ortogonálního prokládání dat rovinou případně analýzu hlavních komponent PCA, které jsou popsané v oddílech 3.5 a 3.6. Pro uvažovaný bod $X_i \in \mathcal{X}$ určíme jeho *k* nejbližších sousedů, toto okolí značme $k_N(X_i)$. Předpokládejme, že pro každé $i = 1, 2, ..., n$ máme body množiny $\mathcal{X}$ uspořádány podle vzdáleností od bodu X_i tj. máme indexy $i_1, i_2, ..., i_n \in \{1, 2, ..., n\}$ vzájemně různé, že

$$\|X_i - X_{i_1}\| \leq \|X_i - X_{i_2}\| \leq ... \leq \|X_i - X_{i_n}\|, \tag{5.2}$$

potom množinu $k_N(X_i)$ definujeme následovně

(5.3) $$k_N(X_i) = \left\{ X_{i_1}, X_{i_2}, ..., X_{i_k} \right\} = \left\{ X_{i_j} \right\}_{j=1}^{k},$$

Poznamenejme, že $i_1 = i$, vzdálenost je nulová, tedy $\left\| X_i - X_{i_1} \right\| = 0$.

Množinu bodů $k_N(X_i)$ nyní proložíme ortogonálně rovinou, viz oddíl 3.5. Tuto pomocnou rovinu aproximující okolí $k_N(X_i)$ bodu X_i budeme značit $\bar{\rho}_i$ a vyjadřovat tradičně obecnou rovnicí, tj.

(5.4) $$n_i \cdot (X - A_i) = 0,$$

kde A_i je bod roviny a n_i je její jednotkový normálový vektor, tj. $\left\| n_i \right\| = 1$. Normálový vektor n_i roviny $\bar{\rho}_i$ je tedy zároveň normálovým vektorem roviny ρ_i. Podle odvozených vztahů v oddíle 3.5 víme, že bod A_i spočítáme jako aritmetický průměr bodů množiny $k_N(X_i)$. Rozepišme si výsledek po souřadnicích

(5.5) $$A_i^x = \frac{1}{k} \sum_{j=1}^{k} X_{i_j}^x,\ A_i^y = \frac{1}{k} \sum_{j=1}^{k} X_{i_j}^y,\ A_i^z = \frac{1}{k} \sum_{j=1}^{k} X_{i_j}^z\ .$$

Nyní máme určený bod A_i, kterým prochází pomocná aproximující rovina $\bar{\rho}_i$. Zbývá dopočítat jednotkový normálový vektor n_i této roviny. Podle oddílu 3.5 jej určíme jako vlastní vektor příslušný nejmenšímu vlastnímu číslu matice M z 3.103. Matici M přepišme pro případ množiny bodů $k_N(X_i)$ a přeznačme jako M_i, tedy

(5.6) $$M_i = \sum_{j=1}^{k} \begin{pmatrix} (w_{i_j}^x)^2 & w_{i_j}^x w_{i_j}^y & w_{i_j}^x w_{i_j}^z \\ w_{i_j}^x w_{i_j}^y & (w_{i_j}^y)^2 & w_{i_j}^y w_{i_j}^z \\ w_{i_j}^x w_{i_j}^z & w_{i_j}^y w_{i_j}^z & (w_{i_j}^z)^2 \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{k} (w_{i_j}^x)^2 & \sum_{j=1}^{k} w_{i_j}^x w_{i_j}^y & \sum_{j=1}^{k} w_{i_j}^x w_{i_j}^z \\ \sum_{j=1}^{k} w_{i_j}^x w_{i_j}^y & \sum_{j=1}^{k} (w_{i_j}^y)^2 & \sum_{j=1}^{k} w_{i_j}^y w_{i_j}^z \\ \sum_{j=1}^{k} w_{i_j}^x w_{i_j}^z & \sum_{j=1}^{k} w_{i_j}^y w_{i_j}^z & \sum_{j=1}^{k} (w_{i_j}^z)^2 \end{pmatrix},$$

kde $w_{i_j}^x, w_{i_j}^y, w_{i_j}^z$ jsou souřadnice vektoru $w_{i_j} = X_{i_j} - A_i$. Podle oddílu 3.5 víme, že takto určený jednotkový normálový vektor n_i minimalizuje kvadratickou formu $\sigma_i^2 = n_i^{\mathrm{T}} M_i n_i$. Minimum této kvadratické formy je rovno právě nejmenšímu vlastnímu číslu matice M_i, což plyne z věty 3.3. Tím máme určenou pomocnou aproximující rovinu $\bar{\rho}_i$. Tuto rovinu nebudeme vykreslovat, použijeme pouze odvozený jednotkový normálový vektor n_i a dosadíme jej do rovnice roviny ρ_i uvedené v (5.1). Takto v daném bodě X_i bodové množiny $\mathcal{X}$ odhadneme jednak tečnou rovinu, jednak směr normály této bodové množiny. Pokud tento postup provedeme pro každý bod množiny $\mathcal{X}$, dostáváme odhady tečných rovin a normál bodové množiny ve všech bodech a prvotní hrubou reprezentaci povrchu.

Stejné odhady tečných rovin a normál bodové množiny v jejích bodech obdržíme, pokud místo ortogonálního prokládání bodů $k_N(X_i)$ rovinou zvolíme metodu analýzy hlavních komponent, kterou jsme odvodili v oddíle 3.6. Sestavíme-li odhad kovarianční matice S_i^0 pro data množiny $k_N(X_i)$ podle 3.111, jednotkový normálový vektor n_i spočteme jako vlastní vektor příslušný nejmenšímu vlastnímu číslu matice S_i^0.

Povšimněme si, že jednotkové normálové vektory v bodech množiny $\mathcal{X}$ získané aproximací nebo analýzou hlavních komponent nejsou orientované. K určení orientace lze použít algoritmy propagace normál, jak je uvedeno v článku (Hoppe *a kol.*, 1992). My alternativně navrhujeme podobnou metodu tzv. hlasovací algoritmus, jehož ověření je plánováno do budoucí práce. Princip hlasovacího algoritmu je následující. Pro každý bod X_i bodové ny $\mathcal{X}$, ve kterém zkoumáme orientaci jednotkového normálového vektoru n_i, sledujeme, jak vypadají orientace jednotkových normálových vektorů v nějakém jeho okolí (lze uvažovat okolí, pro které jsme počítali pomocnou aproximační rovinu). Zajímá nás většinová orientace, tj. pokud je jednotkový normálový vektor n_i v daném bodě X_i opačně orientovaný než většina jednotkových normálových vektorů v jeho okolí, orientaci tohoto vektoru změníme a uvažujeme opačný jednotkový normálový vektor, tj. $-n_i$.

Tento postup hledání správné orientace normálových vektorů lze použít pro libovolnou bodovou množinu popisující reálný povrch. S ohledem na typy bodových množin, které zpracováváme v této disertační práci, předkládáme ještě další možnost řešení správné orientace normálových vektorů a to pro případ bodových množin popisujících obecnou rotační plochu. Postupovat budeme následovně. Podle uvedeného algoritmu určíme odhady tečných rovin a normálových vektorů a dále nalezneme osu bodové množiny. K nalezení osy užijeme buď metod zavedených v oddílech 3.3 (*Ortogonální prokládání dat přímkou*) a 3.6 (*Analýza hlavních komponent*) nebo v oddílech 4.2 (*Hledání osy rotační válcové plochy*) a 4.3 (*Hledání osy obecné rotační plochy*), pokud množina bodů reprezentuje pouze část rotační plochy. Bod X_i, v němž nyní máme určený jednotkový normálový vektor, promítneme ortogonálně na nalezenou osu do bodu X_i'. Dále určíme velikost úhlu $\varphi \in \langle 0, \pi \rangle$, který svírají vektory $X_i' - X_i$ a jednotkový normálový vektor n_i v bodě X_i. Je-li tento úhel φ menší než pravý, potom uvažujeme opačný jednotkový normálový vektor, tj. $-n_i$, je-li úhel φ větší než pravý orientaci normálového vektoru ponecháme. Při správném odhadu tečné roviny nemůže být φ pravý úhel, pokud tato situace nastane, je nutné změnit hodnotu *k*. Pro určení, z jakého intervalu je úhel φ, stačí uvažovat hodnotu skalárního součinu vektorů $X_i' - X_i$ a n_i, neboť

$$\begin{aligned} (X_i' - X_i) \cdot n_i > 0 &\Rightarrow \varphi \in \left\langle 0, \tfrac{\pi}{2} \right) \\ (X_i' - X_i) \cdot n_i < 0 &\Rightarrow \varphi \in \left(\tfrac{\pi}{2}, \pi \right\rangle \\ (X_i' - X_i) \cdot n_i = 0 &\Rightarrow \varphi = \tfrac{\pi}{2}. \end{aligned} \tag{5.7}$$

Musíme dále ošetřit speciální případ, kdy bod X_i leží přímo na ose symetrie bodové množiny. Bod X_i potom splývá s ortogonálním průmětem X_i' a jednotkový normálový vektor n_i je lineárně závislý se směrovým vektorem nalezené osy. Pokud tato situace nastane,

určíme vektor $X_i - T$, kde T je těžiště vstupní bodové množiny, a podle předchozího odvození leží na její ose. Potom stačí rozhodnout, zda jsou vektory $X_i - T$ a n_i shodně orientovány či nikoliv. V případě shodné orientace je jednotkový normálový vektor n_i ponechán, jinak uvažujeme opačný jednotkový normálový vektor, tj. $-n_i$.

V odhadování tečných rovin a normál bodového mračna se objevuje konstanta *k*, která určuje počet bodů v okolí každého bodu vstupní množiny, viz (5.3). Nejvhodnější volbu určíme na základě experimentů. Demonstrujme tedy postup hledání odhadů tečných rovin a normál bodového mračna na konkrétních příkladech. Ukažme, zda je zvolená metoda ortogonálního prokládání dat rovinou a analýzy hlavních komponent tím vhodným postupem pro určování odhadů tečných rovin a normálových vektorů. To provedeme porovnáním odchylek odhadů tečných rovin a normál bodové množiny od tečných rovin a normál bodové množiny určených přesně. Porovnáním těchto odchylek pro různé vstupní hodnoty *k* rovněž vybereme nejvhodnější volbu této konstanty. Pro tyto účely budeme vycházet z faktu, že známe analytický popis povrchu a umíme tedy v každém bodě spočítat tečnou rovinu i normálu plochy, která jej reprezentuje. Abychom toto porovnání mohli provést, uvažujme synteticky generovanou množinu, která vznikla navzorkováním povrchu, jehož analytický popis známe. V následujícím příkladě volíme speciálně povrch rotačního jednodílného hyperboloidu. Uvažovat budeme dva typy bodových množin a to pravidelnou a nepravidelnou.

Příklad 5.1: *ODHADY TEČNÝCH ROVIN A NORMÁLOVÝCH VEKTORŮ BODOVÉ MNOŽINY*. Mějme dánu množinu $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které jsou navzorkované na části povrchu rotačního jednodílného hyperboloidu. Implicitní vyjádření rotačního jednodílného hyperboloidu je

$$\frac{x^2}{10^2} + \frac{y^2}{10^2} - \frac{z^2}{11^2} = 1.$$

V prvním případě tvoří body na povrchu plochy regulární pravidelnou množinu. Obrázek 5.1 znázorňuje postupný výpočet odhadů tečných rovin a jednotkových normálových vektorů této bodové množiny v jejích jednotlivých bodech, výpočet probíhá iteračně, přičemž zobrazujeme jen několik kroků. V další fázi algoritmu určujeme správné orientace normálových vektorů, k čemuž využíváme navrženého postupu, tj. spočítáme osu bodové množiny pomocí ortogonálního prokládání dat přímkou a body množiny na ni ortogonálně promítáme. Na základě hodnot skalárních součinů příslušných vektorů podle (5.7) rozhodneme o správné orientaci odhadů jednotkových normálových vektorů v bodech množiny. Výsledek tohoto procesu znázorňuje obrázek 5.2. Dále vycházíme ze znalosti analytického popisu povrchu a v každém bodě množiny spočítáme jednotkový normálový vektor. Na obrázku 5.3 je možné v detailu pozorovat v každém bodě množiny vykreslené jednotkové normálové vektory – odhad bez orientace (černě), odhad se správnou orientací (červeně) a přesný jednotkový normálový vektor (modře). Vektory jsou kvůli názornosti 3krát zvětšené.

Na základě statistického pokusu vybereme nejlepší hodnotu konstanty *k*, která určuje počet bodů v okolí $k_N(X_i)$ každého bodu X_i, viz (5.3). Měříme odchylky přesných a odhadnutých jednotkových normálových vektorů bodové množiny v jed-

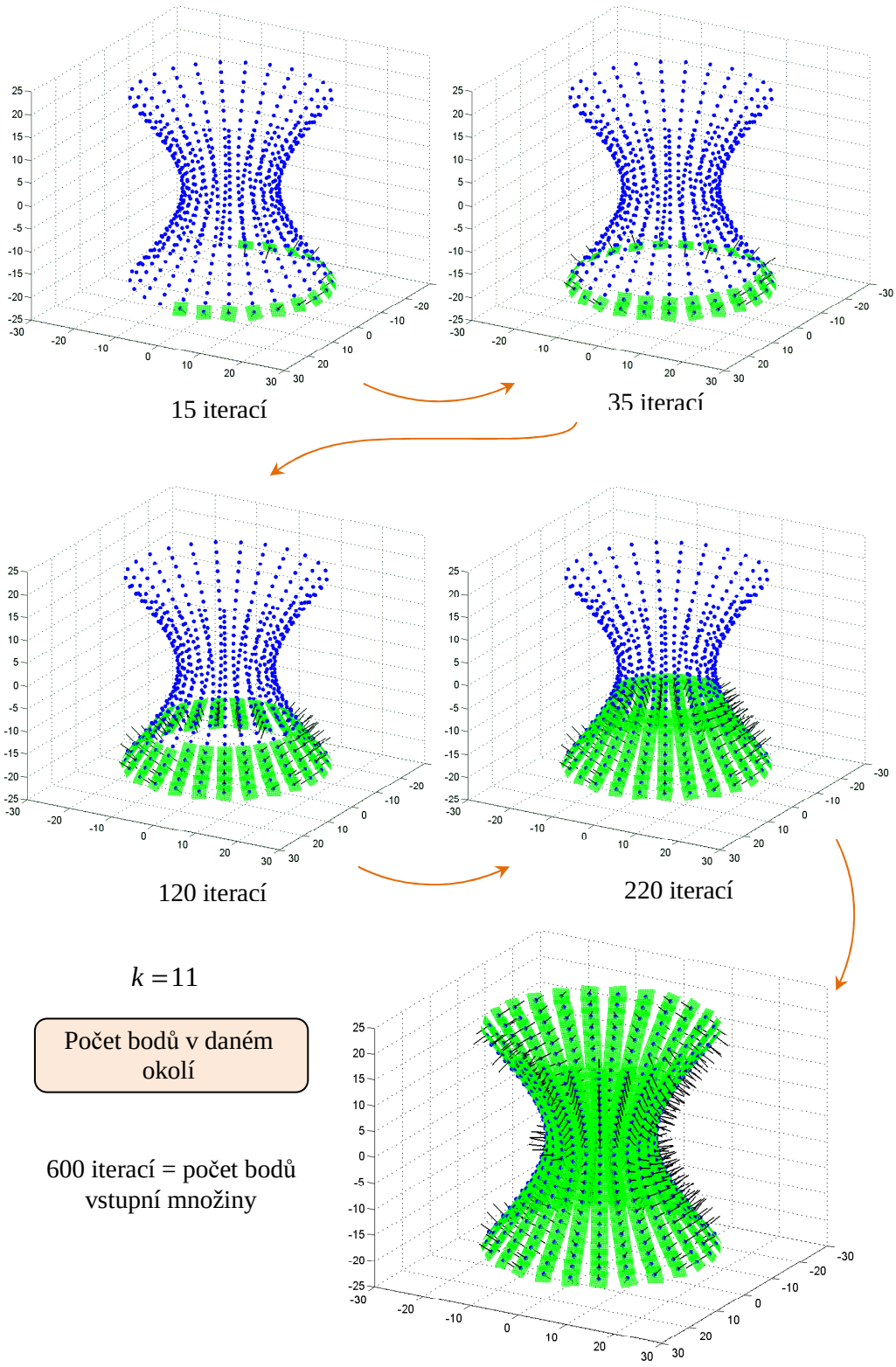
notlivých bodech a zkoumáme statistické vlastnosti tohoto souboru dat. Konkrétně jsou to vlastnosti vyjádřené krabicovým diagramem, což je obrázkové schéma poskytující informaci o mediánu, 25. a 75. percentilu, minimu a maximu a odlehlých bodech v souboru naměřených hodnot. Porovnáním těchto vlastností pro různé hodnoty konstanty *k*, rozhodujeme, který odhad tečných rovin a jednotkových normálových vektorů je nejlepší. V tomto případě analyzujeme výsledky po řadě pro hodnoty $k = 6,7,...,24$ a zobrazujeme příslušné krabicové diagramy, sledujme obrázek 5.4. Pro hodnoty *k* v tomto intervalu, dostáváme dobré výsledky, neboť hodnoty odchylek jsou rozloženy nízko a jejich rozptyly jsou malé. V našem případě vybíráme $k = 11$.

Ve druhém případě se jedná o body, které jsou na povrchu rotačního jednodílného hyperboloidu vygenerovány náhodně. Proces hledání odhadů tečných rovin a normálových vektorů je stejný, výstupy uvádíme analogicky jako v předchozím případě. Obrázek 5.5 znázorňuje postupný výpočet odhadů tečných rovin a jednotkových normálových vektorů této bodové množiny v jejích jednotlivých bodech. Tentokrát jsme k nalezení odhadu osy symetrie bodové množiny užili iterační metodu navrženou v oddíle 4.3 (*Hledání osy obecné rotační plochy*). Minimalizaci chybové funkce jsme provedli pomocí diferenciální numerické metody největšího spádu a to modifikovanou verzí s konstantní délkou kroku. Použitá chybová funkce je definována v (4.10). Obrázek 5.6 ilustruje postupné vylepšování polohy přímky, přičemž přímka je vykreslována jen v několika pozicích. Zvlášť je vykreslen výsledný odhad osy a uvedeny jsou parametrické rovnice, viz obrázek 5.7. Uvádíme rovněž volbu vstupních parametrů iterační modifikované metody největšího spádu při výpočtu odhadu osy symetrie bodové množiny. Na obrázku 5.8 můžeme vidět již správně orientované normálové vektory bodové množiny. Obrázek 5.9 zachycuje detail množiny s jednotlivými normálovými vektory – odhad bez orientace (černě), odhad se správnou orientací (červeně) a přesný jednotkový normálový vektor (modře). Vektory jsou kvůli názornosti 3krát zvětšené.

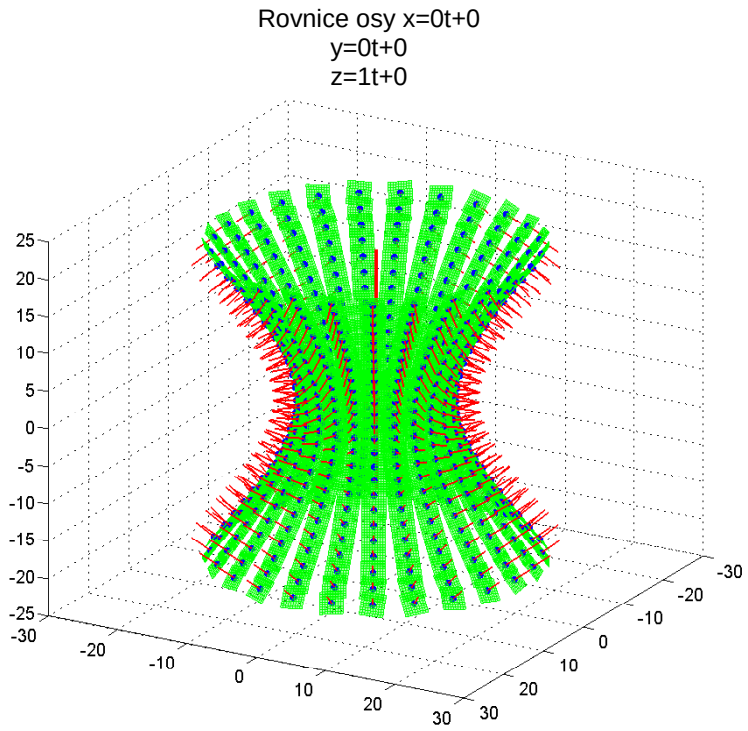
Statistickými výpočty jsme rozhodli, že nejlepší volba konstanty *k* je 17. Vycházeli jsme přitom z porovnání statistických vlastností souboru odchylek přesných a odhadnutých jednotkových normálových vektorů bodové množiny v jejích jednotlivých bodech. Analyzujeme soubory dat po řadě pro hodnoty $k = 3,4,...,24$ a zobrazujeme příslušné krabicové diagramy, viz obrázek 5.10. U náhodných dat se dá očekávat, že pokles odchylek přidáváním bodů do okolí nebude z počátku tak nápadný, jako tomu bylo u pravidelné množiny. Stejně tak růst odchylek přidáváním bodů do okolí po překročení nejvhodnější hodnoty *k* nebude dramatický. Nyní má větší význam sledovat rozptyl souboru dat a množství odlehlých bodů. Na základě zobrazených diagramů je $k = 17$ zřejmě nejvhodnější volbou. Kvůli přehlednosti a malému měřítku nezobrazujeme všechny odlehlé body.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

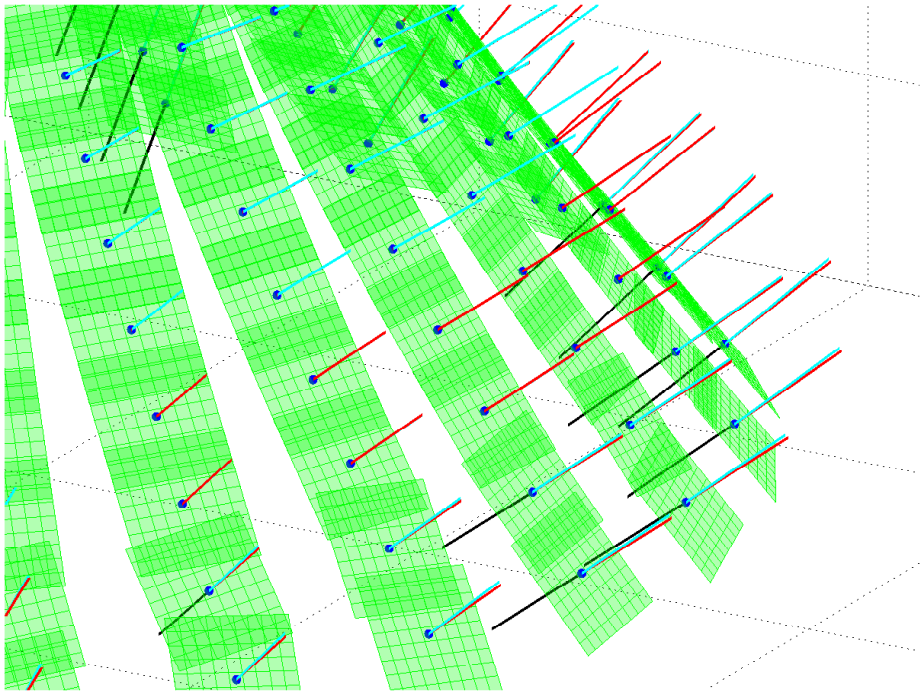
`odhad_tecnych_rovin.m` (programy/tecne_roviny).



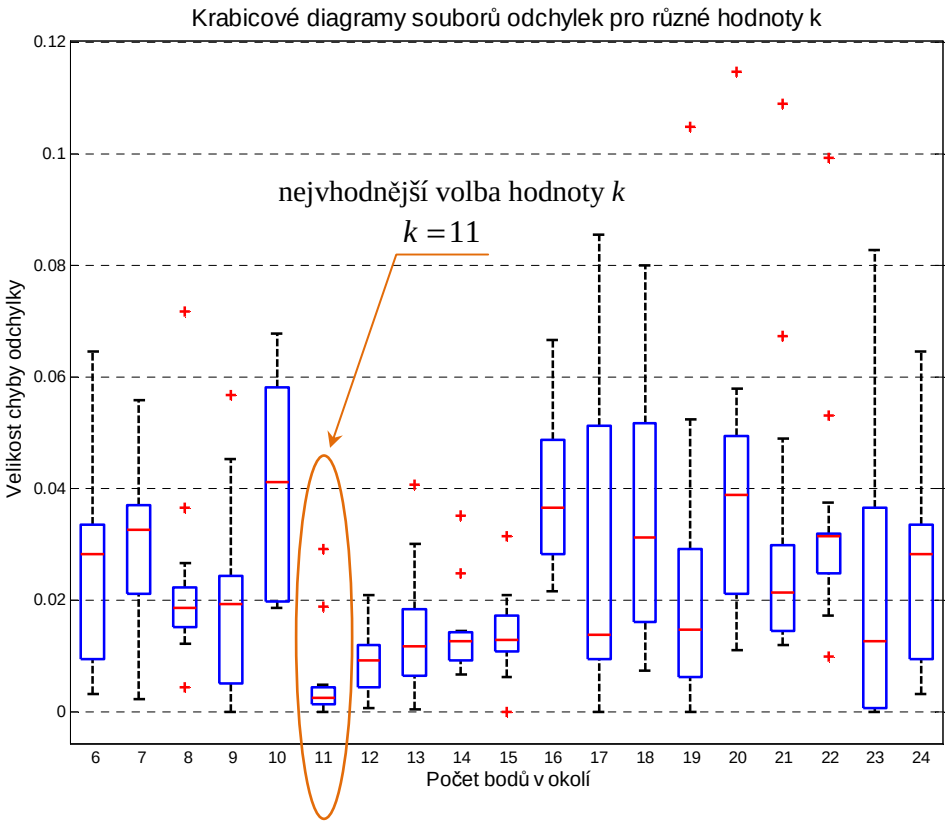
Obrázek 5.1: Postupný výpočet odhadů tečných rovin a jednotkových normálových vektorů (bez zjišťování správné orientace normálových vektorů)



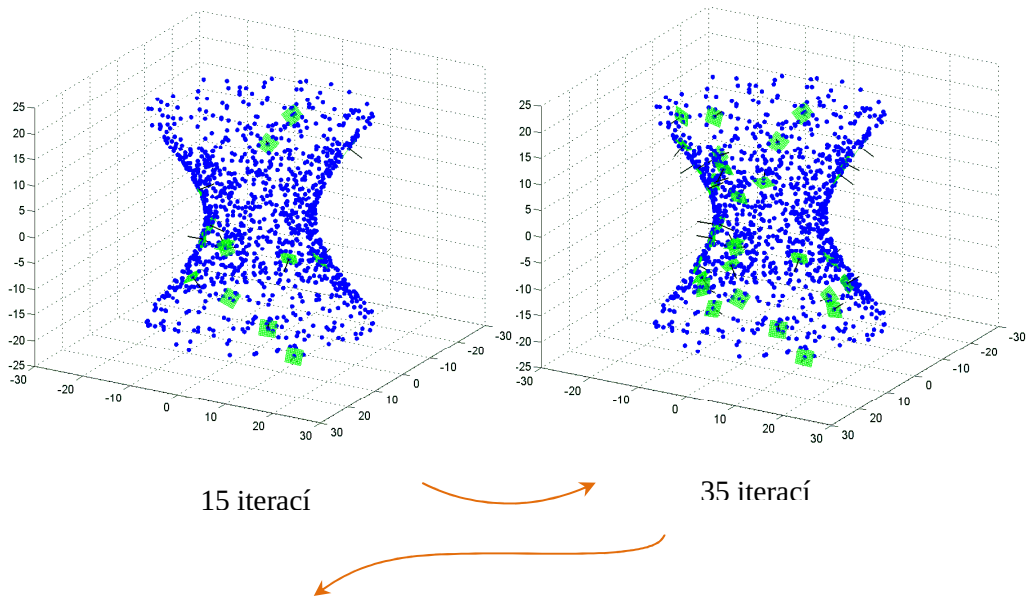
Obrázek 5.2: Odhady tečných rovin a normálových vektorů bodové množiny v jednotlivých bodech – odhady normálových vektorů se správnou orientací

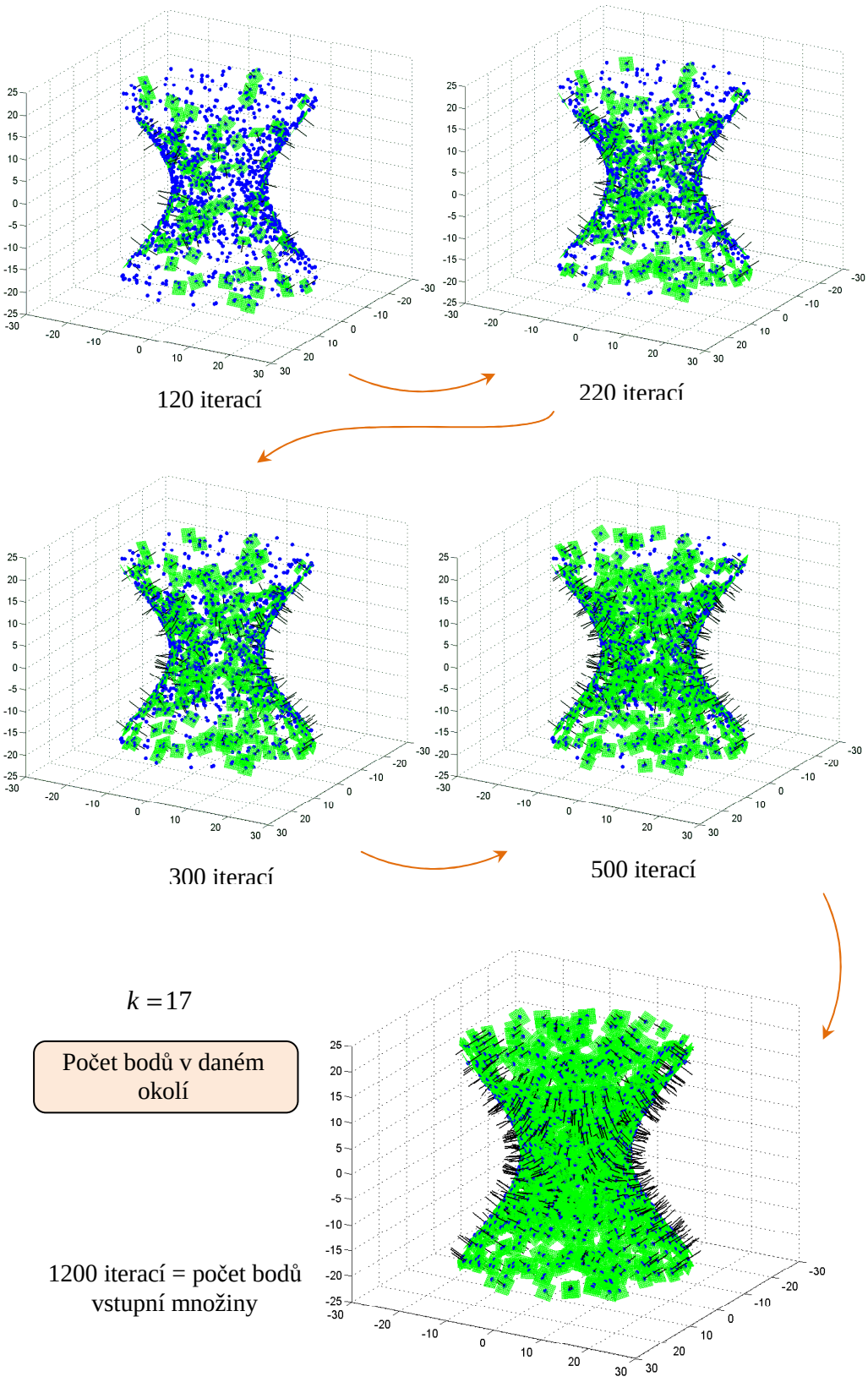


Obrázek 5.3: V každém bodě množiny vykreslené normálové vektory – odhad bez orientace (černě), odhad se správnou orientací (červeně) a přesný jednotkový normálový vektor (modře) – detail množiny

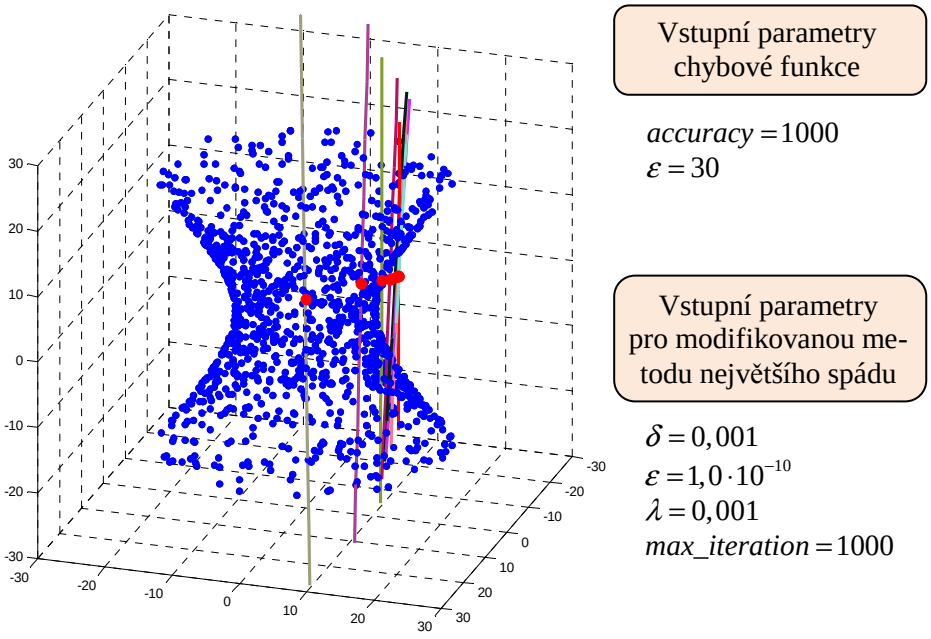


Obrázek 5.4: Krabicové diagramy souborů odchylek odhadů jednotkových normálových vektorů od přesných normálových vektorů pro různé hodnoty *k*

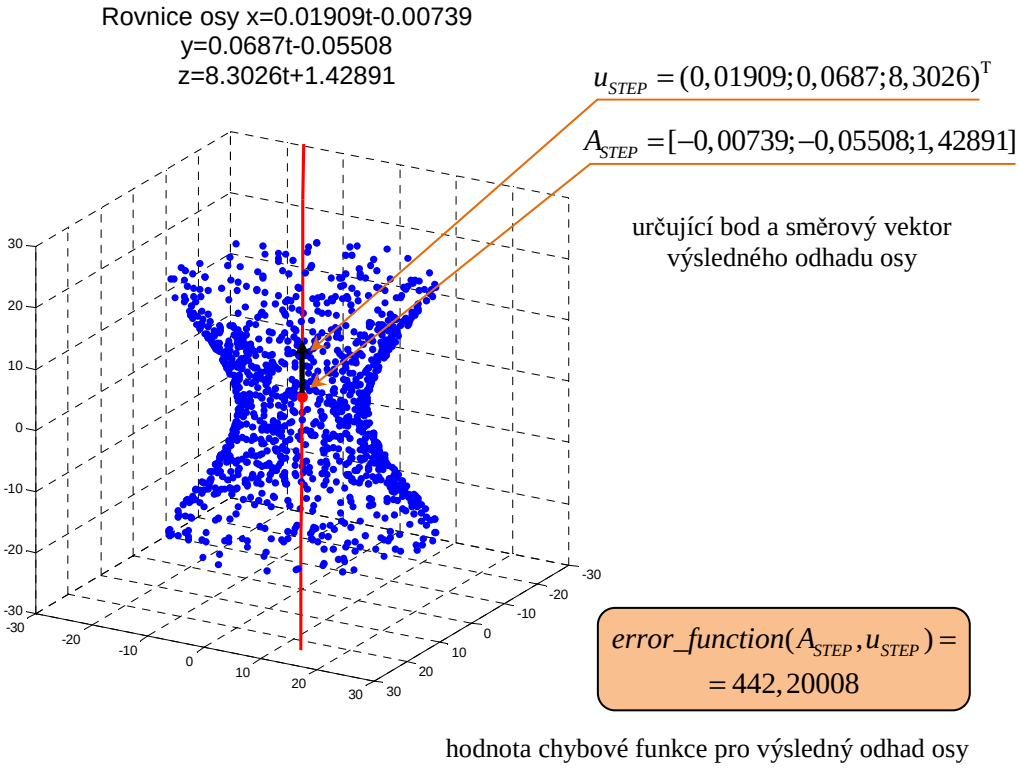




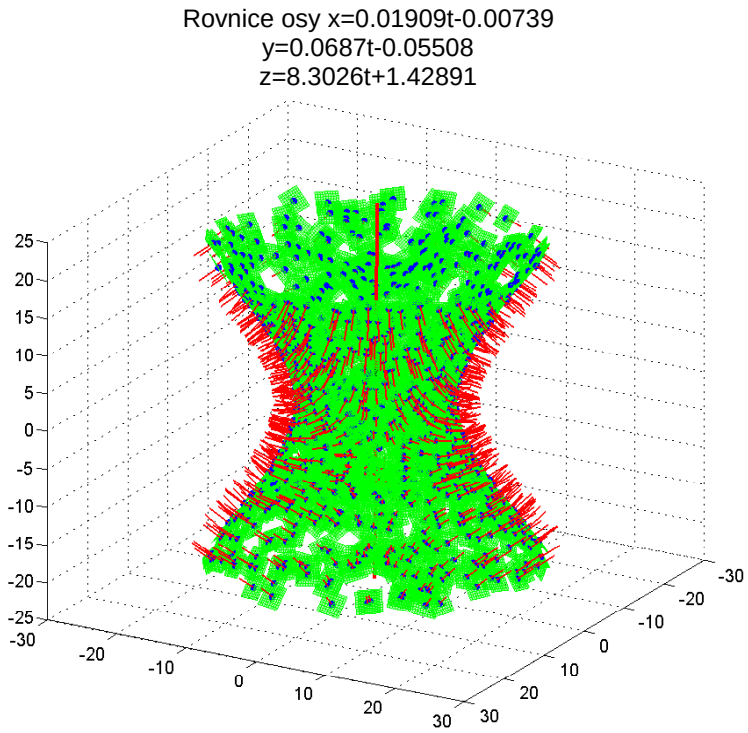
Obrázek 5.5: Postupný výpočet odhadů tečných rovin a jednotkových normálových vektorů (bez zjišťování správné orientace normálových vektorů)



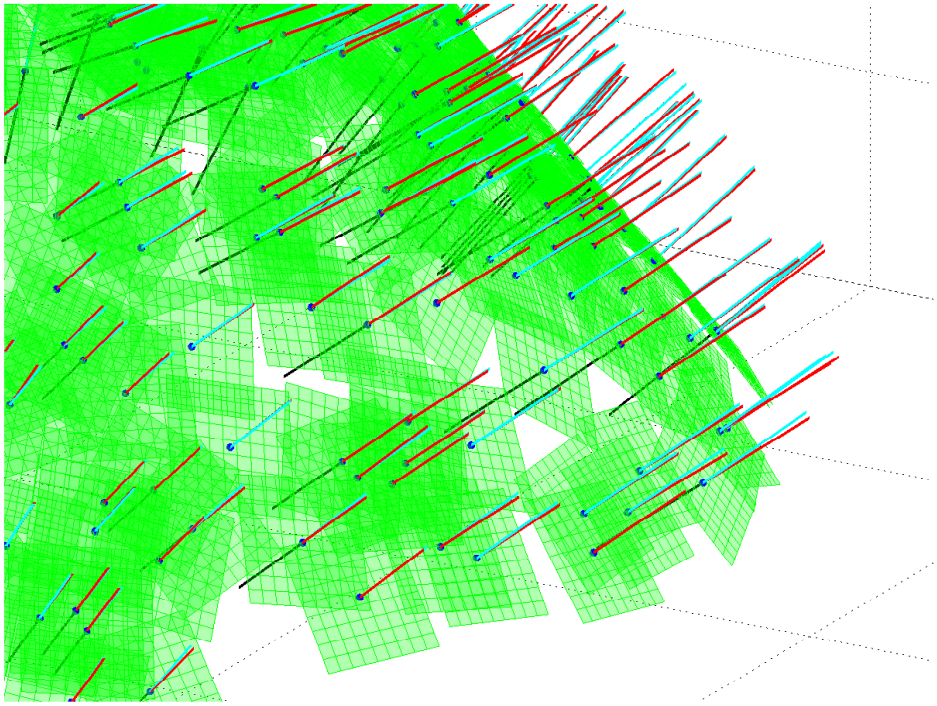
Obrázek 5.6: Vstupní parametry minimalizační metody a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



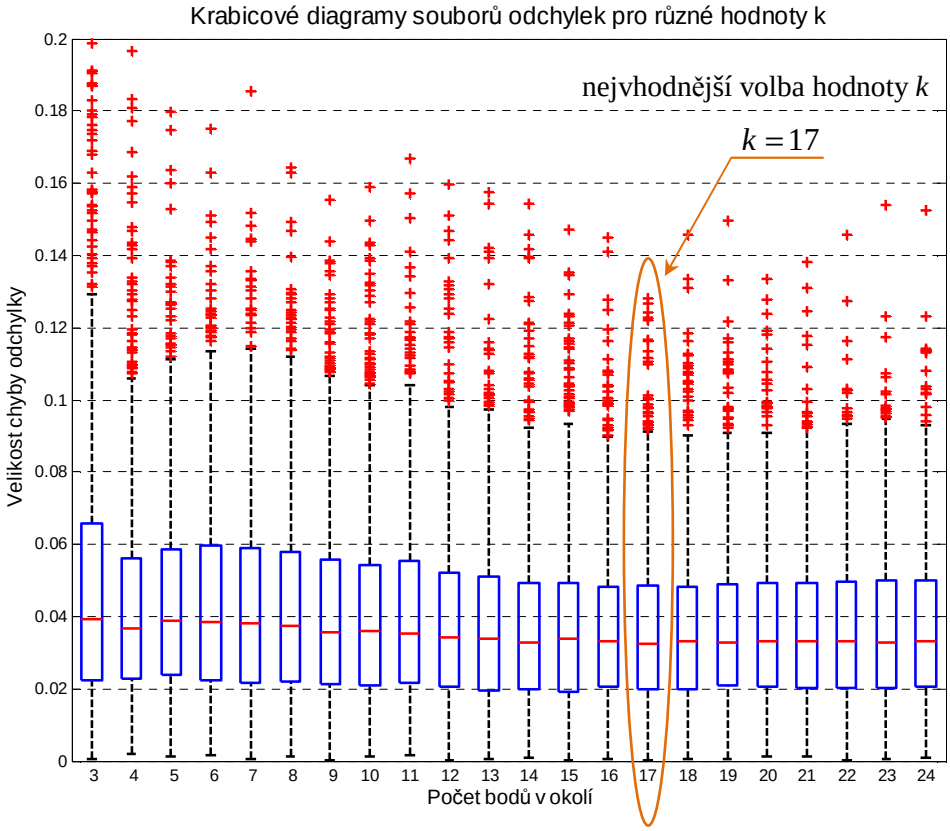
Obrázek 5.7: Výsledný odhad osy – modifikovaná metoda největšího spádu



Obrázek 5.8: Odhady tečných rovin a normálových vektorů bodové množiny v jednotlivých bodech – odhady normálových vektorů se správnou orientací



Obrázek 5.9: V každém bodě množiny vykreslené normálové vektory – odhad bez orientace (černě), odhad se správnou orientací (červeně) a přesný jednotkový normálový vektor (modře) – detail množiny



Obrázek 5.10: Krabicové diagramy souborů odchylek odhadů jednotkových normálových vektorů od přesných normálových vektorů pro různé hodnoty *k*

V případě zpracování reálných množin nemáme k dispozici analytický popis zkoumaného objektu a nelze tedy v bodech množiny počítat přesné normálové vektory plochy, která jej reprezentuje. Pro reálná data proto navrhujeme takový postup, že nejprve na podobných syntetických datech vyzkoušíme na základě statistického pokusu vhodnou volbu okolí a tedy i volbu konstanty *k*. Toto lze provést, neboť většinou známe alespoň základní vlastnosti zkoumaného objektu.

Do jisté míry lze využít představenou metodu odhadu orientovaných jednotkových normálových vektorů v bodech mračna také pro odstraňování nadbytečných bodů v naskenované množině. Ve většině případů totiž zpracováváme speciální bodové množiny, které odpovídají jednoduchým geometrickým plochám, lze předpokládat, že sousední jednotkové normálové vektory nemají příliš velké úhlové odchylky. To však nelze předpokládat o bodech, které vymezují hranici mezi naskenovaným objektem a částmi patřícími naskenovanému okolí (např. ruce, jedná-li se o model menších rozměrů). Odstraňujeme z bodového mračna tedy takové body, pro které je odchylka příslušných normálových vektorů příliš velká, a další body, které s nimi sousedí. Míru toho, kdy je bod podezřelý, že nepatří zkoumanému objektu, určujeme experimentálně.

Na druhou stranu pokud se jedná o složitější objekt s ostrými hranami, lze pomocí odhadu normálových vektorů tyto hrany rozpoznat právě porovnáváním odchylek těchto vektorů v sousedních bodech.

Vhodnou volbu konstanty *k*, která se v odhadování tečných rovin a normálových vektorů (normál) bodového mračna objevuje, v našich úlohách odvozujeme experimentálně na základě znalosti elementárních vlastností rekonstruovaného objektu. Existují však postupy pro její automatické určení. Dočteme se o tom více v článcích (Hoppe *a kol.*, 1992) nebo (Mitra a Nguyen,2003).

Rozvinuty jsou rovněž další postupy určování odhadů tečných rovin a normálových vektorů bodového mračna, které jsou založeny na jiných principech. Více o těchto metodách pojednávají články (Pauly *a kol.*, 2002; Dey *a kol.*, 2005; Boulch a Marlet, 2012). V posledním jmenovaném článku jsou předloženy nové metody pro určování normál speciálně povrchů s ostrými hranami.

5.2 Inkrementální konstrukce polygonální sítě

V tomto oddíle představíme existující algoritmus inkrementální tedy postupné konstrukce trojúhelníkové sítě reprezentující zkoumaný povrch a dále jej doplníme o vlastní vylepšení. Kladli jsme si za cíl vycházet při tvorbě nově navržených algoritmů ze základních geometrických vlastností rekonstruovaných objektů, případně při použití známých algoritmů zdůrazňovat geometrické principy, na kterých jsou založeny. Proto i zde z možných konstrukcí trojúhelníkové sítě vybíráme postup, který se opírá o geometrická pravidla. Inkrementální konstrukce ploškové reprezentace objektu je názorná a je vhodná pro typy množin, které zpracováváme.

Metoda postupné tvorby trojúhelníkové sítě z bodového mračna reprezentující povrch je založena na několika geometrických pravidlech. Tato pravidla jsou popsána v pracích (Yu *a kol.*, 2010; Pešková, 2011). Některá pravidla jsme nahradili vlastními pravidly a uvedené postupy jsme dále vylepšili vlastními technikami, které zde podrobně rozebereme.

Mějme dánu neprázdnou množinu $\mathcal{X}$ *n* bodů v eukleidovském prostoru E_3 popisující nějaký prostorový objekt, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Další předpoklady jsou stejné jako na začátku oddílu 5.1, i zde tedy připouštíme, že body vstupní množiny $\mathcal{X}$ leží na nebo velmi blízko povrchu, který budeme značit $\mathcal{P}$. Hledáme povrchovou triangulaci tak, že vrcholy trojúhelníků jsou všechny body množiny $\mathcal{X}$ a každá hrana je společná nejvýše dvěma trojúhelníkům.

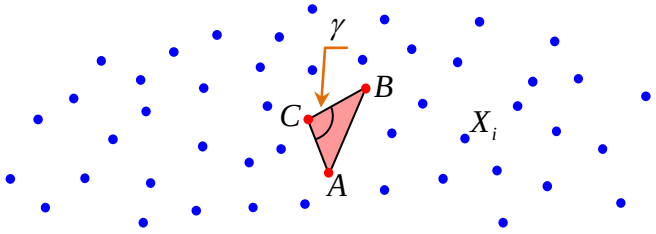
V prvním kroku algoritmu určujeme tzv. základní trojúhelník, od kterého začíná postupná tvorba polygonální sítě. Nechť se vrcholy tohoto trojúhelníka nazývají A, B, C. První vrchol A určíme jako nejbližší bod vstupní bodové množiny $\mathcal{X}$ k jejímu těžišti T, které spočítáme jako aritmetický průměr bodů $\{X_i\}_{i=1}^{n}$. Dále vrchol B je nejbližší bod množiny $\mathcal{X}$ k bodu A. Tím je stanovena první hrana AB v triangulaci. Třetí vrchol trojúhelníka, bod C,

vybereme z množiny $\mathcal{X}$ tak, aby vnitřní úhel v trojúhelníku při vrcholu C byl největší možný. K tomu použijeme kosinovou větu, která platí pro každý trojúhelník, tj.

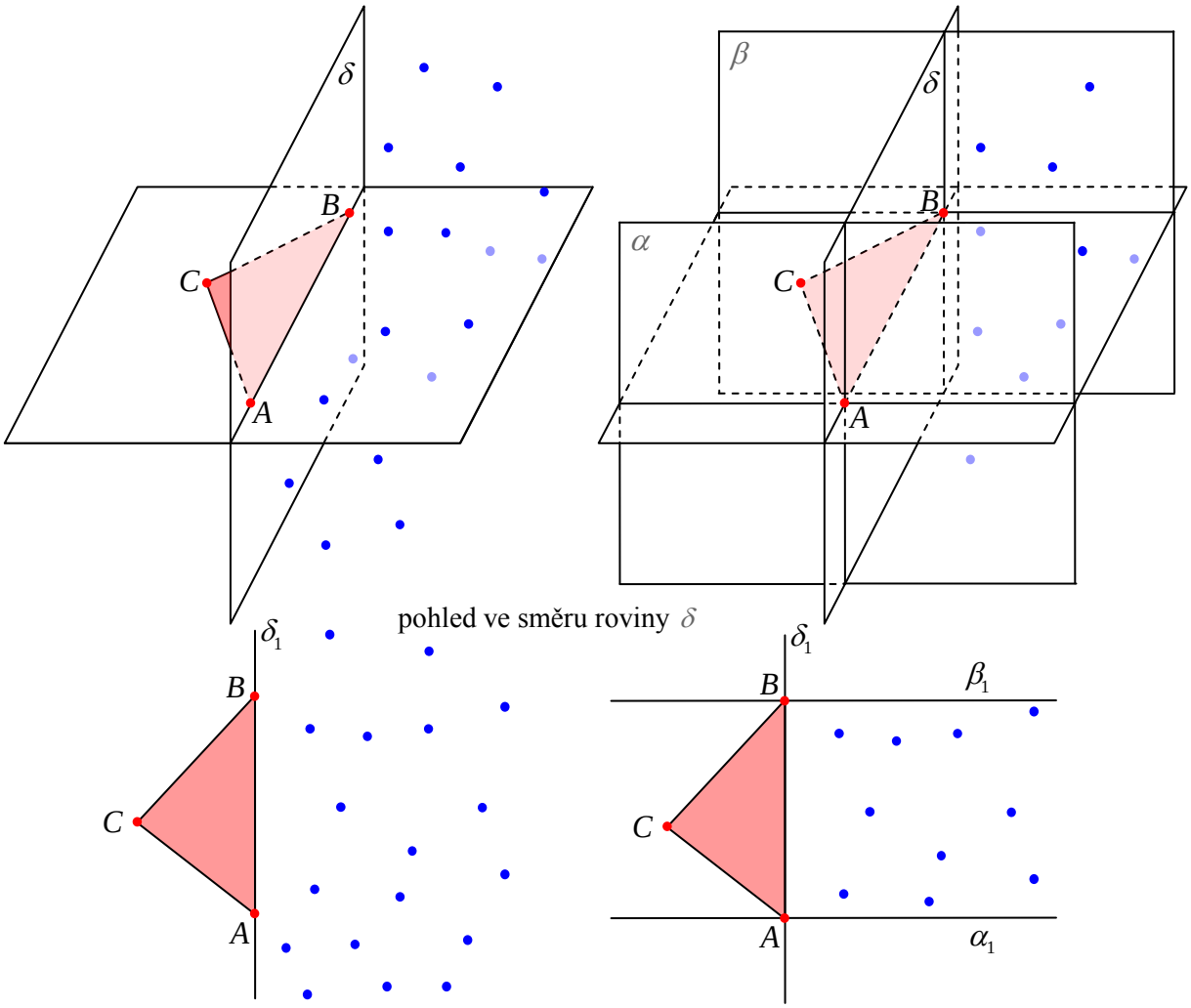
(5.8) $$c^2 = a^2 + b^2 - 2ab\cos\gamma,$$

kde a, b, c jsou po řadě délky stran BC, AC, AB trojúhelníka ABC a γ je vnitřní úhel v trojúhelníku při vrcholu C.

Body A, B, C určují základní trojúhelník, jak vidíme na obrázku 5.11.



Obrázek 5.11: Určení základního trojúhelníka, kterým začíná postupná tvorba trojúhelníkové sítě



Obrázek 5.12: Výběr vhodných bodů pro hranu AB

Při implementaci algoritmu reprezentujeme výsledný polygonální povrch pomocí ukazatelů do seznamu vrcholů, tj. seznamu bodů vstupní bodové množiny $\mathcal{X}$. Každý trojúhelník polygonální sítě je tedy definován trojicí indexů, tj. ukazatelů do $\mathcal{X}$. Při určení základního trojúhelníka uložíme indexy vrcholů A, B, C do seznamu trojúhelníků tvořících polygonální síť, a hrany AB, AC, BC do fronty, neboť budeme dále určovat, které body vstupní ny $\mathcal{X}$ budeme k těmto hranám připojovat. Aktuální seznam trojúhelníků v triangulaci nazveme *triangles*. Fronta hran je opět reprezentována pomocí ukazatelů do seznamu vrcholů, značíme ji jako *edges*.

Vezměme nyní první hranu AB z fronty *edges* a určeme bod D ze vstupní bodové množiny $\mathcal{X}$, který společně s hranou AB vytvoří nový trojúhelník ABD v povrchové triangulaci. Vhodné kandidáty na bod D vybíráme z poloprostoru určeného rovinou δ kolmou k rovině trojúhelníka ABC, v němž neleží vrchol C. Dále omezíme tento výběr podmínkou, že možní kandidáti na bod D leží mezi dvěma rovnoběžnými rovinami α a β. Rovina α obsahuje vrchol A a je kolmá ke hraně AB, rovina β obsahuje vrchol B a je rovněž kolmá ke hraně AB. Tato podmínka zajistí, že výsledné trojúhelníky v triangulaci mají vnitřní úhly při vrcholech ostré. Výslednou množinu bodů ležících v pásu rovin α a β značme jako *adept_points*. Tyto dvě podmínky omezení výběru vhodných bodů ilustruje obrázek 5.12.

Je-li množina *adept_points* prázdná, hranu, ke které hledáme vhodný bod, vyřadíme z fronty *edges*, jedná se o okrajovou hranu v povrchové triangulaci. Jinak z množiny bodů *adept_points* budeme dále na základě několika geometrických pravidel postupně odebírat body, které tato pravidla nesplní, dokud nezískáme jeden bod D. Pravidla uvedená ve jmenovaných zdrojích jsou následující (používáme rovněž stejné pojmenování):

- pravidlo *prahové vzdálenosti*,
- pravidlo *úhlednosti*,
- pravidlo *maximálního úhlu dvou rovin*,
- pravidlo *maximalizace vnitřních úhlů přilehlých ke hraně* AB.

První pravidlo (pravidlo *prahové vzdálenosti*), pomocí něhož se z množiny *adept_points* vyčleňují další body, je založeno na výpočtu vzdáleností každého bodu množiny *adept_points* od středu hrany AB. Odstraní se ty body množiny *adept_points*, které neleží od středu hrany AB ve vzdálenosti menší než je předem daná prahová vzdálenost λ. Hodnota λ je stanovena experimentálně, její volbu uvádíme u konkrétních příkladů bodových množin.

Body splňující první pravidlo, postupují k pravidlu druhému (pravidlo *úhlednosti*). Nejvhodnější body se vybírají na základě úhlu, který svírá normálový vektor roviny trojúhelníka ABC s normálovým vektorem roviny trojúhelníka, který by mohl být přidán ke hraně AB (předpokládáme stejnou orientaci normálových vektorů). Je-li tento úhel ostrý, splňuje uvažovaný bod druhé pravidlo a postupuje dále. Množina bodů *adept_points* je zúžena o body, pro které vycházejí příslušné úhly větší než pravé.

Třetí pravidlo (pravidlo *maximálního úhlu dvou rovin*) vybere z množiny *adept_points* ten bod, pro který vychází úhel roviny trojúhelníka *ABC* s rovinou trojúhelníka, který by mohl být přidán ke hraně *AB*, největší možný (tj. nejmenší možný úhel normálového vektoru roviny trojúhelníka *ABC* s normálovým vektorem roviny trojúhelníka, který by mohl být přidán ke hraně *AB*).

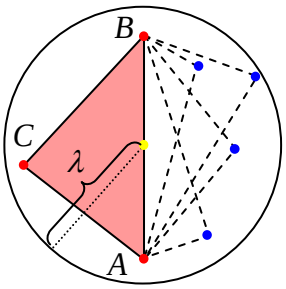
Pokud třetím pravidlem nezískáme pouze jeden bod, tj. existují body, které splňují první a druhé pravidlo a uvažované trojúhelníky svírají s rovinou trojúhelníka *ABC* shodný úhel, přichází na řadu poslední čtvrté pravidlo (pravidlo *maximalizace vnitřních úhlů přilehlých ke hraně AB*). Jako bod *D* se vybere ten bod, pro který menší z vnitřních úhlů trojúhelníka *ABD* přilehlých ke hraně *AB* vychází větší.

V práci (Pešková, 2011) se tato pravidla používají i s předchozími dvěma podmínkami omezení výběru bodů, jak ukazuje obrázek 5.12. Po bližším zkoumání je však zřejmé, že pravidlo úhlednosti je zbytečné, neboť jsme toto pravidlo nahradili podmínkou výběru bodů z poloprostoru určeného rovinou δ kolmou k rovině trojúhelníka *ABC*, v němž neleží vrchol *C*, a pravidlem výběru bodů z pásu rovin α a β. Při použití třetího pravidla maximálního úhlu dvou rovin na našich příkladech se ukázalo, že nedostáváme vždy uspokojivé výsledky. Pravidlo úhlednosti a maximálního úhlu dvou rovin proto nahrazujeme vlastním pravidlem *minimalizace úhlu dvou normál*.

Při konstrukci povrchové triangulace tedy používáme podmínky omezení výběru bodů podle obrázku 5.12 a pravidla

- pravidlo *prahové vzdálenosti*,
- pravidlo *minimalizace úhlu dvou normál*,
- pravidlo *maximalizace vnitřních úhlů přilehlých ke hraně AB*.

Princip prvního pravidla je znázorněn na obrázku 5.13.

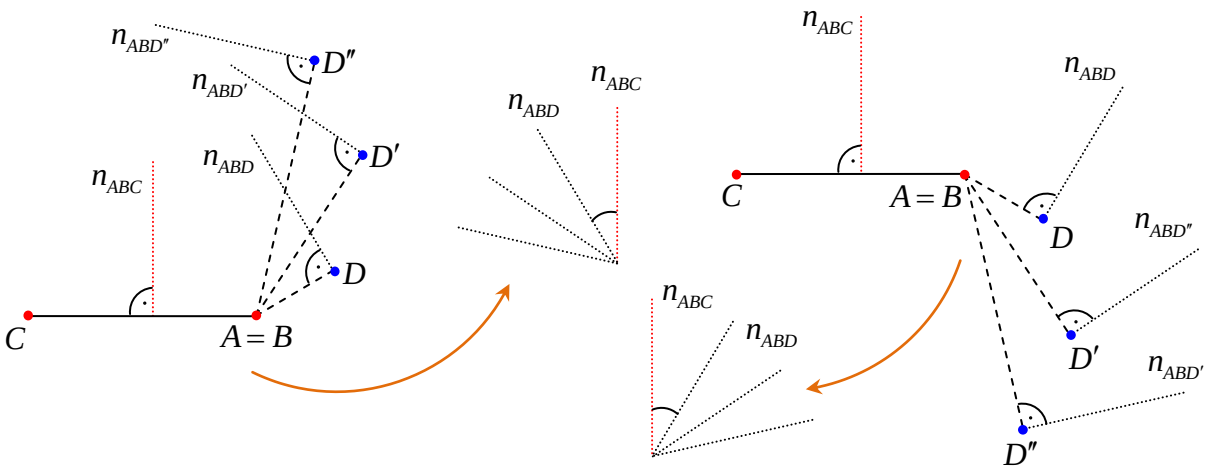


Obrázek 5.13: Pravidlo prahové vzdálenosti, modře označené body jsou adepty pro vrchol *D* trojúhelníka *ABD*

Nově zavedené pravidlo (pravidlo *minimalizace úhlu dvou normál*) funguje následovně. Z bodů, které prošli prvním pravidlem a tvoří nyní množinu *adept_points*, vybereme nejvhodnější bod na základě úhlu, který svírá normála roviny trojúhelníka *ABC* s normálou roviny trojúhelníka, který by mohl být přidán ke hraně *AB*. Vybíráme ten bod, pro který tento úhel vychází nejmenší možný. Princip tohoto pravidla můžeme vidět na obrázku 5.14, trojúhelníky jsou zobrazeny v pohledu ze strany, zobrazují se do úseček.

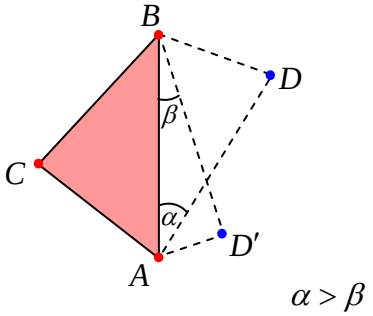
Pokud druhým pravidlem nezískáme pouze jeden bod, tj. existují body, které splňují první pravidlo a uvažované trojúhelníky svírají s rovinou trojúhelníka *ABC* shodný úhel (tedy mezi uvažovanými trojúhelníky existují trojúhelníky, jejichž roviny mají normály shodné-

ho směru), přichází na řadu poslední pravidlo. Užití třetího pravidla je zobrazeno na obrázku 5.15.



Obrázek 5.14: Pravidlo minimalizace úhlu dvou normál – zde v obou případech vybíráme bod *D*, neboť normály trojúhelníka *ABC* a *ABD* svírají nejmenší úhel

Předpokládejme nyní, že jsme použitím těchto pravidel nalezli bod D. Do výsledné triangulace tedy přidáváme nový trojúhelník ABD. Zaktualizujeme seznam *triangles* trojúhelníků v triangulaci, tj. přidáme indexy vrcholů A, B, D a z fronty *edges* odstraníme hranu AB. Do fronty hran *edges* přidáme nové dvě hrany AD a BD. Ve frontě *edges* jsou nyní čtyři hrany, tj. $edges = \{AC, BC, AD, BD\}$ a v seznamu trojúhelníků dva trojúhelníky, tj. $triangles = \{ABC, ABD\}$.



Obrázek 5.15: Pravidlo maximalizace vnitřních úhlů přilehlých ke hraně *AB* – zde vybíráme bod *D*

V dalším kroku algoritmu vybíráme z fronty *edges* další hranu a opakujeme postup výběru vhodného bodu pro vytvoření dalšího trojúhelníka v triangulaci. To znamená, že určíme body v pásu rovnoběžných rovin a dále uplatníme popsaná geometrická pravidla. Tento postup budeme provádět do té doby, dokud není fronta hran *edges* prázdná.

5.2.1 Ošetření speciálních případů

Pokud se užitím prvního nebo druhého pravidla nalezne pouze jeden bod, další pravidla se neuplatňují a tento bod je přidán do triangulace. V případě, že se prvním nebo druhým pravidlem nenalezne žádný vhodný bod, zavádíme nově pravidlo, že hranu, ke které hledáme vhodný bod, prozatím vyřadíme z fronty *edges*.

Při použití těchto geometrických pravidel je nutné ošetřit speciální případy, kdy ve výsledné triangulaci vzniká nežádoucí křížení trojúhelníků. V článku (Yu *a kol.*, 2010)

nebyly tyto případy řešeny vůbec, v práci (Pešková, 2011) částečně – tento přístup však nepoužíváme, navrhujeme vlastní metody.

	left – 1.	*right* – 1.	*left* – 2.	*right* – 2.	Výsledek
1.	*False*	*False*	*False*	*False*	*edges_first* ← *left* *edges_first* ← *right*
2.	*False*	*False*	*False*	*True*	nenastane
3.	*False*	*False*	*True*	*False*	nenastane
4.	*False*	*False*	*True*	*True*	nenastane
5.	*False*	*True*	*False*	*False*	*edges_first* ← *left* *edges_second* ← *right*
6.	*False*	*True*	*False*	*True*	křížení
7.	*False*	*True*	*True*	*False*	nenastane
8.	*False*	*True*	*True*	*True*	nenastane
9.	*True*	*False*	*False*	*False*	*edges_second* ← *left* *edges_first* ← *right*
10.	*True*	*False*	*False*	*True*	nenastane
11.	*True*	*False*	*True*	*False*	křížení
12.	*True*	*False*	*True*	*True*	nenastane
13.	*True*	*True*	*False*	*False*	*edges_second* ← *left* *edges_second* ← *right*
14.	*True*	*True*	*False*	*True*	křížení
15.	*True*	*True*	*True*	*False*	křížení
16.	*True*	*True*	*True*	*True*	křížení

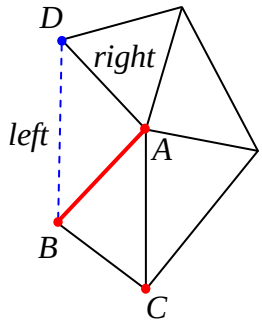
Tabulka 5.1: Případy při přidání trojúhelníka do povrchové triangulace

K detekci chybných trojúhelníků v síti a jejich vyřazení navrhujeme použít kontrolu, kolikrát se přidávaná hrana nebo hrana ve frontě *edges* objevuje v triangulaci. Nechť *ABC* je trojúhelník, k jehož hraně *AB* v daném kroku konstrukce polygonální sítě hledáme bod *D*. Předpokládáme, že se nacházíme v libovolném kroku inkrementální konstrukce, to znamená, že trojúhelník *ABC* nemusí být již základním trojúhelníkem. Rozeberme případy, které mohou nastat při přidání trojúhelníka *ABD* do triangulace, tj. do seznamu *triangles*. Označme hranu *AD* jako *left*, hranu *BD* jako *right*. Nyní zkoumáme, kolikrát jsou hrany *left* a *right* v aktuální povrchové triangulaci. K tomuto účelu vytvoříme seznam hran použitých v triangulaci povrchu poprvé, ozn. *edges_first*, a seznam hran použitých v triangulaci povrchu podruhé, ozn. *edges_second*. Je-li např. hrana *left* již v triangulaci, tj. je v seznamu *edges_first*, přidáme ji do fronty *edges* a zároveň do seznamu *edges_second*. Je-li hrana *left* v triangulaci již dvakrát, tj. je v seznamu *edges_second*, do fronty *edges* ji nepřidáme.

Můžou nastat různé kombinace případů. Jelikož máme dvě hrany *left* a *right* a každá z nich v triangulaci buď ještě není, nebo je jednou, nebo dvakrát, dostáváme celkem 16 případů. Některé případy je možné rovnou vyloučit, protože nenastanou, ostatní jednotlivě vyřešíme. Případy jsou popsány v tabulce 5.1.

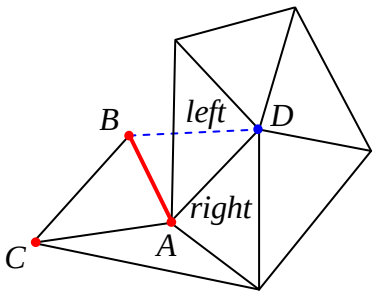
Případy, které nemohou po přidání trojúhelníka *ABD* do triangulace nastat, jsou zřejmé, v tabulce 5.1 jsou označeny šedě.

Ve čtyřech situacích, které jsou označeny zeleně, trojúhelník *ABD* do triangulace přidáme, ovšem rozlišíme možnosti doplnění nových hran do fronty *edges*. V případě 1. doplníme do fronty *edges* obě hrany *left* a *right*, v případě 5. pouze hranu *left*, v případě 9. pouze hranu *right* a v posledním případě 13. žádnou hranu, neboť se jedná o vyplnění díry v povrchové triangulaci. Vždy musíme zároveň kontrolovat, zda se přidáním trojúhelníka *ABD* ve frontě hran nedostaneme do situace, že je nějaká hrana ve frontě čekající na zpracování sdílena již dvěma trojúhelníky. Takovou situaci znázorňuje obrázek 5.16, jedná se o 5. případ (F-T-F-F). Po přidání nového trojúhelníka *ABD* do triangulace proto odstraňujeme ty hrany z fronty *edges*, které patří dvěma sousedním trojúhelníkům, neboť každá hrana v triangulaci může být společná nejvýše dvěma trojúhelníkům.



Obrázek 5.16: Případ F-T-F-F z tabulky 5.1, do fronty *edges* doplníme pouze hranu *left*

Ve zbývajících případech, které jsou v tabulce označeny červeně, by po přidání trojúhelníka *ABD* do povrchové triangulace došlo k nežádoucímu křížení nově připojovaného trojúhelníka s již existujícími trojúhelníky v triangulaci. V takovém případě použijeme nově zavedené pravidlo, že hranu *AB*, ke které hledáme vhodný bod, prozatím vyřadíme z fronty *edges*. Příklad takového nežádoucího křížení trojúhelníků můžeme vidět na obrázku 5.17, přičemž zde se speciálně jedná o 6. situaci z tabulky 5.1.



Obrázek 5.17: Případ křížení F-T-F-T z tabulky 5.1, hranu *AB* prozatím vyřazujeme z fronty *edges*

Tím, že během konstrukce trojúhelníkové sítě některé hrany, ke kterým hledáme bod jako vrchol nového trojúhelníka, vynecháváme, je třeba doplnit další vylepšení konstrukce povrchové triangulace. Pokud totiž k hranám z fronty *edges* v daném kroku konstrukce nenalezneme vhodný bod jako třetí vrchol nového trojúhelníka, nemusí to znamenat, že by vhodný bod neexistoval. Proto může po použití všech hran z fronty *edges* docházet k tomu, že se nevytvoří celý povrch, některé body vstupní ny $\mathcal{X}$ totiž nebudou sdíleny žádnými trojúhelníky. Ve výsledné triangulaci mohou být díry nebo je povrch nedokončený.

Pro řešení této situace zavádíme proto nově možnost restartování konstrukce trojúhelníkové sítě. To znamená, že dostaneme-li se nakonec fronty *edges*, již nemáme hranu,

ke které bychom hledali vhodný vrchol nového trojúhelníka, spustíme proces tvorby sítě znovu pro novou frontu hran. Nová fronta hran, označme ji *new_edges*, bude tvořena hranami, které již v povrchové triangulaci jsou, ovšem patří pouze jednomu trojúhelníku. Hrany ve frontě *new_edges* jsou tedy okrajové hrany děr nebo okrajové hrany sítě. Jedná-li se o hrany, které skutečně tvoří okraj výsledného triangulovaného povrchu, další postup konstrukce sítě tyto hrany vyřadí z fronty *new_edges*. Restartování procesu tvorby povrchové triangulace může být spuštěno vícekrát, počet tzv. restartů je řízen uživatelem na základě vizuálního posouzení dosavadní triangulace. Jsou-li všechny body vstupní množiny zpracovány, tj. náleží alespoň jednomu trojúhelníku, a triangulace neobsahuje díry, proces tvorby je zastaven. Při novém spuštění konstrukce je rovněž možná změna hodnoty prahové vzdálenosti λ. Ukázku takového restartování můžeme vidět v příkladě 5.2.

Průběh konstrukce povrchové triangulace si nyní ukažme na několika příkladech experimentálních vstupních bodových množin.

Příklad 5.2: *INKREMENTÁLNÍ KONSTRUKCE POLYGONÁLNÍ SÍTĚ.* Mějme dánu množinu $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které popisují nějaký povrch. V našich případech se jedná většinou o rotační plochy nebo jejich části, případně jejich různé kombinace. V následujícím příkladě demonstrujme předvedený algoritmus na několika takových vstupních bodových množinách. Ve všech případech volíme záměrně řídkou množinu bodů kvůli přehlednosti. Vždy uvádíme hodnotu prahové vzdálenosti λ.

V prvním případě se jedná o body, které jsou rozloženy pravidelně na části rotační válcové plochy, jak vidíme na obrázku 5.18, kde je rovněž zakreslen základní trojúhelník, od kterého začíná postupná konstrukce sítě. Obrázek 5.19 znázorňuje postupný výpočet trojúhelníkové sítě, přičemž je vždy zvýrazněna hrana, ke které se v daném kroku hledá vhodný vrchol nového trojúhelníka. Obrázek 5.20 zachycuje výslednou povrchovou triangulaci bodové množiny.

Obdobným způsobem je prezentována konstrukce trojúhelníkové sítě pro množinu bodů, jež jsou na povrchu části rotační válcové plochy rozloženy pravidelně, ovšem množina obsahuje pouze body jednoho poloprostoru určeného rovinou procházející osou plochy. Zpracování této bodové množiny je ukázáno na obrázcích 5.21-5.23.

V dalším případě se věnujme množině bodů, které jsou navzorkovány pravidelně na obecnější rotační ploše. Rotační plochu, ze které jsme získali body pro testování, jsme určili pomocí polomeridiánu, který je složen z oblouků kružnic, osu rotační plochy jsme zvolili v ose *z*. Jedná se o stejnou bodovou množinu, kterou jsme užili v oddíle 4.3.1. Postupná tvorba trojúhelníkové sítě je předložena na obrázku 5.24 a výsledná triangulace na obrázku 5.25.

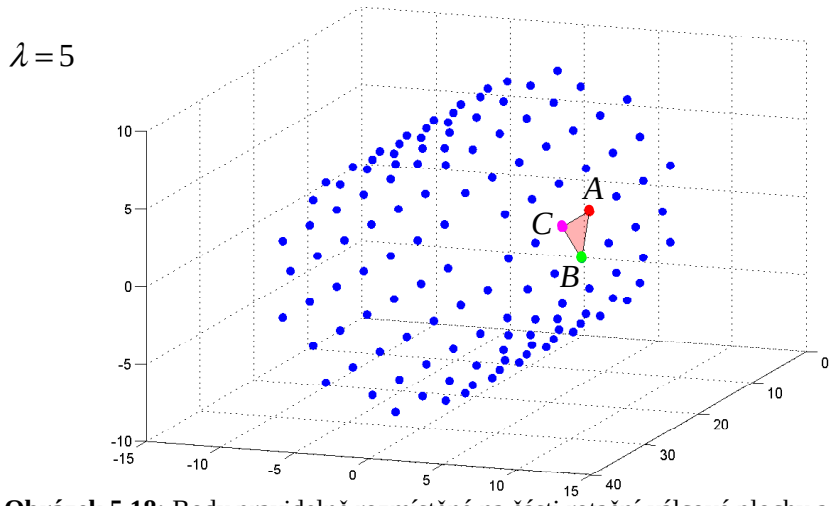
Další bodová množina je rovněž navzorkována pravidelně na části obecnější rotační plochy, kterou jsme určili rotací polomeridiánu, jenž je složen z oblouků kružnic. Osu rotační plochy jsme zvolili opět v ose *z*. Rotace polomeridiánu kolem osy je provedena o úhel menší než $360°$. Opět se jedná o stejnou bodovou množinu, kterou jsme užili v oddíle 4.3.1. Obrázky 5.26-5.27 ilustrují tvorbu sítě.

V těchto čtyřech případech stačí k sestrojení sítě jeden průchod, nebylo tedy nutné restartování procesu konstrukce sítě.

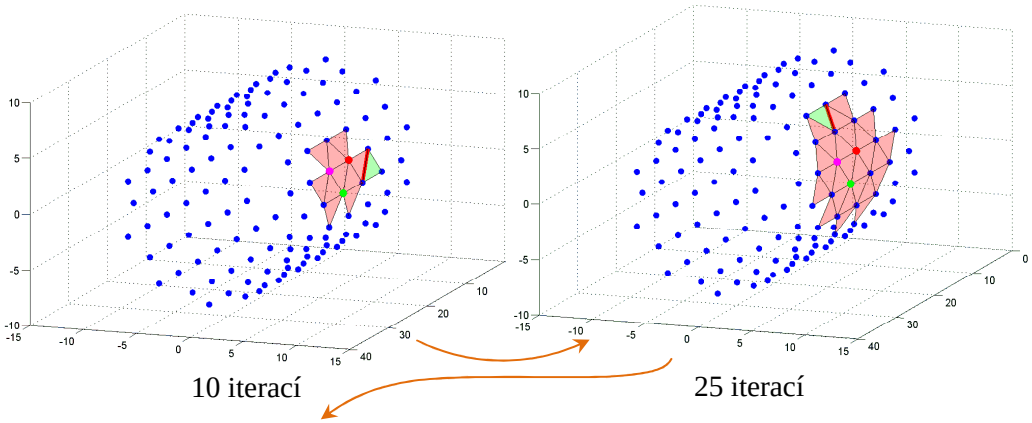
Poslední bodovou množinu, kterou zde ukážeme, je množina bodů získaných z povrchu rotačního jednodílného hyperboloidu. Původní pravidelná síť bodů je ve všech souřadnicových směrech zašuměna, viz obrázek 5.28, tj. ke každé souřadnici je přičteno náhodné číslo s normálním rozdělením. Zde se již jedná o množinu bodů, kdy je nutné použít restartování procesu konstrukce sítě. Na obrázku 5.28 je vidět první okamžik, kdy se konstrukce sítě zastaví, neboť je již prázdná fronta hran *edges*, které čekají na zpracování. V případě zastavení konstrukce, počítáme novou frontu hran *new_edges* (zeleně), která obsahuje okrajové hrany děr nebo okrajové hrany sítě. Na obrázku 5.29 je vidět postupná konstrukce trojúhelníkové sítě v několika okamžicích, kdy se konstrukce sítě zastaví a proces tvorby sítě se restartuje. Výsledek povrchové triangulace znázorňuje obrázek 5.30.

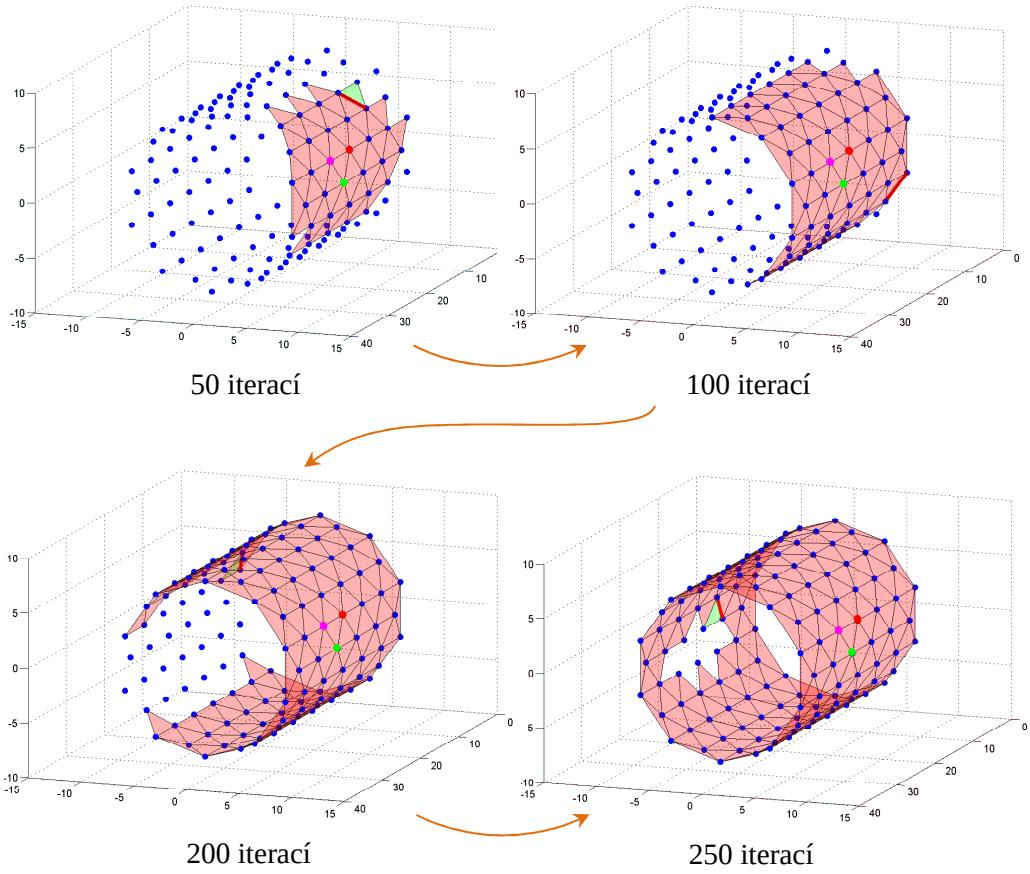
Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

`inkrementalni_povrch.m` (programy/povrch/inkrement).

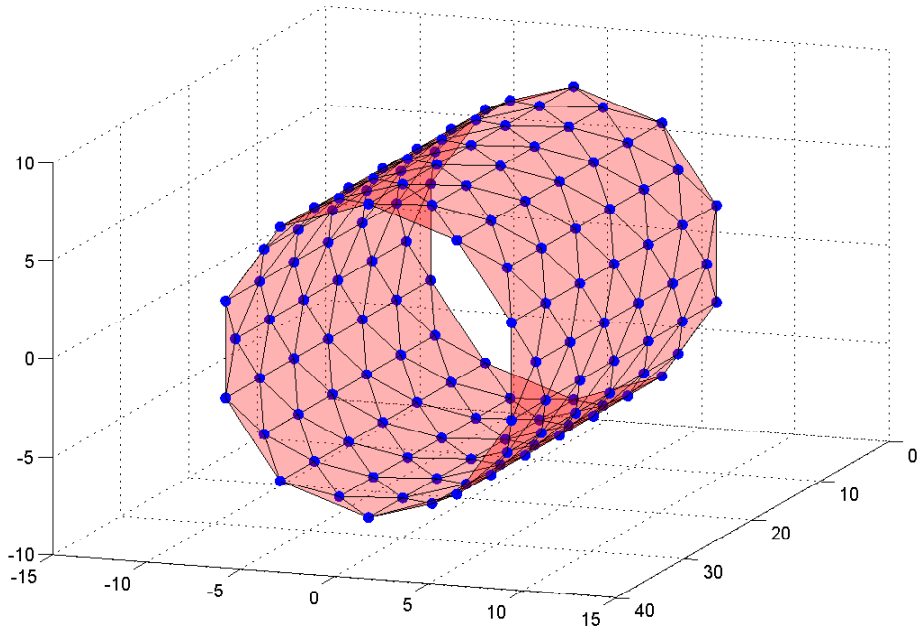


Obrázek 5.18: Body pravidelně rozmístěné na části rotační válcové plochy a základní trojúhelník *ABC*

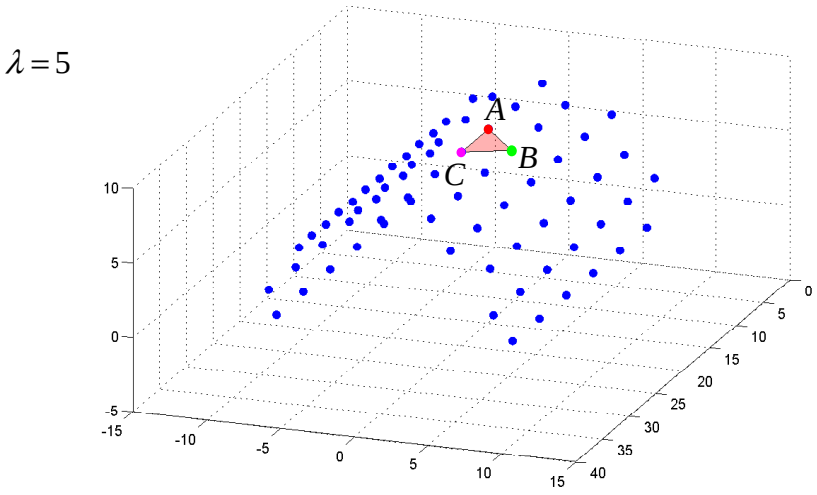




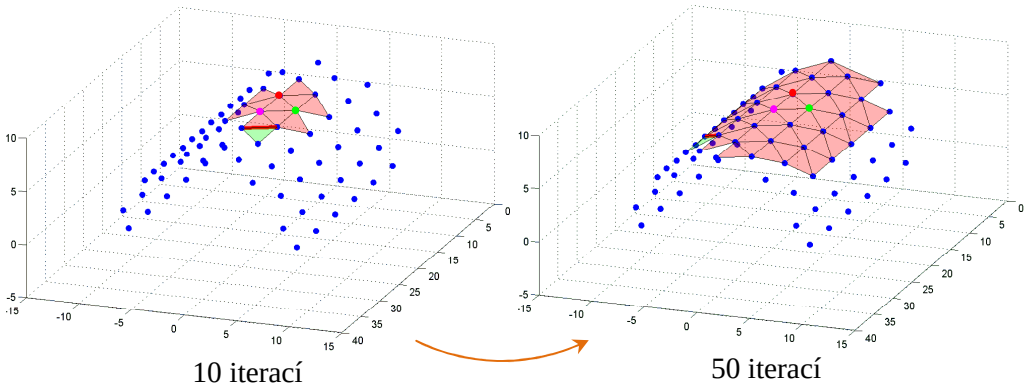
Obrázek 5.19: Postupná konstrukce trojúhelníkové sítě



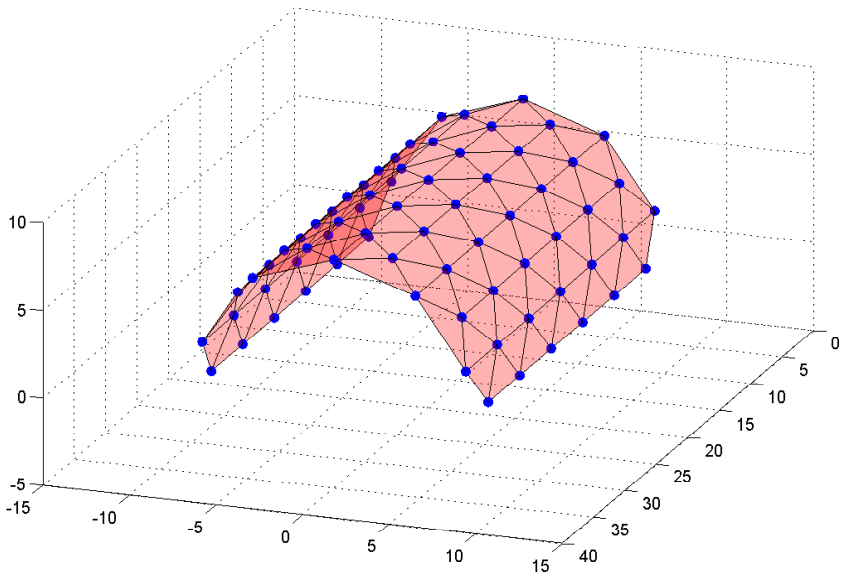
Obrázek 5.20: Výsledná povrchová triangulace



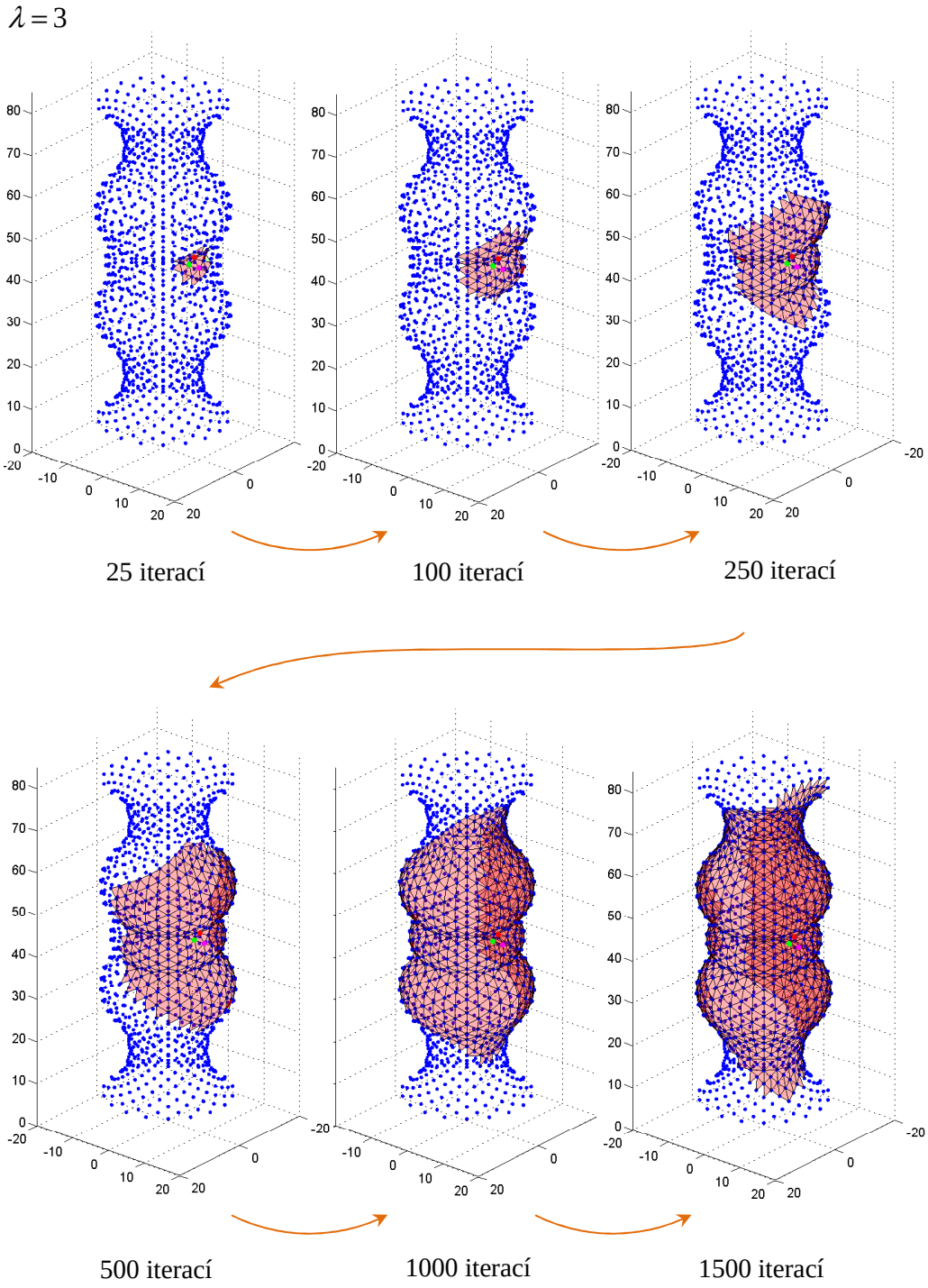
Obrázek 5.21: Body pravidelně rozmístěné na části rotační válcové plochy a základní trojúhelník *ABC*



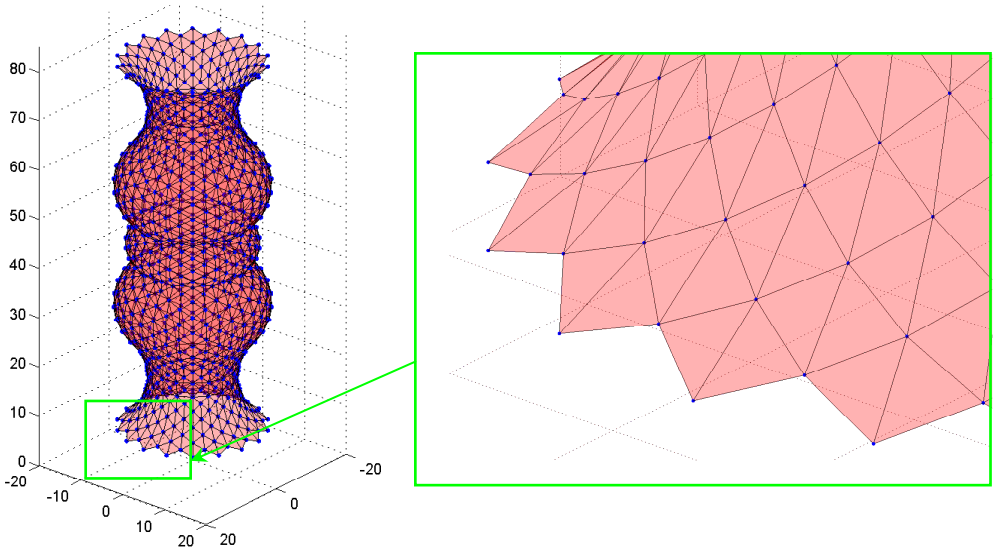
Obrázek 5.22: Postupná konstrukce trojúhelníkové sítě



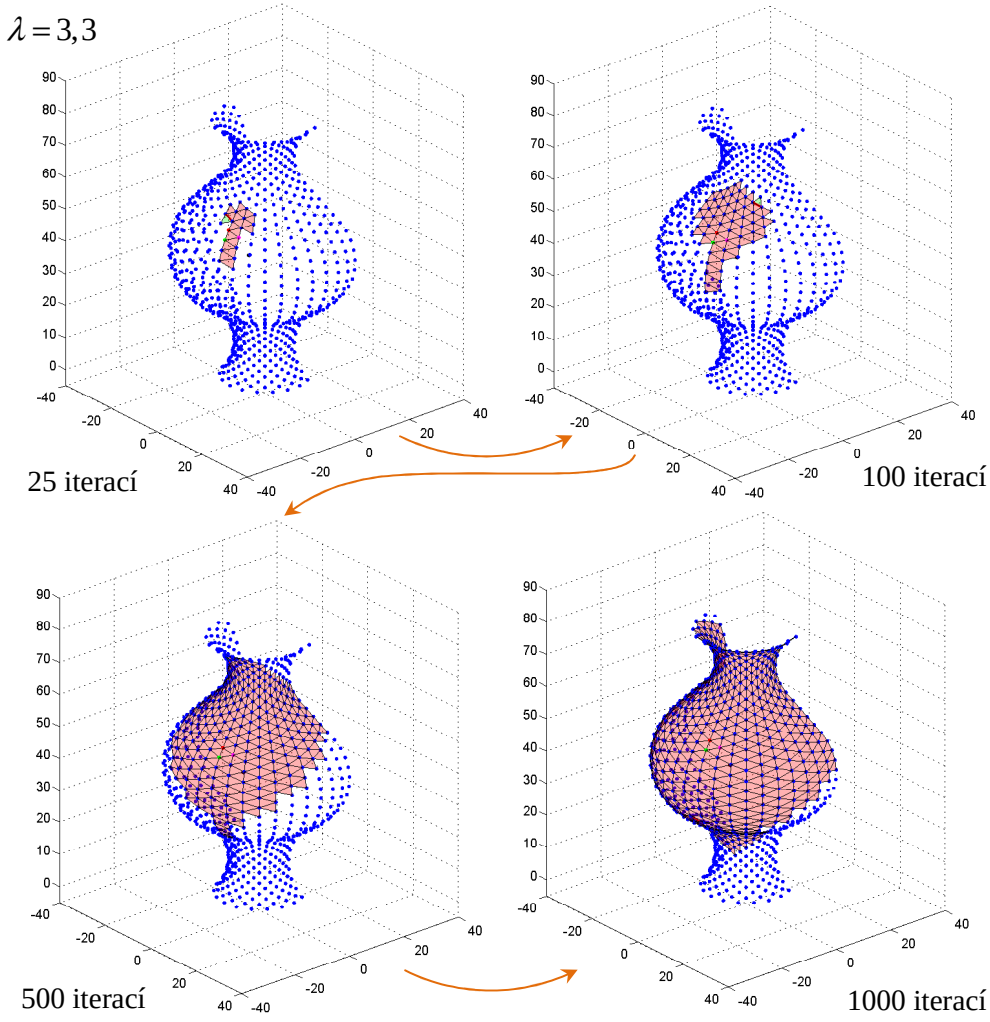
Obrázek 5.23: Výsledná povrchová triangulace



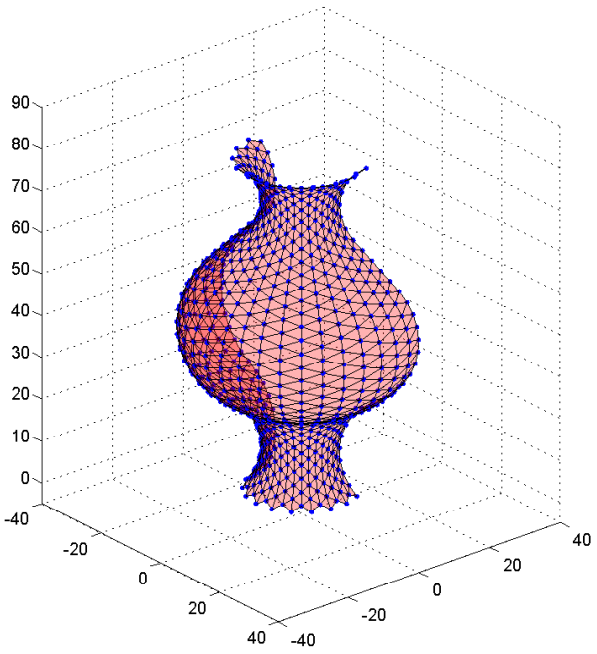
Obrázek 5.24: Postupná konstrukce trojúhelníkové sítě



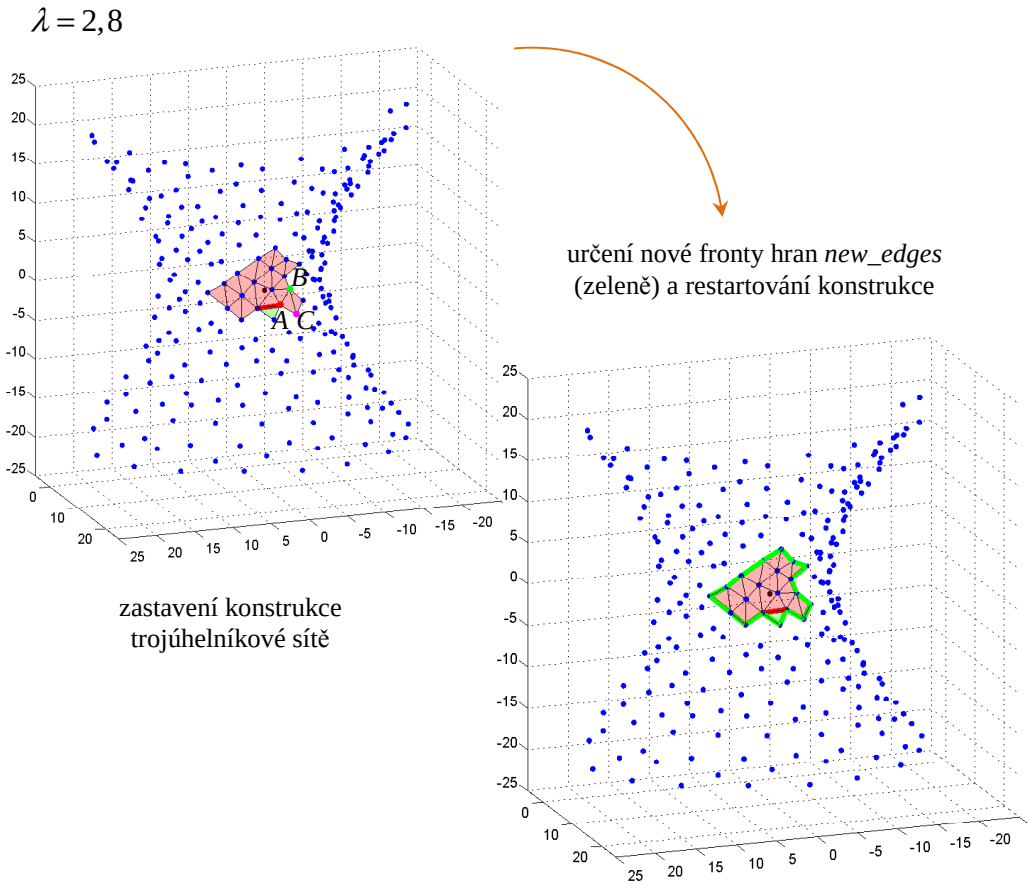
Obrázek 5.25: Výsledná povrchová triangulace a její detail



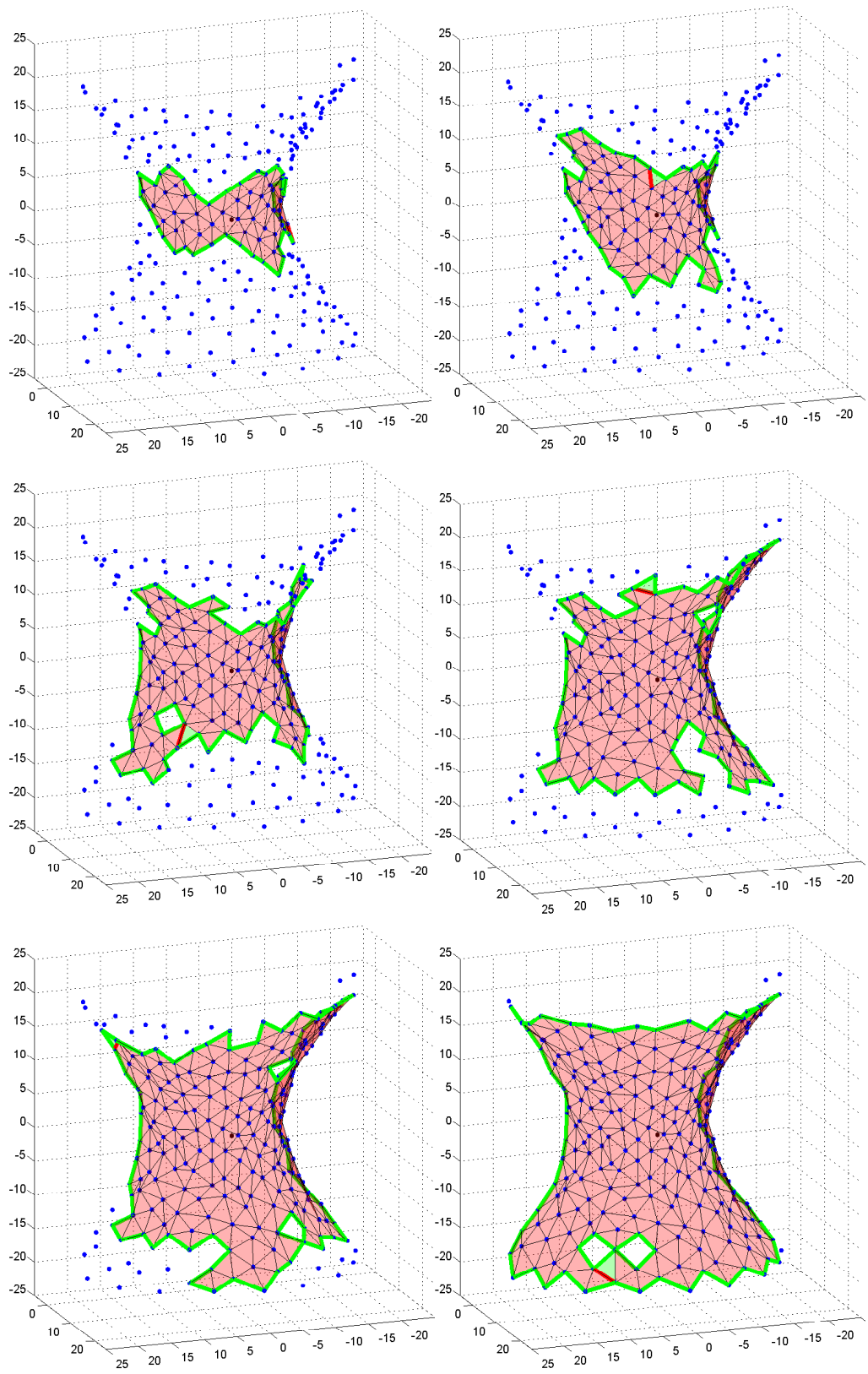
Obrázek 5.26: Postupná konstrukce trojúhelníkové sítě



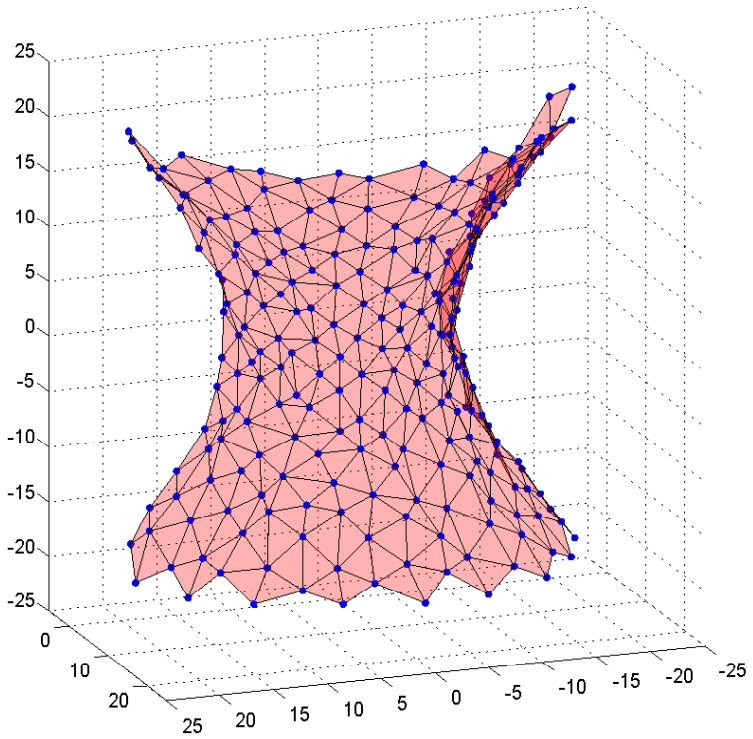
Obrázek 5.27: Výsledná povrchová triangulace



Obrázek 5.28: Zašuměná množina bodů získaná z části povrchu rotačního jednodílného hyperboloidu – první zastavení konstrukce a restartování procesu tvorby sítě



Obrázek 5.29: Zastavení konstrukce povrchové triangulace, výpočet okrajových částí a děr v síti a restartování procesu tvorby sítě



Obrázek 5.30: Výsledná povrchová triangulace

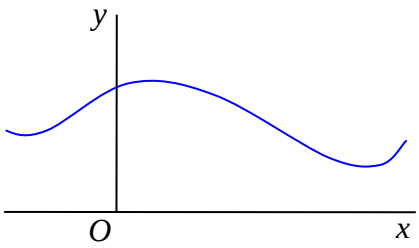
Nutno podotknout, že při použití inkrementální konstrukce sítě můžeme teoreticky narazit na další typy nežádoucího křížení trojúhelníků. Například mohou vznikat trojúhelníky, které mají společný vrchol a jejich roviny svírají velmi malý úhel. Řešení takové situace je již velmi komplikované. Při testování algoritmu na bodových množinách, které jsme měli k dispozici, však k takovému křížení nedocházelo. V budoucí práci plánujeme tyto speciální případy také ošetřit.

Při testování algoritmu na reálných bodových množinách jsme se rozhodli pro další vylepšení a to alternativní výběr základního trojúhelníka. Kvůli restartování procesu konstrukce sítě je v některých případech výhodnější začínat konstrukci od trojúhelníka, jehož první hrana má nejmenší možnou délku. To znamená, že v prvním kroku algoritmu hledáme nejkratší spojnici dvou bodů, které budou tvořit první hranu AB v triangulaci. Třetí vrchol, bod C, spočteme stejně, jako jsme zavedli na začátku tohoto oddílu. To znamená, že vnitřní úhel v trojúhelníku při vrcholu C je největší možný. Body A, B, C tedy určují základní trojúhelník.

Další příklady inkrementální konstrukce povrchové triangulace uvedeme později pro reálné bodové množiny, které obsahují velké množství bodů.

5.3 Triangulace povrchu pomocí transformace

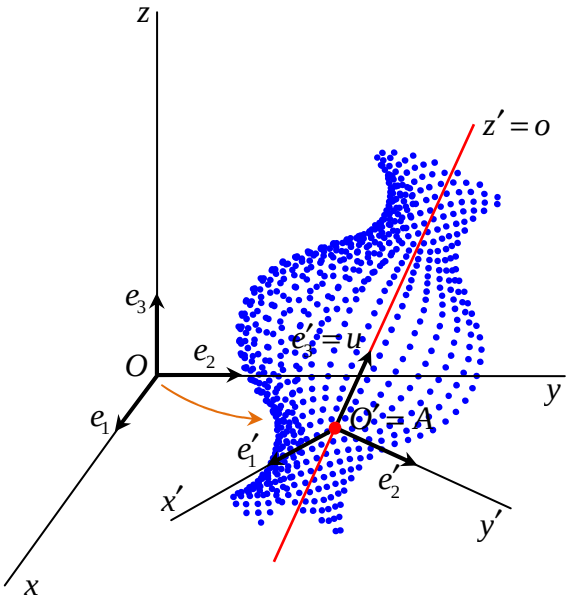
V následující části se zaměříme na další možnost tvorby polygonální sítě reprezentující zkoumaný povrch založenou na transformaci bodů mračna do cylindrických souřadnic. Tento převod budeme využívat v případech, pracujeme-li s rotačními plochami, jejich částmi nebo kombinacemi. Nechť je dána neprázdná množina $\mathcal{X}$ n bodů v eukleidovském prostoru E_3 popisující rotační plochu, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Body vstupní množiny $\mathcal{X}$ leží na nebo velmi blízko povrchu. Vždy budeme předpokládat, že polomeridián rotační plochy, ze které jsme získali skenováním jejího povrchu bodové mračno $\mathcal{X}$, je po zobrazení do kartézské soustavy souřadnic v rovině s osou x odpovídající ose polomeridiánu grafem funkce. Vysvětlení tohoto předpokladu můžeme vidět na obrázku 5.31.



Obrázek 5.31: Zobrazení polomeridiánu v pomocné soustavě souřadnic v rovině – polomeridián jako graf funkce

Hledáme povrchovou triangulaci tak, že vrcholy trojúhelníků jsou všechny body množiny $\mathcal{X}$ a každá hrana je společná nejvýše dvěma trojúhelníkům. Algoritmus, který navrhujeme, má několik dílčích kroků. V prvním kroku algoritmu je nutné nalézt osu symetrie bodové množiny. K nalezení osy používáme nejčastěji metodu hledání osy rotační plochy navrženou v oddíle 4.3 (*Hledání osy obecné rotační plochy*), případně metody odvozené v oddílech 3.3 (*Ortogonální prokládání dat přímkou*), 3.6 (*Analýza hlavních komponent*) nebo 4.2 (*Hledání osy rotační válcové plochy*).

V obecném případě neodpovídá nalezená osa žádné souřadnicové ose, dalším krokem algoritmu je proto transformace kartézské soustavy souřadnic v prostoru do nové polohy. Nechť je nalezená osa symetrie o určena bodem A a jednotkovým směrovým vektorem u. Ve všech případech pro jednoduchost volíme takovou transformaci, že nalezená osa symetrie o splývá s osou z' nové kartézské soustavy souřadnic v prostoru a počátek O' nové kartézské soustavy souřadnic je umístěn do určujícího bodu A nalezené osy symetrie. To znamená, že označíme-li původní kartézskou soustavu souřadnic $S = \{O, e_1, e_2, e_3\}$ a novou kartézskou soustavu souřadnic $S' = \{O', e_1', e_2', e_3'\}$, potom počátek O posuneme do určujícího bodu A odhadnuté osy a vektory e_1, e_2, e_3 přejdou



Obrázek 5.32: Transformace kartézské soustavy souřadnic v prostoru do nové polohy

do vektorů e'_1, e'_2, e'_3, tj. počátek $O' = A$ a jednotkový vektor $e'_3 = u$. Jednotkové vektory e'_1 a e'_2 volíme libovolně tak, aby společně s vektorem e'_3 tvořily ortonormální bázi. Podívejme se na obrázek 5.32. Transformaci tří navzájem kolmých vektorů do nové polohy jsme podrobně probrali v oddíle 4.4, nebudeme tedy znovu vypisovat vzorce potřebné pro výpočet. Množinu bodů v nové kartézské soustavě souřadnic $S' = \{O', e'_1, e'_2, e'_3\}$ budeme značit $\mathcal{X}' = \{X'_i\}_{i=1}^n$.

Dalším krokem algoritmu je převedení souřadnic bodů $\{X'_i\}_{i=1}^n$ mračna $\mathcal{X}'$ z kartézské soustavy souřadnic $S' = \{O', e'_1, e'_2, e'_3\}$ v prostoru do cylindrických souřadnic. K převodu souřadnic bodu X'_i užijeme známého vzorce

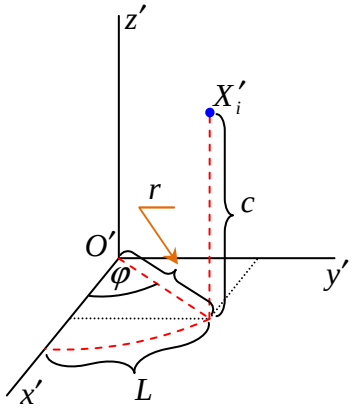
$$\begin{aligned} \varphi &= \operatorname{arctg} 2(y', x'), \\ c &= z', \\ r &= \sqrt{(x')^2 + (y')^2}, \end{aligned} \tag{5.9}$$

kde x', y', z' jsou souřadnice bodu X'_i v kartézské soustavě souřadnic S' a funkce $\operatorname{arctg} 2(y', x')$ je definována následovně

$$\operatorname{arctg} 2(y', x') = \begin{cases} \operatorname{arctg}\left(\dfrac{y'}{x'}\right), \text{ je-li } (x' > 0) \wedge (y' > 0), \\ \operatorname{arctg}\left(\dfrac{y'}{x'}\right) + \pi, \text{ je-li } (x' < 0), \\ \operatorname{arctg}\left(\dfrac{y'}{x'}\right) + 2\pi, \text{ je-li } (x' > 0) \wedge (y' < 0). \end{cases} \tag{5.10}$$

Převod do cylindrických souřadnic podle vzorce (5.9) pro naše účely mírně modifikujme a to dvěma způsoby. V prvním případě místo úhlu φ budeme uvažovat orientovanou délku oblouku příslušného tomuto úhlu, značit ji budeme L. K převodu souřadnic bodů $\{X'_i\}_{i=1}^n$ mračna $\mathcal{X}'$ budeme užívat vztahů

$$\begin{aligned} L &= \sqrt{(x')^2 + (y')^2}\, \operatorname{arctg} 2(y', x'), \\ c &= z', \\ r &= \sqrt{(x')^2 + (y')^2}\,. \end{aligned} \tag{5.11}$$



Obrázek 5.33: Význam hodnot r, L, c

Význam hodnot r, L, c je ilustrován na obrázku 5.33. Hodnoty r, L, c nyní vyneseme do kartézské soustavy souřadnic S', přičemž hodnotu L nanášíme na osu x', hodnotu c na osu y' a hodnotu r na osu z'. Takto získané body značíme $\{X_i^C\}_{i=1}^n$. Vzorce (5.11) bude-

me užívat pouze v případě, jsou-li body navzorkované na rotační válcové ploše. Potom se totiž jedná o rozvinutí válcové plochy do roviny.

Ve druhém případě, kdy se jedná o body na povrchu obecné rotační plochy, je nutné navrhnout alternativní modifikaci vztahů (5.9). Místo úhlu φ budeme nyní uvažovat orientovanou délku oblouku příslušného tomuto úhlu, ovšem měřit ji budeme na rotační válcové ploše s osou v ose z' s poloměrem rovnajícím se největší hodnotě r. To znamená, že k převodu souřadnic bodů $\{X_i'\}_{i=1}^n$ mračna $\mathcal{X}'$ budeme užívat vztahů

$$\begin{aligned} d &= R\,\mathrm{arctg}\,2(y', x'), \\ c &= z', \\ r &= \sqrt{(x')^2 + (y')^2}\,, \end{aligned} \tag{5.12}$$

kde R je největší hodnota ze všech vzdáleností $\sqrt{(x')^2 + (y')^2}$.

Dále postupujeme stejně, hodnoty r, d, c nyní vyneseme do kartézské soustavy souřadnic S', přičemž hodnotu d nanášíme na osu x', hodnotu c na osu y' a hodnotu r na osu z'. Takto získané body budeme značit rovněž $\{X_i^C\}_{i=1}^n$. Vztahy (5.12) zavádíme kvůli dalšímu postupu, kdy využíváme rovinné triangulace. Vše objasníme později na konkrétních příkladech.

V kartézské soustavě souřadnic S' dostáváme s využitím vztahů (5.11) nebo (5.12) třídimenzionální model povrchu rotační plochy reprezentovaný sítí bodů $\{X_i^C\}_{i=1}^n$. V následujícím kroku algoritmu uvažujeme pouze půdorysy bodů $\{X_i^C\}_{i=1}^n$ a provedeme triangulaci v rovině (x', y').

Definice 5.1 (*Rovinná triangulace*). Triangulace T nad množinou bodů $\mathcal{V} = \{V_i\}_{i=1}^n$ v eukleidovské rovině E_2 je množina trojúhelníků s vrcholy V_i, přičemž každé dva trojúhelníky mají společnou nejvýše jednu hranu nebo vrchol a sjednocení všech trojúhelníků tvoří souvislou množinu v E_2. ■

Definice 5.1 je velmi obecná, pro danou množinu bodů v rovině může takových triangulací existovat mnoho. Obecným požadavkem pro výběr vhodného typu triangulace je především optimální tvar trojúhelníků. V našem případě se rozhodujeme pro *Delaunayovu triangulaci*, neboť tato triangulace produkuje trojúhelníky blížící se rovnostranným.

Definice 5.2 (*Delaunayova triangulace*). Triangulace *DT* nad množinou bodů $\mathcal{V} = \{V_i\}_{i=1}^n$ v eukleidovské rovině E_2 se nazývá *Delaunayova*, jestliže kružnice opsaná libovolnému trojúhelníku v triangulaci neobsahuje žádný jiný bod z množiny $\mathcal{V}$. ■

Uveďme stručně několik vlastností Delaunayovy triangulace. Výsledné trojúhelníky v *DT* se v porovnání se všemi známými triangulacemi nejvíce blíží rovnostranným, to je také hlavní důvod výběru této triangulace pro náš algoritmus. Hranicí *DT* je konvexní obálka. Právě kvůli této vlastnosti zavádíme vztahy (5.12), jinak v triangulaci v rovině vznikají nežádou-

cí trojúhelníky. Na tuto vlastnost ještě upozorníme u konkrétního příkladu. *DT* je jednoznačná, pokud žádné čtyři body, nad kterými je triangulace sestrojována, neleží na kružnici.

Pro konstrukci *DT* existuje celá řada algoritmů. Uveďme například inkrementální konstrukci, která je založena na postupném přidávání bodů do již vytvořené *DT* a užití kritéria prázdné kružnice z definice 5.2. Ke konstrukci *DT* využíváme knihovní funkci, která je k dispozici v prostředí MATLAB.

Další vlastnosti a možnosti konstrukcí *DT* je možné nalézt v celé řadě publikací, jmenujme například (Edelsbrunner, 2001; Preparata a Shamos, 1985; de Berg *a kol.*, 2010).

Nyní máme v rovině (x', y') vytvořenou Delaunayovu triangulaci s vrcholy v půdorysech bodů $\left\{X_i^C\right\}_{i=1}^n$. Přejděme zpět do prostoru, tj. k vrcholům trojúhelníků v triangulaci přidáme třetí souřadnici tedy hodnotu r. V kartézské soustavě souřadnic S' získáváme triangulaci v prostoru. Takto vytvořenou povrchovou triangulaci budeme nazývat *výškovou mapou*. Při implementaci algoritmu reprezentujeme polygonální povrch v kartézské soustavě souřadnic S' pomocí ukazatelů do seznamu vrcholů, tj. seznamu bodů $\left\{X_i^C\right\}_{i=1}^n$. Každý trojúhelník polygonální sítě je tedy definován trojicí indexů, tj. ukazatelů do tohoto seznamu.

Posledním krokem navrženého algoritmu je převod bodů $\left\{X_i^C\right\}_{i=1}^n$ zpět do kartézských souřadnic spolu s vytvořenými trojúhelníky povrchové triangulace. Jelikož je každý trojúhelník definován trojicí indexů do seznamu bodů, stačí tyto ukazatele použít i pro body $\left\{X_i'\right\}_{i=1}^n$.

Je zřejmé, že výsledná povrchová triangulace není „uzavřená“, což je způsobeno převodem do cylindrických souřadnic. Plocha, na které je naměřené bodové mračno, je totiž transformací souřadnic v určitém místě rozpojena a body na jejím povrchu přetransformovány do nových souřadnic. Trojúhelníky v povrchové triangulaci, které by měly správně přecházet přes tento „šev“ nemohou být proto určeny. Pokud se jedná o bodové množiny získané vzorkováním pouze částí rotačních ploch, tj. pro jejich určení uvažujeme rotaci polomeridiánu o úhel menší než $360°$, není třeba tento problém řešit. Výsledná triangulace povrchu bude neuzavřená. Je ovšem nutné ohlídat vhodné natočení nové kartézské soustavy souřadnic $S' = \{O', e_1', e_2', e_3'\}$ kolem osy z', což zajistíme volbou jednotkových vektorů e_1' a e_2', tak aby byla plocha rozpojena v místech, kde nejsou navzorkované žádné body.

V případech, kdy uvažujeme rotaci polomeridiánu o úhel $360°$, je nutné provést spojení výsledné povrchové triangulace. To zajistíme tím, že při transformaci do cylindrických souřadnic zobrazíme znovu část bodů ve směru napojení. Potom po zpětné transformaci do kartézských souřadnic dostaneme povrchovou triangulaci, která se částečně překrývá. Překrývající se trojúhelníky jsou ale díky použití Delaunayovy triangulace shodné, neboť *DT* je jednoznačná, pokud žádné čtyři body, nad kterými je triangulace sestrojována, neleží na kružnici. V obecném případě toto platí. Odstranění překrývajících se trojúhelníků tedy není příliš problematické.

Poznamenejme, že díky převodu do cylindrických souřadnic a zobrazení těchto hodnot v soustavě S' zde můžeme nalézt souvislost se zpracováním digitálního modelu terénu. Pro modely terénů, kdy každému bodu mapy lze jednoznačně přiřadit výška, se v počítačové grafice používá označení 2,5D model (Bien, 2008; Zábranský, 2005). Terén je vyjadřován

jako funkce dvou proměnných, přičemž takto nelze popisovat speciální tvary terénu, jako jsou například různé převisy. V našem případě můžeme situaci v kartézské soustavě souřadnic S' po převodu do cylindrických souřadnic chápat právě jako výškovou mapu popsanou pouze diskrétně, tj. konečnou množinou bodů. Rovněž počítáme s určitým omezením, neboť uvažujeme pouze rotační plochy se speciálními polomeridiány, jak jsme v úvodu tohoto oddílu vysvětlili. Povrchová triangulace reprezentující plochu terénu, je potom nazývána jako 2,5D triangulace.

Ukažme si popsaný algoritmus na konkrétních příkladech.

Příklad 5.3: *TRIANGULACE POVRCHU POMOCÍ TRANSFORMACE.* Mějme dánu množinu $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^n$ v eukleidovském prostoru E_3, které popisují povrch rotační plochy. Vždy volíme záměrně řídkou množinu bodů kvůli přehlednosti.

V prvním případě se jedná o body, které jsou rozloženy pravidelně na části rotační válcové plochy, jde o stejnou množinu jako v příkladě 5.2. Předpokládejme, že jsme již našli osu symetrie této množiny některou z popsaných metod a uvažujme až situaci v kartézské soustavě souřadnic S'. Zobrazujeme tedy body $\{X_i'\}_{i=1}^n$ a kladné části poloos x', y', z'. Dále jsou body $\{X_i'\}_{i=1}^n$ převedeny do cylindrických souřadnic podle vzorců (5.11) a zobrazeny v soustavě S' jako body $\{X_i^C\}_{i=1}^n$. Tyto dvě situace zachycuje obrázek 5.34. Obrázek 5.35 ilustruje výsledek Delaunayovy triangulace provedenou nad půdorysy bodů $\{X_i^C\}_{i=1}^n$. Jelikož jsou body vstupní bodové ny $\mathcal{X}$ přesně na povrchu rotační válcové plochy, mají všechny body $\{X_i^C\}_{i=1}^n$ stejnou třetí souřadnici rovnající se poloměru rotační válcové plochy. Přidáním třetí souřadnice dostáváme proto posunutí všech bodů a trojúhelníků v triangulaci do výšky tohoto poloměru. Na obrázku 5.35 je znázorněn také zpětný přechod ke kartézským souřadnicím a výsledná triangulace povrchu. Kvůli přechodu k cylindrickým souřadnicím není výsledná triangulace uzavřená. Při transformaci do cylindrických souřadnic proto zobrazíme znovu část bodů ve směru napojení, jak velkou, je vyznačeno na obrázku 5.36. Vhodnou velikost zvolíme. Postup tvorby povrchové triangulace je zopakován. Překrývající se trojúhelníky ve výsledné povrchové triangulaci jsou dodatečně odstraněny. Výslednou triangulaci ilustruje obrázek 5.36.

Ve druhém případě uvažujeme množinu bodů, jež jsou pravidelně rozloženy na části povrchu obecnější rotační plochy, kterou jsme určili rotací polomeridiánu, ovšem rotace je provedena o úhel menší než $360°$. Jedná se o stejnou množinu bodů jako v příkladě 5.2. Opět předpokládáme, že jsme již nalezli osu rotační plochy, na které jsou body navzorkovány a uvažujeme až situaci v kartézské soustavě souřadnic S'. K převodu bodů $\{X_i'\}_{i=1}^n$ do cylindrických souřadnic užíváme tentokrát vzorců (5.12). Obrázek 5.37 znázorňuje vykreslené body $\{X_i'\}_{i=1}^n$ (a kladné části poloos x', y', z') a $\{X_i^C\}_{i=1}^n$ v soustavě S'. Na obrázku 5.38 je zachycen výsledek Delaunayovy triangulace nad půdorysy bodů $\{X_i^C\}_{i=1}^n$ a výšková mapa, která vznikne zpětným přidáním třetí souřadnice. Na obrázku 5.39 je znázorněna výsledná triangulace povrchu. Pro zajímavost doplňujeme také výsledek Delaunayovy triangulace nad půdorysy bodů $\{X_i^C\}_{i=1}^n$, kdy jsme k jejich převodu užili vzorců (5.11), a výslednou povrchovou triangulaci, která je zcela nevyhovující, viz obrázek 5.40. V triangu-

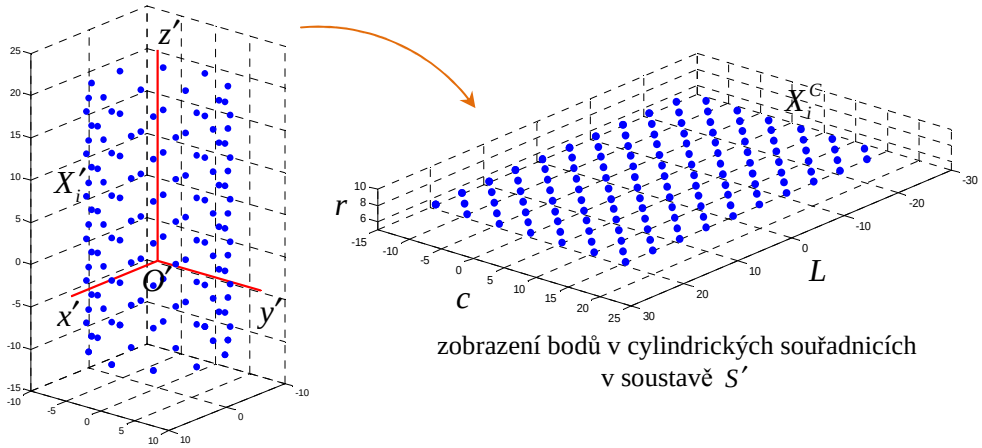
laci v rovině totiž vznikají nežádoucí trojúhelníky, kvůli vlastnosti *DT*, neboť její hranicí je konvexní obálka. To je právě důvod zavedení vztahů (5.12).

Ve výsledné triangulaci, kterou vidíme na obrázku 5.39, přesto narážíme na problém, který bude u složitějších množin ještě patrnější. Na okrajích výsledné triangulace dostáváme příliš úzké trojúhelníky. Opět je to způsobeno použitím *DT*. Tento problém řešíme opět zopakováním částí bodů na obou okrajích triangulace, jak je znázorněno na obrázku 5.41. Jak velká část bodů se zopakuje, je určeno experimentálně. Ve výsledné triangulaci však tyto trojúhelníky již neuvažujeme, viz obrázek 5.42.

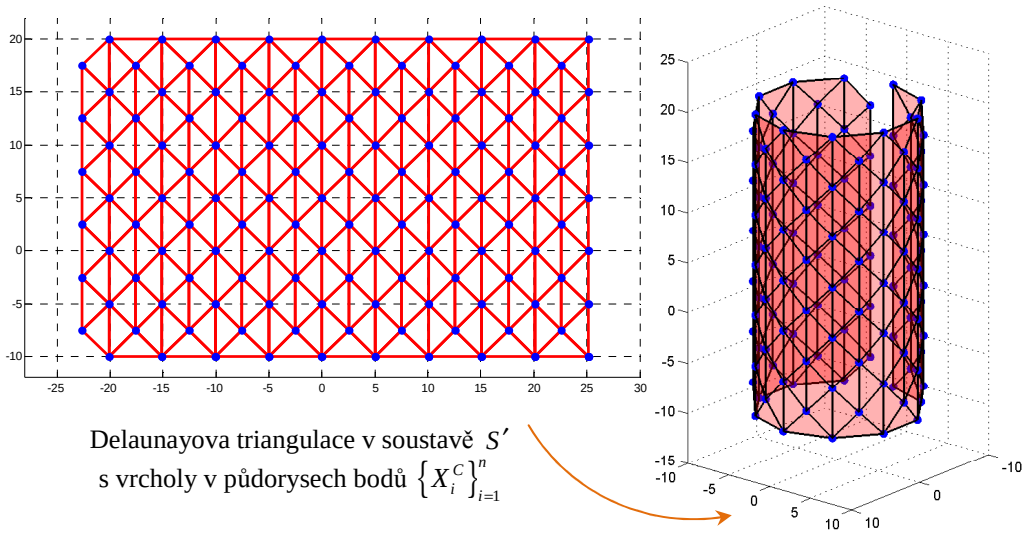
Poslední bodová množina, kterou v tomto příkladě zpracováváme, je již nepravidelná a je získaná z povrchu rotačního jednodílného hyperboloidu. Jedná se o body, jež jsou na povrchu rotačního hyperboloidu vygenerovány náhodně. Znovu předpokládáme, že již známe přibližnou osu rotační plochy, ze které byly body získány a uvažujeme až situaci v kartézské soustavě souřadnic S'. K převodu do cylindrických souřadnic používáme vztahů (5.12). Na obrázku 5.43 můžeme vidět vykreslené body $\{X_i'\}_{i=1}^n$ (a kladné části poloos x', y', z') a $\{X_i^C\}_{i=1}^n$ v soustavě S'. Na obrázku 5.44 je zakreslen výsledek Delaunayovy triangulace nad půdorysy bodů $\{X_i^C\}_{i=1}^n$ a výšková mapa, kterou dostaneme přidáním třetí souřadnice. Vidíme, že nedostáváme uspokojivý výsledek, neboť na okrajích výškové mapy vznikají velmi úzké trojúhelníky, ve směru napojení dokonce zcela nesprávné. Je nutné uvažovat navíc část bodů ve směru napojení jako v prvním případě kvůli uzavření výsledné povrchové triangulace. Tímto se však nevyhneme úzkým trojúhelníkům. Tento problém ošetříme tak, že při transformaci do cylindrických souřadnic zopakujeme na všech okrajích znovu část bodů, jak velkou, je vyznačeno na obrázku 5.45. Vhodnou velikost zvolíme. Dále zopakujeme postup tvorby výškové mapy. Překrývající se trojúhelníky ve výsledné povrchové triangulaci jsou dodatečně odstraněny. Rovněž jsou vynechány trojúhelníky, které vznikly přidáním okrajů. Výslednou povrchovou triangulaci této množiny znázorňuje obrázek 5.46.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

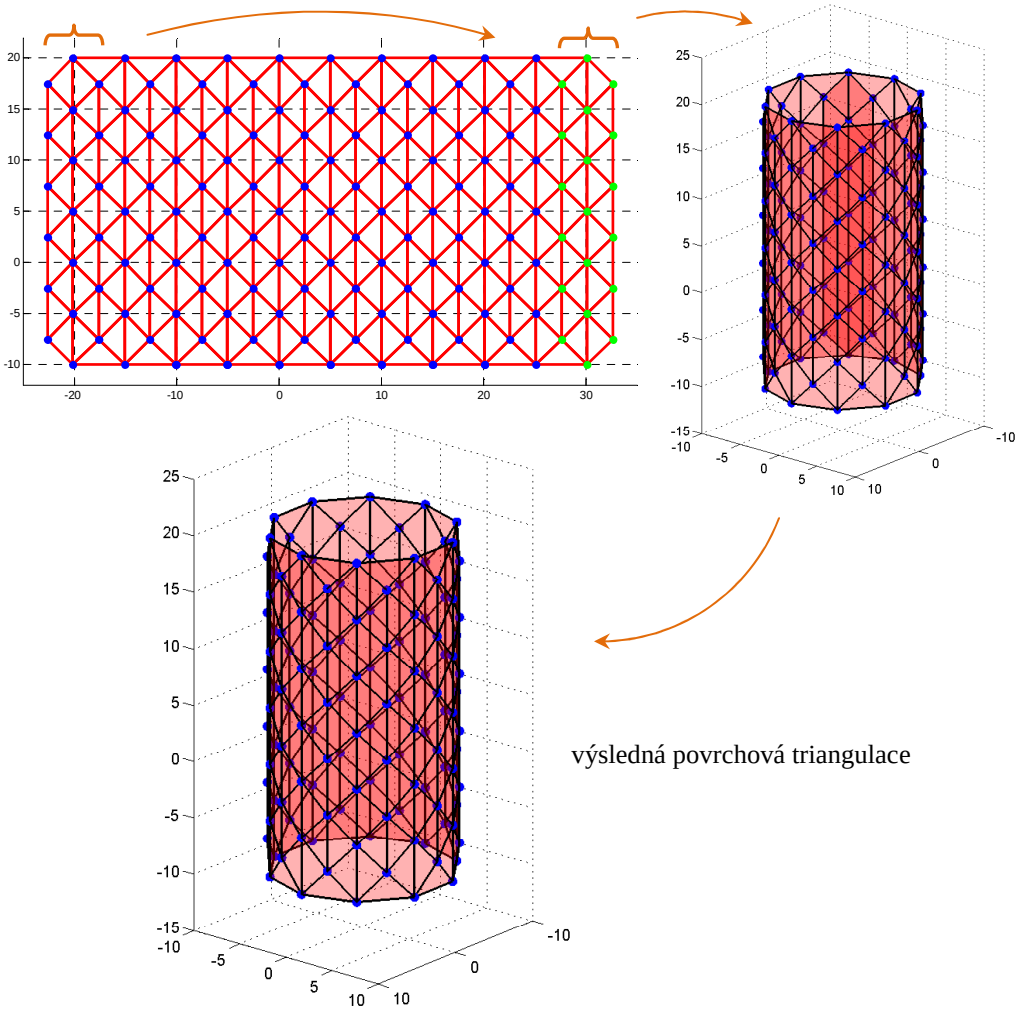
`cylindricke_souradnice.m` (programy/povrch/cylindr).



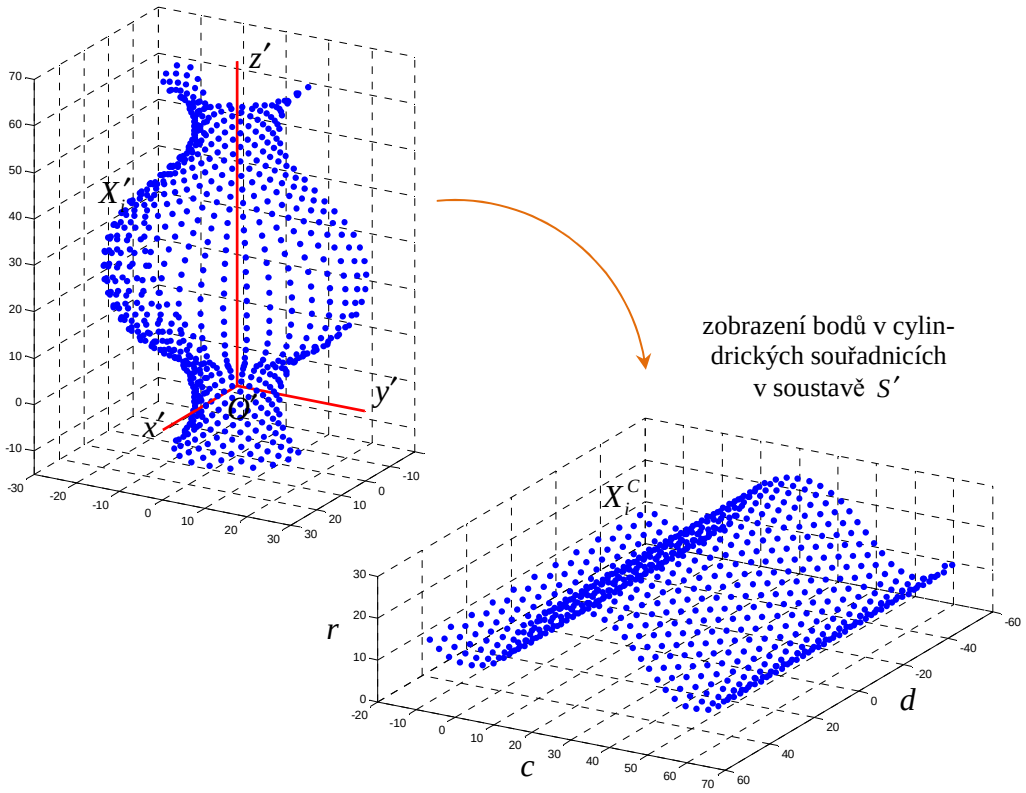
Obrázek 5.34: Body pravidelně rozmístěné na části rotační válcové plochy a převod do cylindrických souřadnic



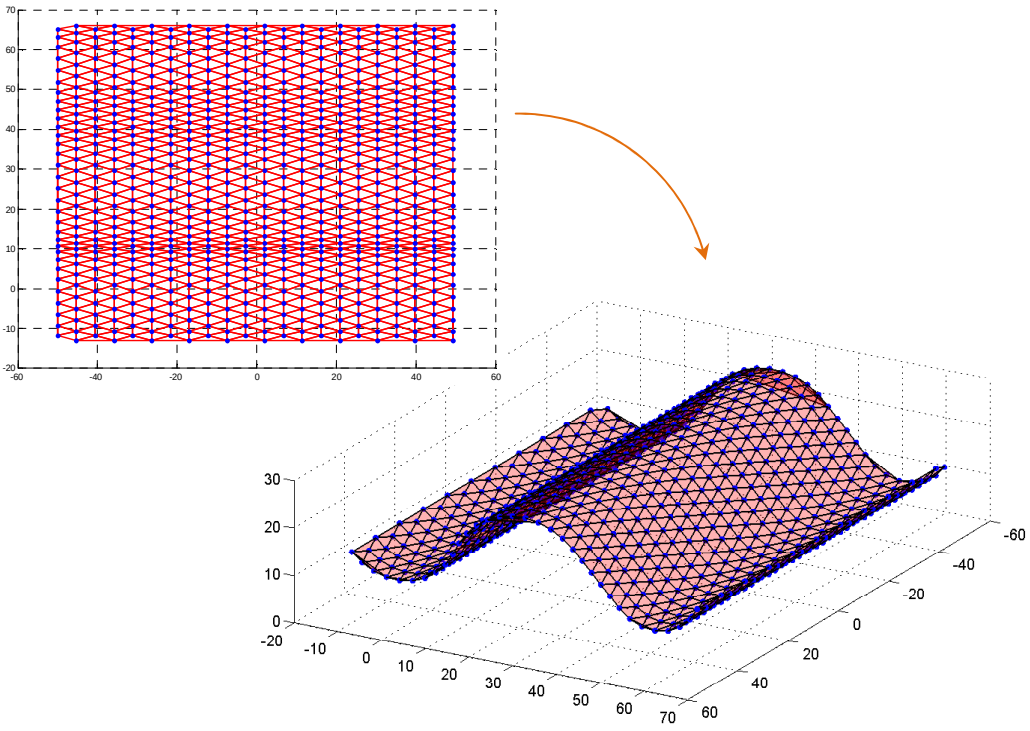
Obrázek 5.35: Delaunayova triangulace na množině půdorysů bodů a výsledná povrchová triangulace, která není uzavřená



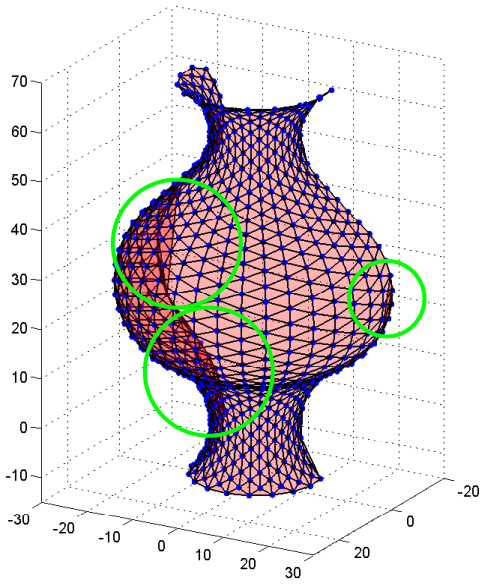
Obrázek 5.36: Delaunayova triangulace na množině půdorysů bodů, výsledná povrchová triangulace s překrývajícími se trojúhelníky a po odstranění nadbytečných trojúhelníků



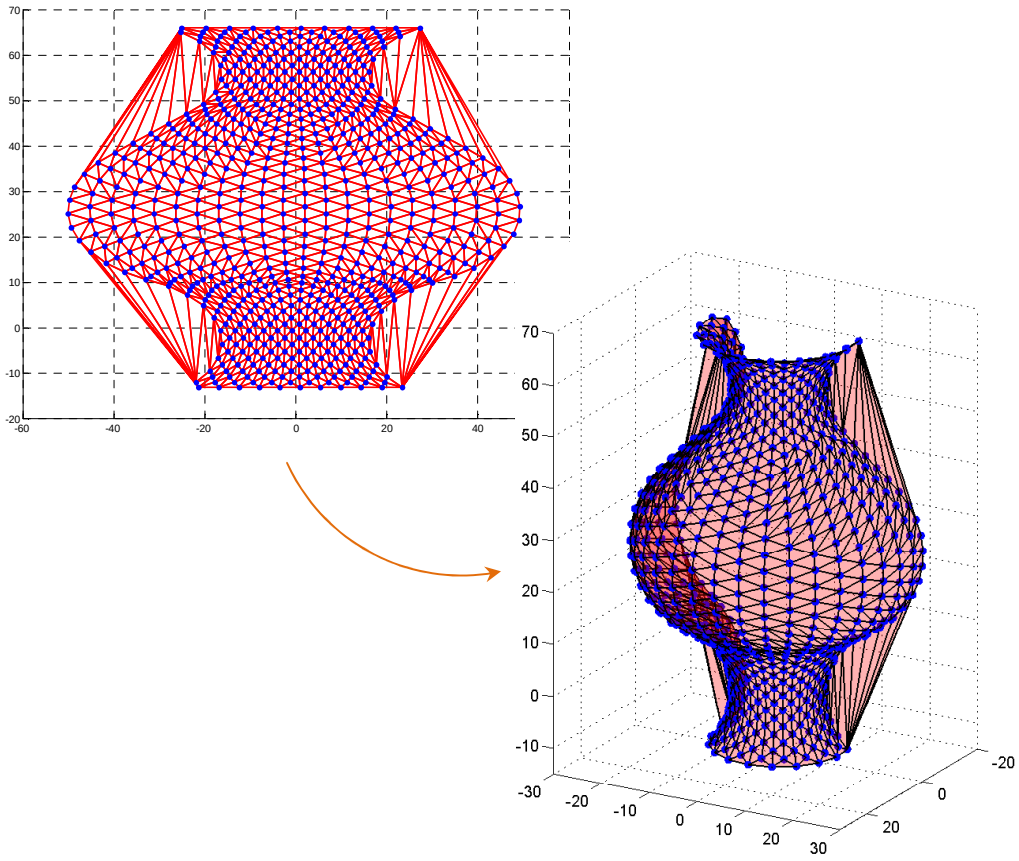
Obrázek 5.37: Body pravidelně rozmístěné na části rotační plochy a převod do cylindrických souřadnic



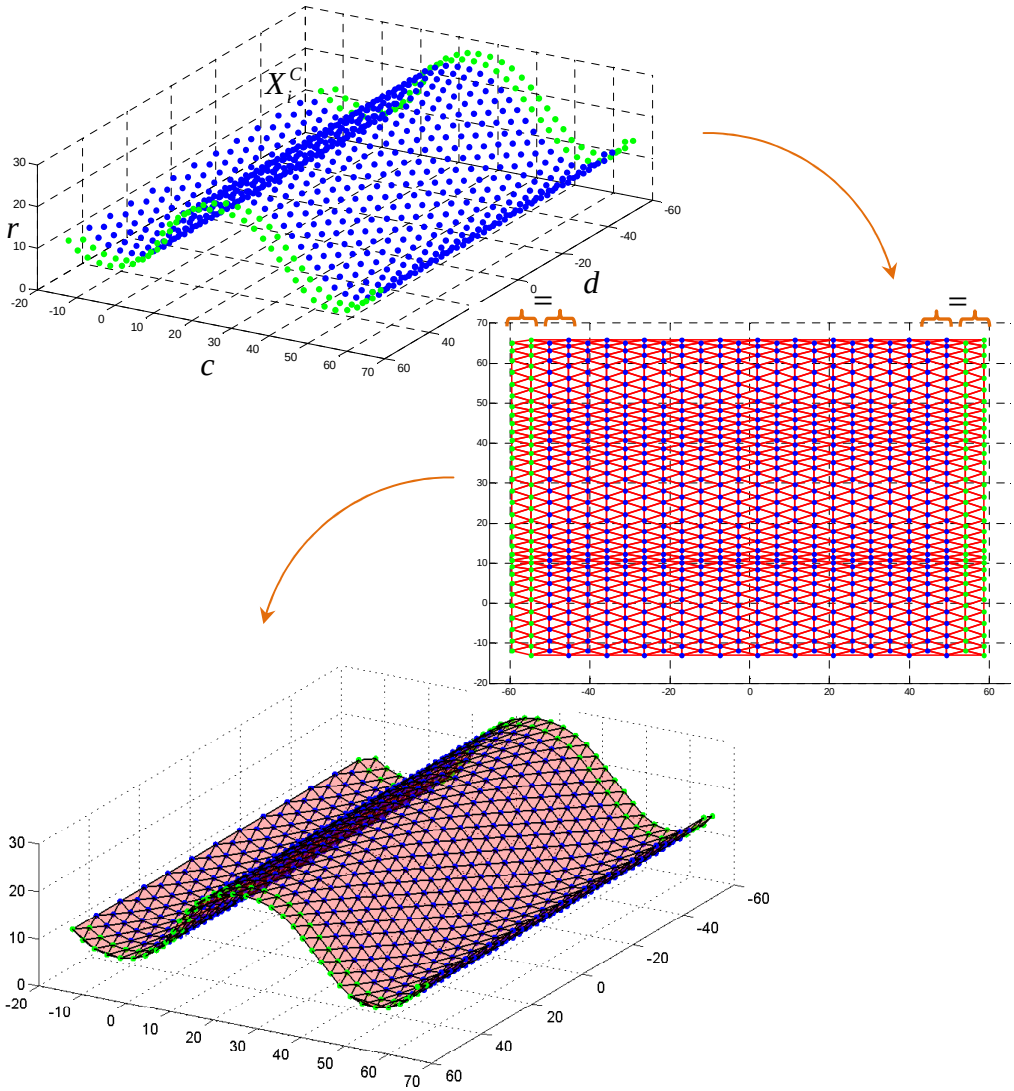
Obrázek 5.38: Delaunayova triangulace na množině půdorysů bodů a výšková mapa



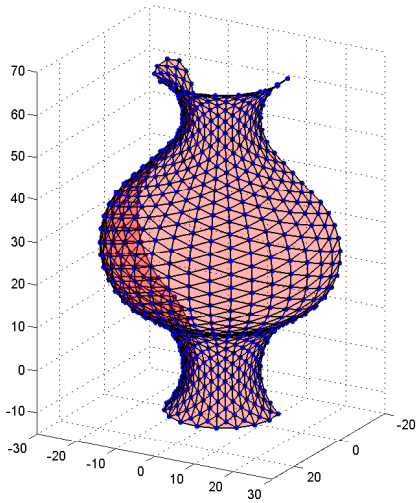
Obrázek 5.39: Výsledná povrchová triangulace s nežádoucími trojúhelníky



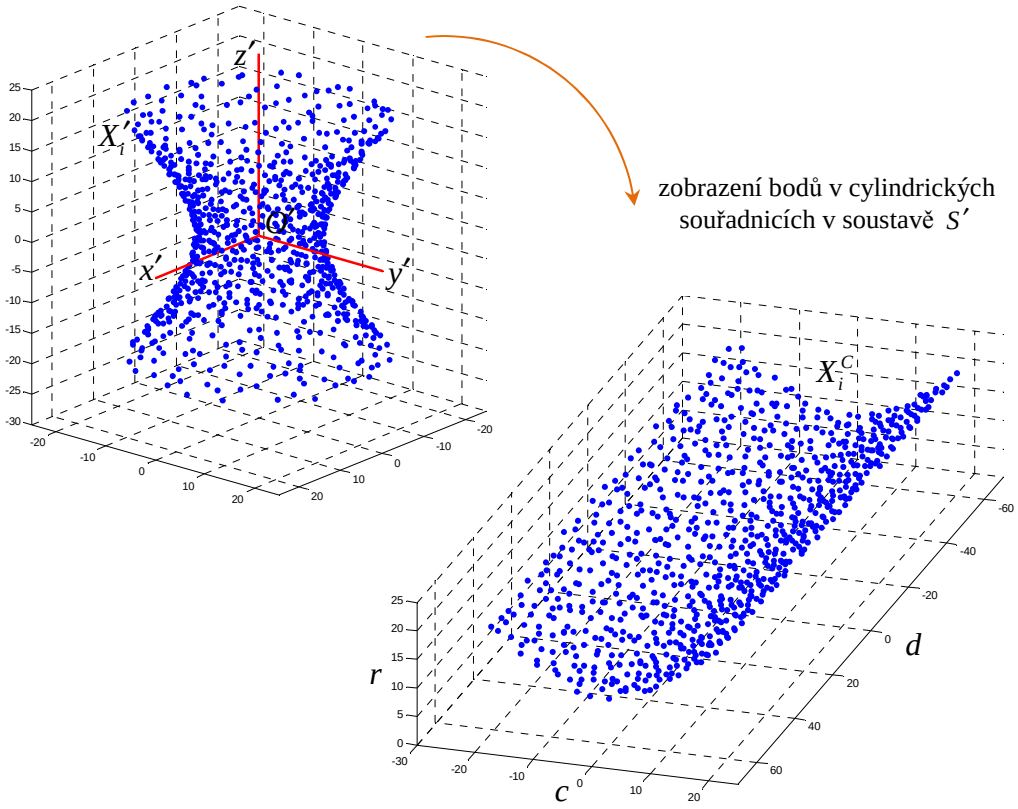
Obrázek 5.40: Delaunayova triangulace na množině půdorysů bodů a výsledná povrchová triangulace – nevhodné užití vztahů (5.11)



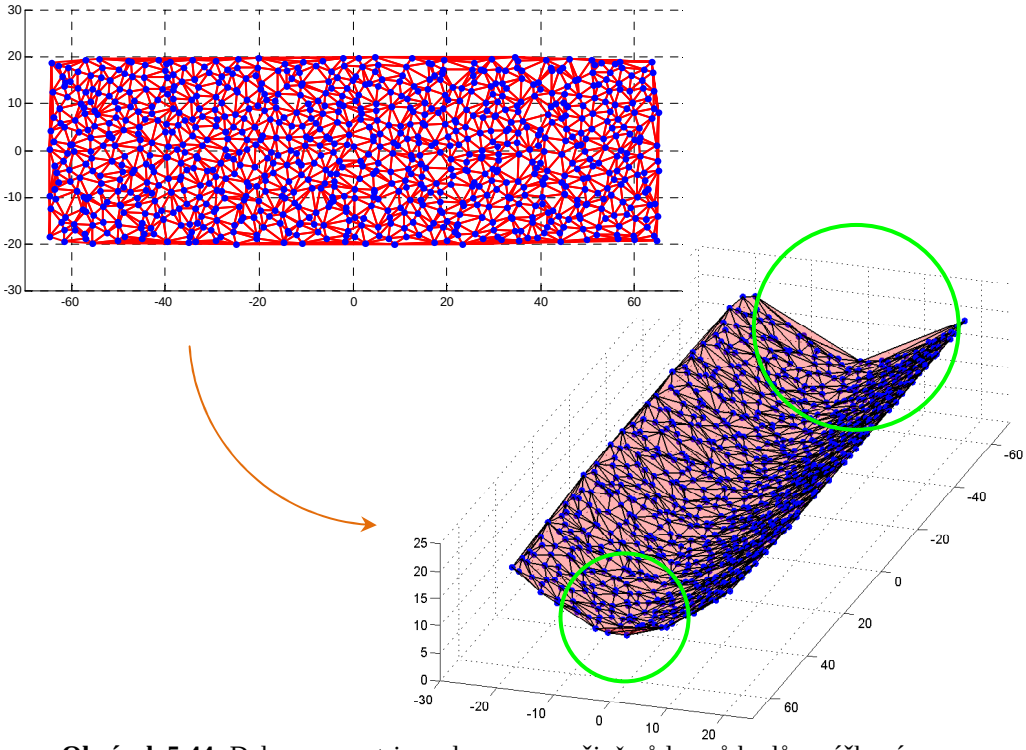
Obrázek 5.41: Delaunayova triangulace na množině půdorysů bodů a výšková mapa s opakujícími se body na okrajích



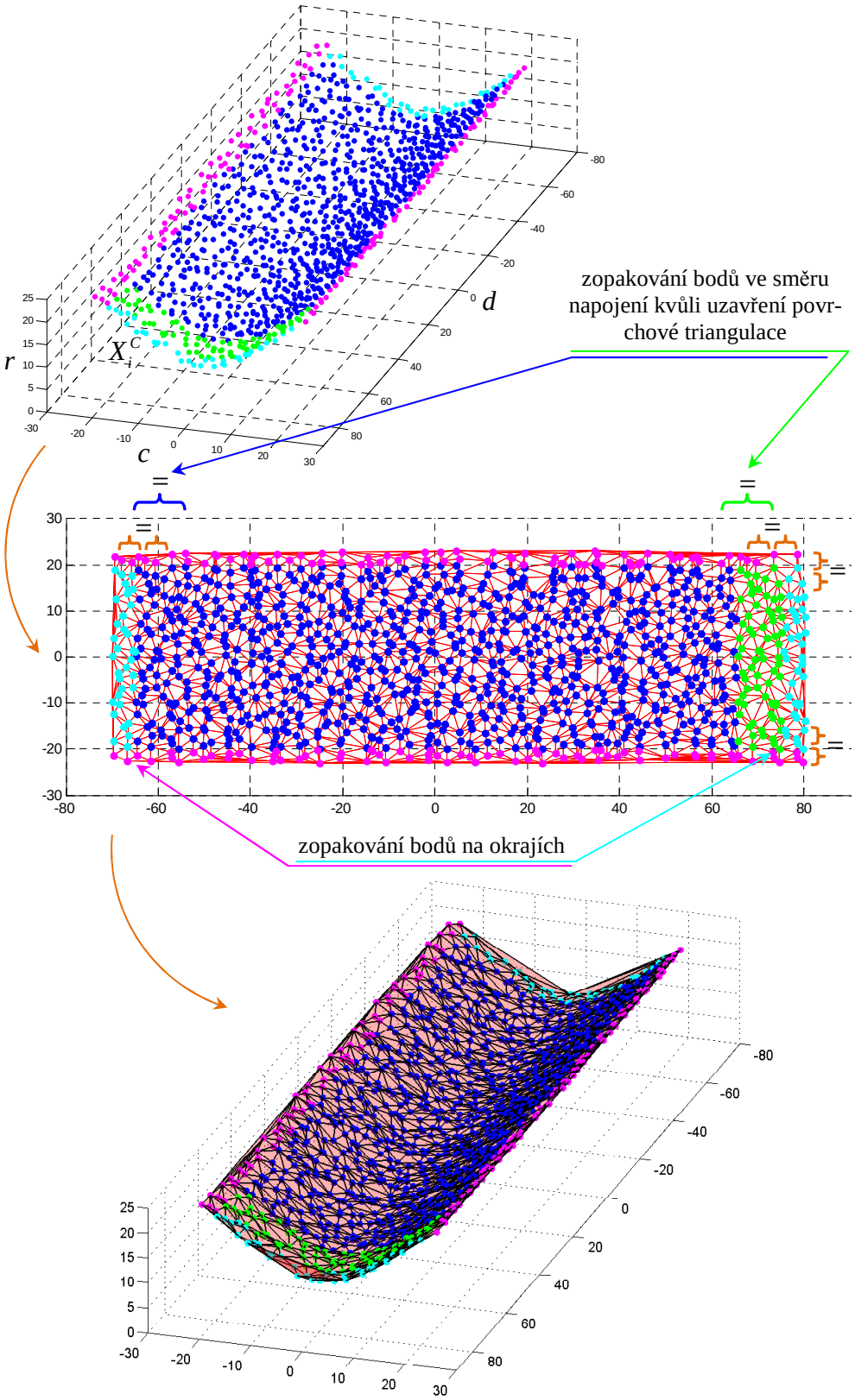
Obrázek 5.42: Výsledná povrchová triangulace bez chyb



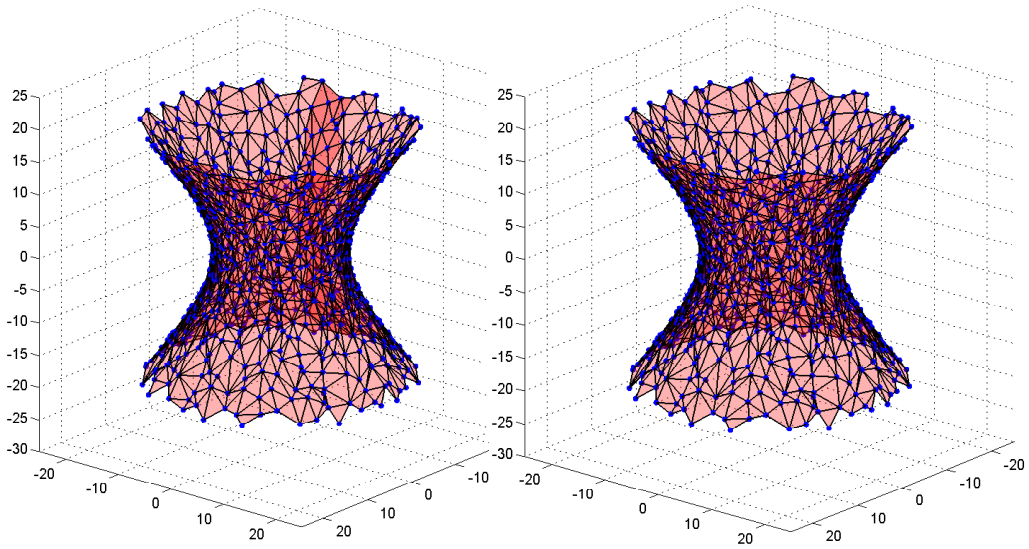
Obrázek 5.43: Náhodně vygenerovaná množina bodů na části povrchu rotačního jednodílného hyperboloidu a převod do cylindrických souřadnic



Obrázek 5.44: Delaunayova triangulace na množině půdorysů bodů a výšková mapa s nežádoucími trojúhelníky



Obrázek 5.45: Delaunayova triangulace na množině půdorysů bodů a výšková mapa s opakujícími se body na okrajích



Obrázek 5.46: Výsledná povrchová triangulace bez chyb před a po odstranění překrývajících se trojúhelníků

Další příklady konstrukce povrchové triangulace pomocí transformace do modifikovaných cylindrických souřadnic uvedeme později pro reálné bodové množiny, které obsahují velké množství bodů. Užívat budeme nejčastěji třetí verzi programu tj. v příkladě 5.3 užitou pro body získané z rotačního jednodílného hyperboloidu.

Důvod, proč jsme vyvinuli metodu založenou na rovinné triangulaci, je opodstatněný. Úlohu převádíme na triangulaci v dvojrozměrném případě, pro kterou existují výkonnější algoritmy než pro trojrozměrný případ, kde nastává mnoho nejednoznačných situací. I když narážíme na singulární a nejednoznačné případy na okrajích rovinné triangulace, které nastávají i po zpětném převodu do trojrozměrných souřadnic v povrchové triangulaci, jedná se stále o podstatně jednodušší situace. Okraje ploch lze vyřešit právě uvedeným prodloužením povrchu, případně zopakováním některých bodů tak, aby vznikl potřebný překryv, ze kterého se posléze doplní chybějící část triangulace do trojrozměrného prostoru. Rovněž lze postupovat tak, že příliš úzké a dlouhé trojúhelníky jsou dodatečně detekovány a odstraněny.

Na závěr uveďme, že ve dvojrozměrném případě v rovině (x', y') lze volit také jinou triangulační techniku ne nutně Delaunayovu triangulaci. Pro Delaunayovu triangulaci jsme se rozhodli kvůli vlastnostem, které splňuje, a snadnější implementaci.

5.4 Konstrukce povrchové triangulace pomocí rovinových symetrií

V posledním oddíle této kapitoly navrhujeme třetí možnost konstrukce polygonální sítě a to pomocí rovinových symetrií. Tuto metodu budeme užívat tehdy, jedná-li se o bodovou množinu reprezentující povrch, který je symetrický podle jedné roviny, dvou nebo tří navzájem kolmých rovin. Pokud je bodová množina rovinově symetrická nebo alespoň přibližně

rovinově symetrická lze pracovat pouze s částí povrchu a dále využívat rovinové symetrie. Nalezením rovinových symetrií bodové množiny lze bodové mračno orientovat, což plánujeme také dále využít k analytickému popisu povrchu. Abychom mohli navrhovaný algoritmus použít, musíme fakt, že je bodová množina symetrická podle nějakých rovin, znát. Jak již bylo řečeno v předchozích oddílech, u analýzy bodových množin odpovídajících reálným povrchům to předpokládáme na základě vlastností naskenovaného povrchu, které předem známe, či z vizuálního posouzení bodové množiny. Pokud je povrch reprezentovaný bodovou množinou symetrický podle více rovin, lze algoritmus dodatečně modifikovat.

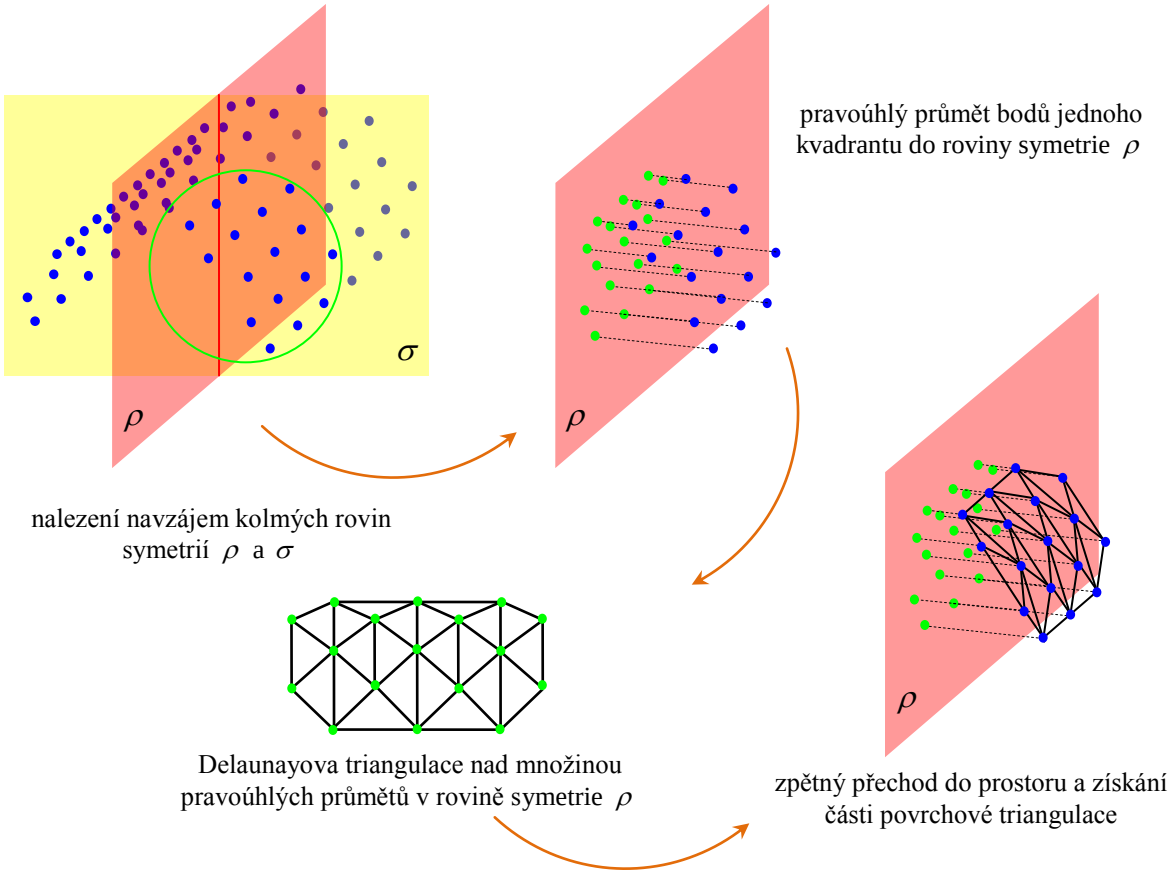
Nechť je dána neprázdná množina $\mathcal{X}$ *n* bodů v eukleidovském prostoru E_3 popisující povrch symetrický podle jedné roviny, dvou nebo tří navzájem kolmých rovin, tradičně značme tuto množinu $\mathcal{X} = \{X_i\}_{i=1}^{n}$. Body vstupní množiny $\mathcal{X}$ leží na nebo velmi blízko povrchu, tento předpoklad je stejný jako na začátku oddílu 5.1. Hledáme opět povrchovou triangulaci tak, že vrcholy trojúhelníků jsou všechny body množiny $\mathcal{X}$ a každá hrana v triangulaci je společná nejvýše dvěma trojúhelníkům.

V prvním kroku nově navrhovaného algoritmu určujeme rovinové symetrie bodové množiny pomocí metod předložených v oddíle 4.4 (*Hledání rovinových symetrií*).

Dále lze postupovat různě. Jedna z možností, jak tvořit výslednou povrchovou triangulaci, je pracovat s bodovou množinou po částech. To znamená, jedná-li se o množinu bodů, která je symetrická nebo přibližně symetrická podle jedné roviny, zpracováváme dál zvlášť části bodové množiny v jednom a druhém poloprostoru. Analogicky tomu bude pro množinu bodů, která je symetrická nebo přibližně symetrická podle dvou navzájem kolmých rovin. V takovém případě rozdělíme bodovou množinu do čtyř kvadrantů, které opět zkoumáme odděleně. U bodové množiny symetrické nebo přibližně symetrické podle tří navzájem kolmých rovin pracujeme dále s jednotlivými oktanty. Leží-li některý bod vstupní bodové množiny přímo v rovině symetrie, rozhodneme, do kterého poloprostoru, kvadrantu případně oktantu bude započítán (u reálných množin k tomuto nedochází).

Máme-li určené roviny symetrií a bodovou množinu rozdělenou na příslušné úseky, vezmeme jednu část množiny a dále užijeme podobného postupu jako v předchozím oddíle 5.3. Body této části pravoúhle promítneme do některé z rovin symetrií, vzhledem ke které je tato část bodů výškovou mapou (nyní reprezentována konečným počtem bodů) a provedeme triangulaci v rovině nad těmito pravoúhlými průměty. Vždy tedy předpokládáme, že části bodových množin v jednotlivých poloprostorech, kvadrantech či oktantech, jsou vzhledem k některé z rovin symetrie výškovými mapami. Výběr roviny, do které pravoúhle promítáme, provádíme na základě znalosti vlastností povrchu reprezentovaného množinou bodů. V rovině se dále rozhodujeme pro triangulaci Delaunayovu, což není nutnou podmínkou. Poté přejdeme zpět do prostoru a dostáváme tak povrchovou triangulaci části bodové množiny, kterou jsme uvažovali. Využíváme toho, že každý trojúhelník v rovinné triangulaci je definován trojicí indexů, tj. ukazatelů do seznamu bodů, nad kterými je triangulace sestrojena. Při zpětném přechodu do prostoru stačí tyto ukazatele použít i pro body vstupní množiny.

Takto konstruujeme povrchovou triangulaci i pro zbývající části vstupní bodové množiny. Princip tohoto možného přístupu je znázorněn na obrázku 5.47 pro případ bodové množiny symetrické podle dvou navzájem kolmých rovin ρ a σ.



Obrázek 5.47: Princip algoritmu konstrukce povrchové triangulace v jednom kvadrantu určeného dvěma navzájem kolmými rovinami symetrie

Použijeme-li tuto metodu ke konstrukci povrchové triangulace, je nutné ošetřit napojení jednotlivých částí sítě vytvořených v poloprostorech, kvadrantech či oktantech. Jak je zřejmé z obrázku 5.47, výsledná povrchová triangulace nebude spojená. Proto dále určujeme okraje sítí, podél nichž mají být dílčí sítě napojeny. Nalezení okrajů sítí není implementačně náročné. Posledním krokem algoritmu je tedy napojení sítí podél těchto okrajů, což řešíme inkrementální konstrukcí odvozenou v oddíle 5.2.

Druhou možnost hledání povrchové triangulace pomocí rovinových symetrií, navrhujeme pro bodové množiny symetrické nebo přibližně symetrické podle dvou navzájem kolmých rovin. Jelikož často zpracováváme bodové množiny, které reprezentují klenby či nějakou formu zastřešení, lze i zde najít souvislost s digitálními modely terénu. U těchto typů povrchů může být každému bodu jednoznačně přiřazena výška. Klenbu nebo nějakou stropní plochu tedy můžeme chápat jako výškovou mapu vzhledem k rovině, nad kterou je konstruo-

vána. Tuto rovinu budeme hledat právě pomocí rovinových symetrií, neboť u bodových množin, které zpracováváme, je na tyto roviny symetrií kolmá. Máme-li bodovou množinu symetrickou nebo přibližně symetrickou podle dvou navzájem kolmých rovin ρ a σ, promítáme pravoúhle množinu bodů do pomocné roviny α, která je kolmá k oběma rovinám symetrií a v prostoru umístěna libovolně. V této rovině opět provádíme Delaunayovu triangulaci nad pravoúhlými průměty bodů, poté přecházíme zpět do prostoru a dostáváme tak povrchovou triangulaci. K popisu trojúhelníků v triangulaci rovněž využíváme ukazatelů do seznamu bodů, nad kterými je triangulace v rovině sestrojena. Stejné ukazatele užíváme i pro body vstupní množiny.

Algoritmus lze jednoduše modifikovat také pro bodové množiny, které jsou symetrické podle více rovin se společnou průsečnicí. Často takové množiny popisují právě různé typy kleneb, které jsou opět vzhledem k rovině kolmé k rovinám symetrií výškovými mapami.

U obou přístupů můžeme kvůli použití Delaunayovy triangulace v rovině narazit na problém, že na okrajích triangulace v rovině a tedy i ve výsledné povrchové triangulaci vznikají příliš úzké a dlouhé trojúhelníky. Tento problém řešíme stejně jako při konstrukci povrchové triangulace pomocí transformace souřadnic bodů do cylindrických souřadnic. To znamená, že opakujeme části bodů na okrajích triangulace, ovšem ve výsledné triangulaci již tyto body ani trojúhelníky neuvažujeme. V případě, že vznikají dlouhé a úzké trojúhelníky, které do triangulace povrchu nemají patřit, odstraňujeme je tak, že vynecháváme ty trojúhelníky, jejichž hrany jsou delší než experimentálně určená mezní délka. Jelikož jsou body vstupní množiny i v případech reálných množin navzorkovány na povrchu ve všech místech stejně hustě, lze jednoduše zjistit průměrnou délku hran v triangulaci a trojúhelníky, jejichž hrany tuto hodnotu výrazně převyšují odstranit. Tyto nevhodné trojúhelníky můžeme detekovat jednak v Delaunayově triangulaci v rovině α nebo ve výsledné povrchové triangulaci.

Ukažme si použití navrhovaného algoritmu na typické bodové množině, která je přibližně symetrická podle dvou navzájem kolmých rovin.

Příklad 5.4: *TRIANGULACE POVRCHU POMOCÍ ROVINOVÝCH SYMETRIÍ.* Mějme dánu množinu $\mathcal{X}$ n bodů $\{[x_i, y_i, z_i]\}_{i=1}^{n}$ v eukleidovském prostoru E_3, které popisují povrch křížové klenby, jež je tvořena dvěma částmi rotačních válcových ploch se shodnými poloměry a navzájem kolmými osami. Původní pravidelná síť bodů povrchu je ve všech souřadnicových směrech zašuměna, tj. ke každé souřadnici je přičteno náhodné číslo s normálním rozdělením. Opět volíme záměrně řídkou množinu bodů kvůli přehlednosti. Vstupní bodová množina je znázorněna na obrázku 5.48.

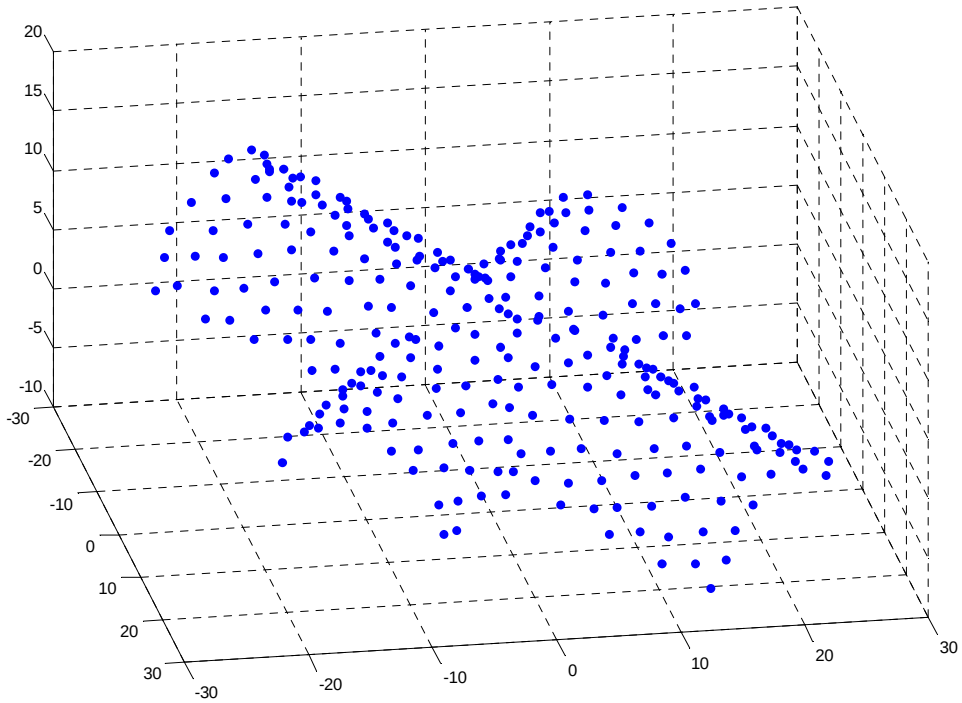
Sledujme obrázek 5.49. V prvním kroku algoritmu nalezneme přibližnou polohu navzájem kolmých rovin symetrií ρ a σ (v obrázku červeně) pomocí metod předložených v oddíle 4.4. Dále dourčíme rovinu α (v obrázku zeleně) kolmou k oběma rovinám symetrií, v prostoru ji umístíme libovolně. V dalším kroku algoritmu promítneme body pravoúhle do roviny α, viz obrázek 5.50, a přetransformujeme kartézskou soustavu souřadnic v prostoru tak, aby rovina α přešla do některé souřadnicové roviny, zde je zvolena rovina (x, y). Nad pravoúhlými průměty (nyní přetrans-

formované do roviny (x, y)) provedeme Delaunayovu triangulaci, viz obrázek 5.50. Poté užijeme zpětnou transformaci kartézské soustavy souřadnic v prostoru a dostáváme Delaunayovu triangulaci v rovině α. Na závěr přejdeme zpět k bodům vstupní množiny a získáváme povrchovou triangulaci, viz obrázek 5.51. Jak již bylo řečeno, využíváme toho, že každý trojúhelník v rovinné triangulaci je definován trojicí indexů, tj. ukazatelů do seznamu bodů, nad kterými je triangulace sestrojena. Při zpětném přechodu do prostoru stačí tyto ukazatele použít i pro body vstupní množiny.

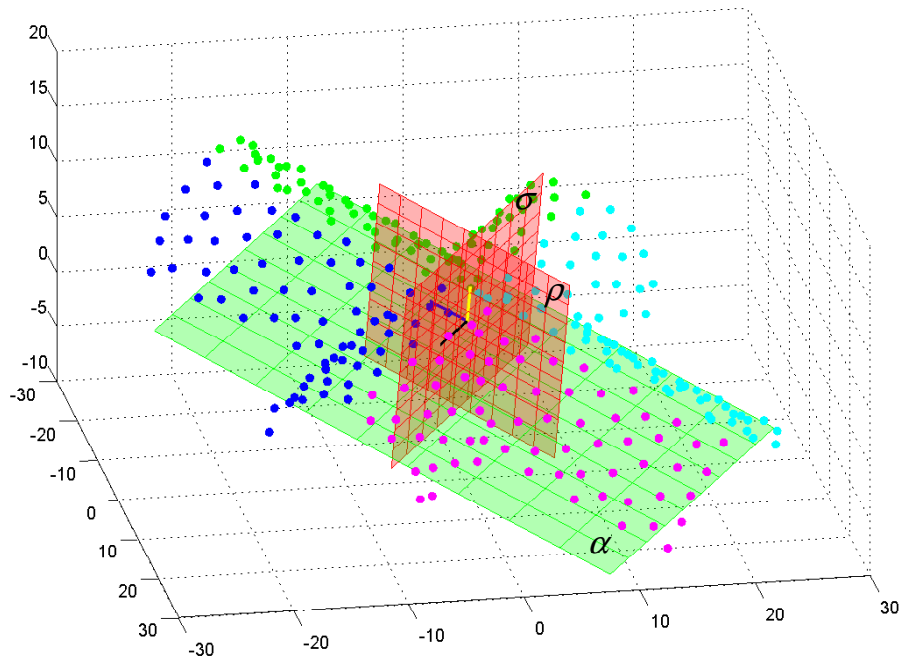
Jak vidíme na obrázku 5.51, není výsledná povrchová triangulace uspokojivá. Kvůli použité Delaunayově triangulaci vznikají v síti nevhodné trojúhelníky. Tyto trojúhelníky v povrchové triangulaci snadno detekujeme a odstraníme je. Postupujeme tak, že v triangulaci v rovině α vynecháváme ty trojúhelníky, jejichž hrany jsou delší než experimentálně určená mezní délka. Zde je to hodnota 4. Na obrázku 5.52 je znázorněna výsledná povrchová triangulace po odstranění nevhodných trojúhelníků.

Program z výpočetního prostředí MATLAB (popis cesty k programu na přiloženém výměnném médiu):

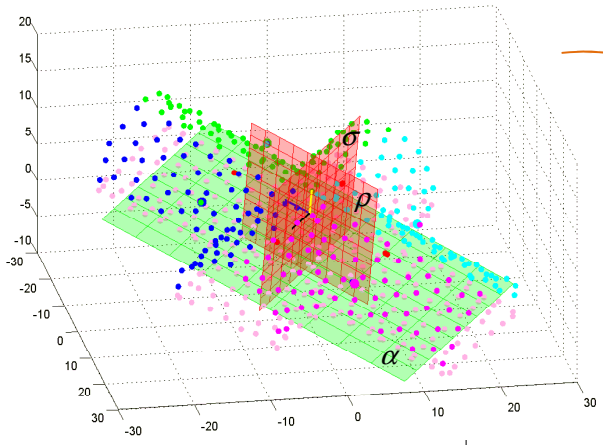
`triangulace_symetrie.m` (programy/povrch/symetrie).



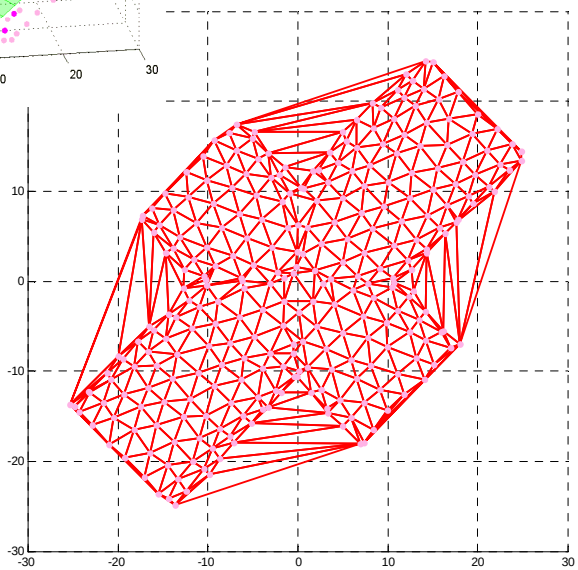
Obrázek 5.48: Zašuměná množina bodů získaná z povrchu tvořeného dvěma částmi rotačních válcových ploch se shodnými poloměry a navzájem kolmými osami



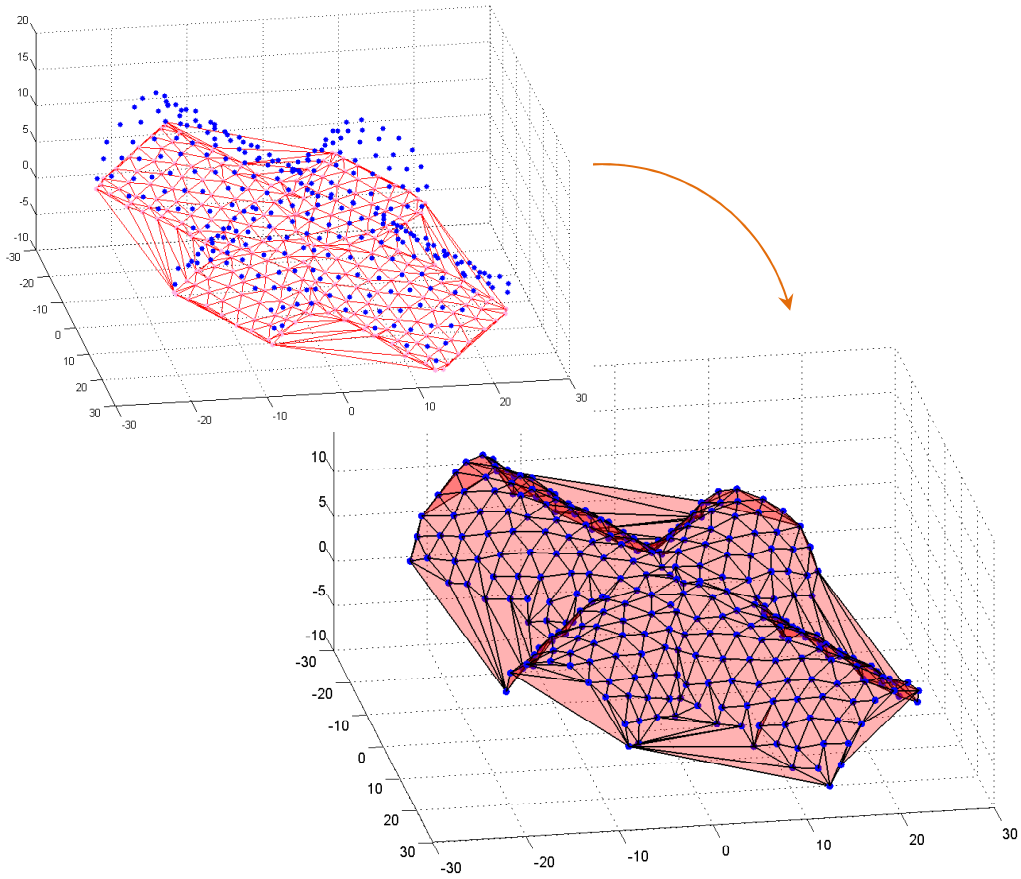
Obrázek 5.49: Dvě navzájem kolmé roviny symetrií ρ a σ a třetí na ně kolmá pomocná rovina α



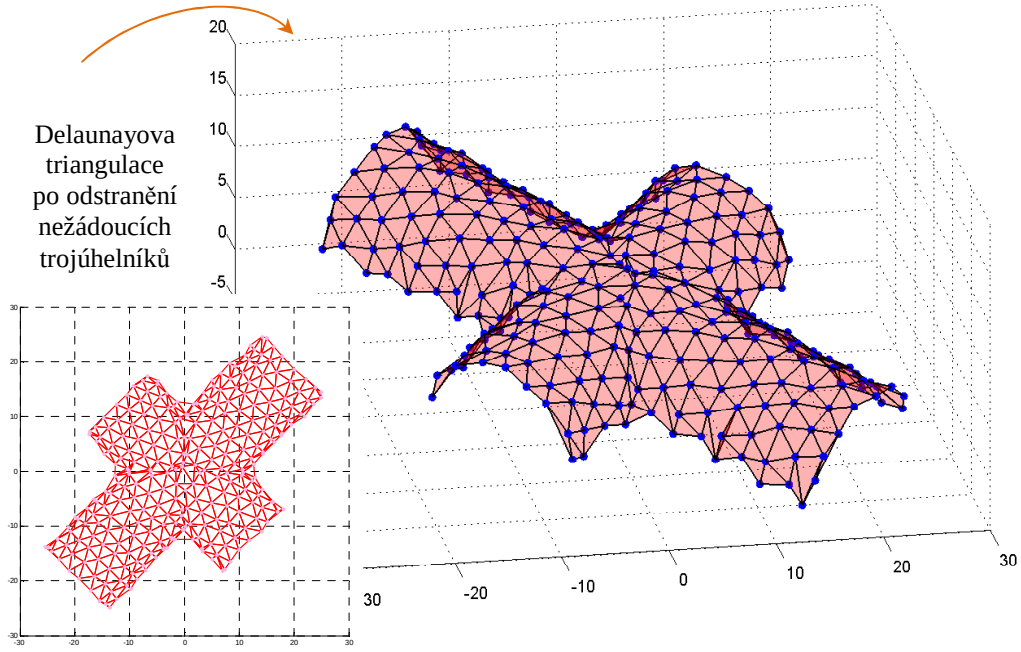
transformace kartézské soustavy souřadnic v prostoru a Delaunayova triangulace nad pravoúhlými průměty bodů do roviny α (nyní v rovině (x, y))



Obrázek 5.50: Delaunayova triangulace nad pravoúhlými průměty bodů v přetransformované rovině α



Obrázek 5.51: Delaunayova triangulace v rovině α a zpětný přechod k bodům vstupní množiny – povrchová triangulace s chybami



Obrázek 5.52: Výsledná povrchová triangulace po odstranění nevhodných trojúhelníků

Další příklady konstrukce povrchové triangulace pomocí hledání rovinových symetrií ukážeme později pro reálné bodové množiny, které obsahují velké množství bodů.

Jako motivaci pro budoucí práci ještě uveďme další důvody konstrukce povrchové triangulace metodami založenými na rovinových symetriích. Chceme se věnovat analytické deskripci povrchu reprezentovaného bodovým mračnem. Pro tyto účely vyvíjíme metody evolucí hladkých povrchů, pro které bude orientování bodového mračna nalezením jeho rovinových symetrií výchozím bodem.

Kapitola 6

Experimenty s reálnými daty

Veškeré navržené a popsané řešící postupy budou nyní ověřeny na bodových množinách reprezentujících skutečné povrchy. Bodová mračna prezentovaná v této kapitole jsme získali skenováním reálných objektů několika různými skenovacími zařízeními. U každé bodové množiny, která reprezentuje nějaký povrch, vždy uvedeme, jakým typem 3D skeneru byla získána. Jelikož jsme měli přístup ke speciálním typům skenovacích zařízení určených především ke skenování objektů malých rozměrů, vybírali jsme ke skenování geometrické modely reálných objektů o velikostech v rozmezí od 10 cm do 50 cm. Jednalo se o geometrické modely rotačních ploch, jejich částí a různých kombinací a modely kleneb. I když tyto bodové množiny reprezentují povrchy menších rozměrů, obsahují i tak velké množství bodů (desetitisíce až statisíce). Jejich zpracování je tedy s použitím dostupné výpočetní techniky velice časově náročné. Navíc zobrazení takových bodových množin je kvůli hustotě bodů téměř nemožné. Bodové množiny proto ztenčujeme nebo zobrazujeme v detailech pouze některé jejich části. Jaký způsob zobrazení jsme zvolili, vždy uvedeme u konkrétního příkladu.

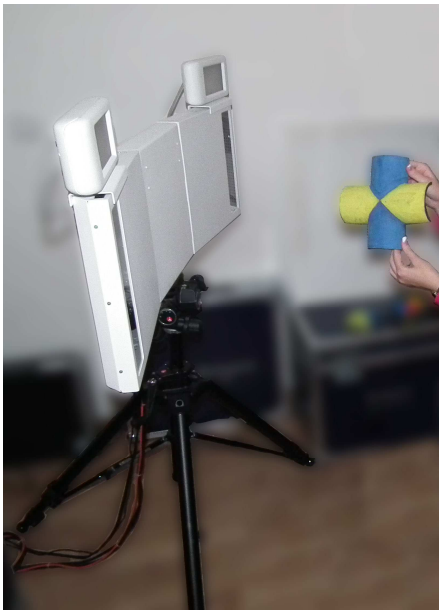
Samotné skenování geometrických modelů představovalo velice zdlouhavý a náročný proces, jednak kvůli nesnadnému přístupu ke skenovacím zařízením, a jednak kvůli osvojení si správného způsobu zacházení s těmito skenery. Někdy bylo nutné skenování některých objektů opakovat kvůli chybným měřením.

Další bodové množiny, jejichž zpracování jsme se věnovali, reprezentují již reálné objekty. Jedná se o rozsáhlá bodová mračna, která obsahují až miliony bodů. Opět je tedy problematické množiny zobrazovat a velice časově náročné je zpracovat. Výsledky rekonstrukce takové bodové množiny prezentujeme na jednom příkladě naskenovaného Vladislavského sálu. Bodové množiny reprezentující povrchy reálných staveb jsme získali od Stavební fakulty ČVUT.

6.1 Reálná data reprezentující povrchy geometrických modelů

V tomto oddíle se zaměříme na zpracování bodových množin, které jsme získali skenováním geometrických modelů menších rozměrů. K dispozici jsme měli tři typy skenovacích zařízení – dotykový skener MicroScribe G2X (obrázek 6.1), optický systém Vectra3D (obrázek 6.2 vlevo) a laserový skener Roland Picza LPX-1200 (obrázek 6.2 vpravo). Skenovací zařízení nám byla zapůjčena na pracovišti Laboratoř 3D zobrazovacích a analytických metod, Katedra antropologie a genetiky člověka, Přírodovědecká fakulta Univerzity Karlovy v Praze (Laboratoř 3D zobrazovacích a analytických metod, 2013).

Obrázek 6.1: Dotykový skener MicroScribe G2X – zapůjčený na pracovišti Laboratoř 3D zobrazovacích a analytických metod, Katedra antropologie a genetiky člověka, Přírodovědecká fakulta UK v Praze



Obrázek 6.2: Optický systém Vectra3D (vlevo) a laserový skener Roland Picza LPX-1200 (vpravo) – zapůjčené na pracovišti Laboratoř 3D zobrazovacích a analytických metod, Katedra antropologie a genetiky člověka, Přírodovědecká fakulta UK v Praze

6.1.1 Model balustrády – optický systém Vectra3D

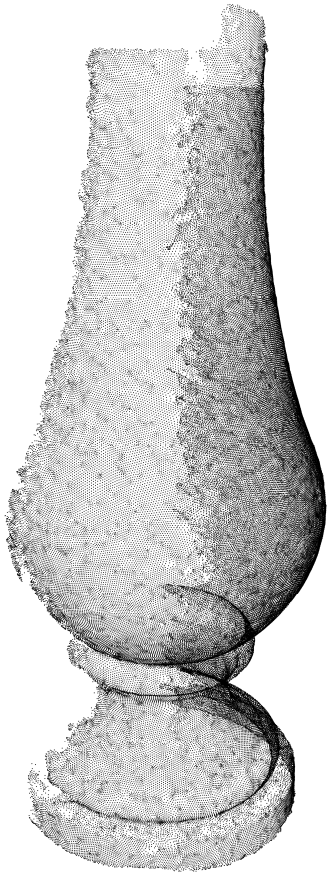
První bodová množina vznikla skenováním modelu balustrády, viz obrázek 6.3, optickým systémem Vectra3D. Balustráda je tvořena elementárními souosými rotačními plochami, tj. polomeridián plochy je sestaven z kruhových oblouků a úseček. Na obrázku 6.4 je zobrazeno toto mračno bez jakýchkoliv úprav, množina obsahuje 96 263 bodů. Je zřejmé, že při jednom skenování mohl optický skener zachytit pouze část skenovaného objektu. Vycházíme ze znalosti, že bodová množina reprezentuje povrch osově symetrický. Pracovat budeme pouze s naskenovanou částí a nebudeme vyžadovat registraci více bodových mračen. Registraci řeší komerční softwary, které bývají dodávány k danému typu skenovacího zařízení.

Obrázek 6.3: Model balustrády tvořené souosými rotačními plochami

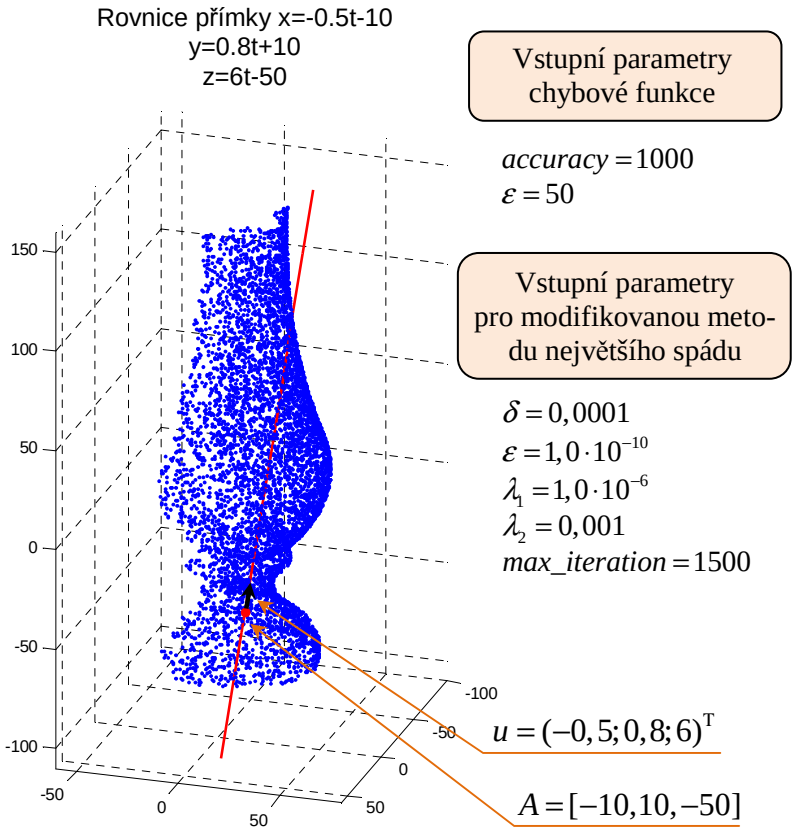
Osu symetrie bodové množiny hledáme v tomto případě naším postupem navrženým v oddíle 4.3 (*Hledání osy obecné rotační plochy*), přičemž volíme modifikovanou metodu největšího spádu.

Na obrázku 6.5 je společně s bodovým mračnem zakreslena počáteční poloha přímky, kterou postupně optimalizujeme, a předloženy jsou její parametrické rovnice. Obrázek 6.6 ilustruje postupné vylepšování polohy přímky, při-

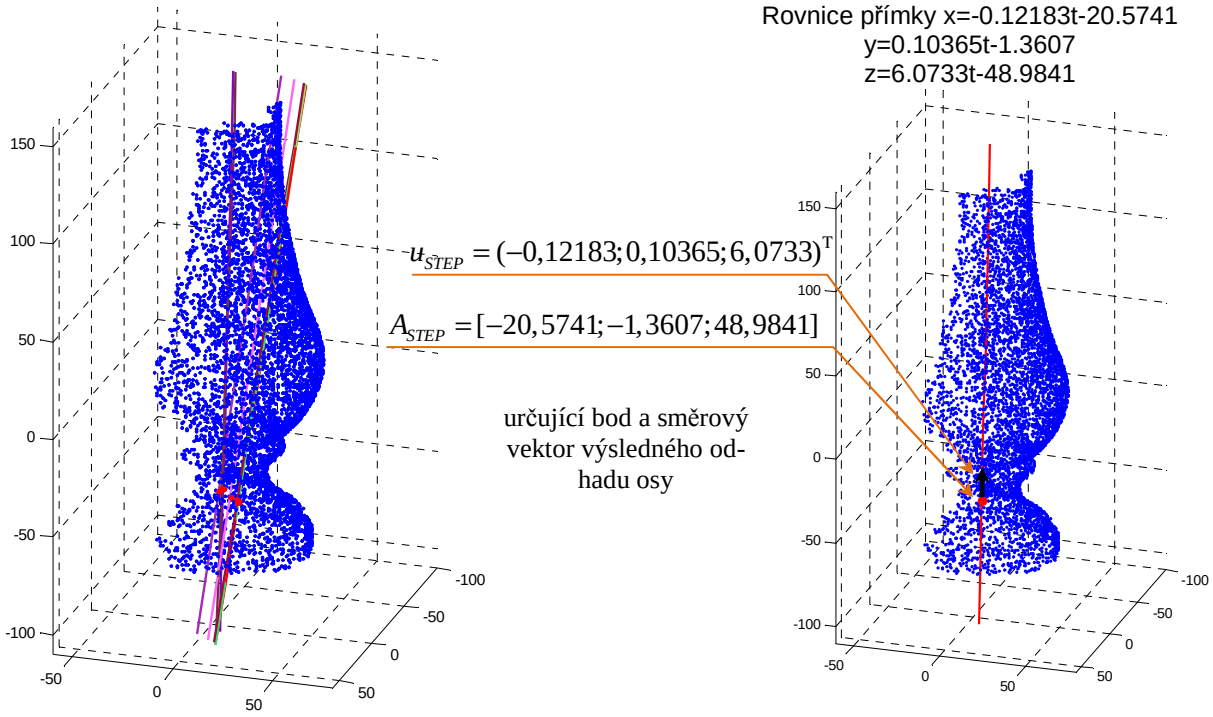
čemž kvůli přehlednosti se přímka vykresluje pouze v několika pozicích. Výpočty probíhají se všemi body vstupní množiny, ovšem ve všech obrázcích zobrazujeme pouze ztenčenou množinu bodů (zde je použita šestnáctina) kvůli větší přehlednosti. Zvlášť je vykreslena výsledná poloha osy a vypsány jsou její parametrické rovnice. Na obrázku 6.7 je zobrazena situace v pomocné soustavě souřadnic v rovině v prvním a posledním kroku iteračního procesu. Dostatečně dobrý odhad osy rotační plochy dostáváme pro uvedenou volbu vstupních parametrů metody (1500 iterací). Pro optimalizaci směrového vektoru a určujícího bodu osy volíme rozdílnou délku kroku, neboť u takto rozsáhlé množiny se již výrazně projeví rozdílnost spádu těchto parametrů. Hodnota λ_1 je délka kroku v optimalizaci směrového vektoru a hodnota λ_2 v optimalizaci určujícího bodu přímky. Pro přesnější výsledek optimalizace je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroků λ_1 nebo λ_2.



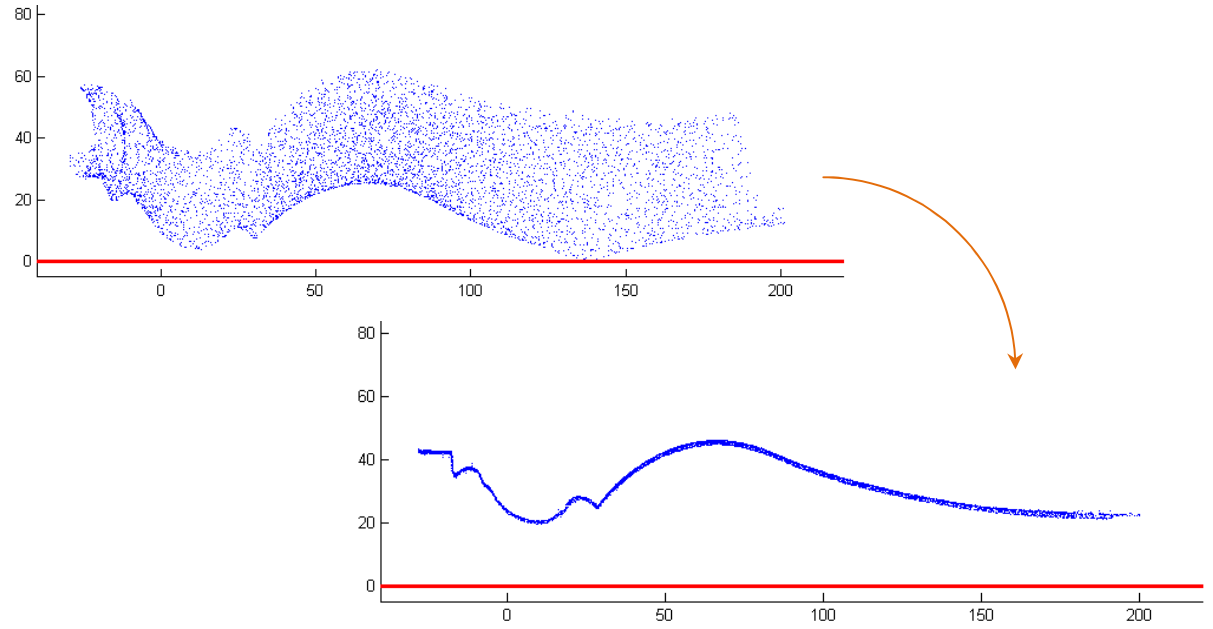
Obrázek 6.4: Bodová množina bez úprav reprezentující část povrchu balustrády tvořené souosými rotačními plochami – výstup z optického systému Vectra3D



Obrázek 6.5: Počáteční poloha přímky a vstupní parametry modifikované metody největšího spádu a chybové funkce



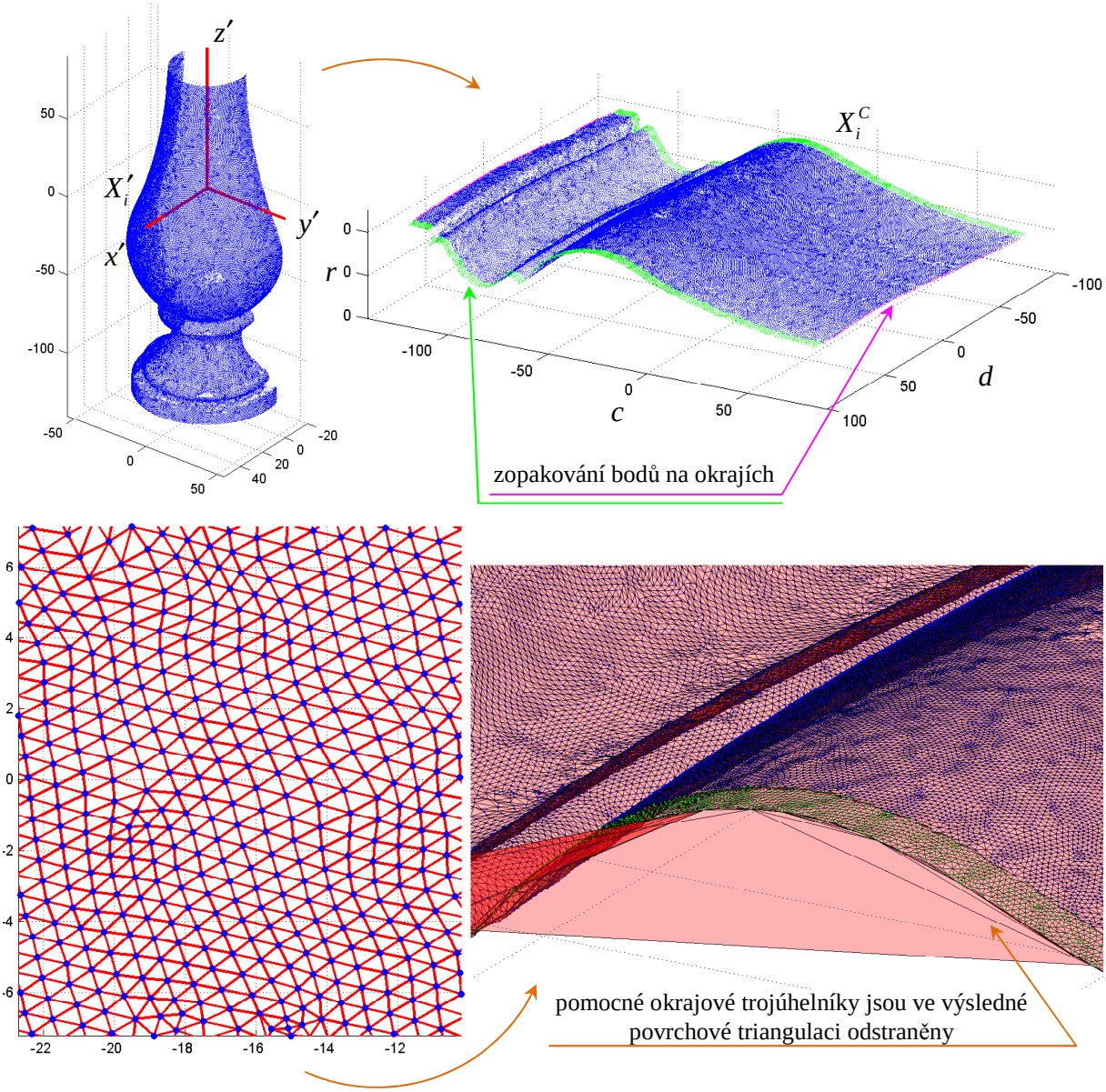
Obrázek 6.6: Postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu a výsledný odhad osy



Obrázek 6.7: Situace v pomocné soustavě souřadnic v rovině pro počáteční polohu optimalizované přímky a pro výsledný odhad osy

Nalezenou osu nyní použijeme k určení povrchové triangulace, bodovou množinu však předtím mírně upravíme. Jak je vidět na obrázku 6.4, není bodová množina v okrajových

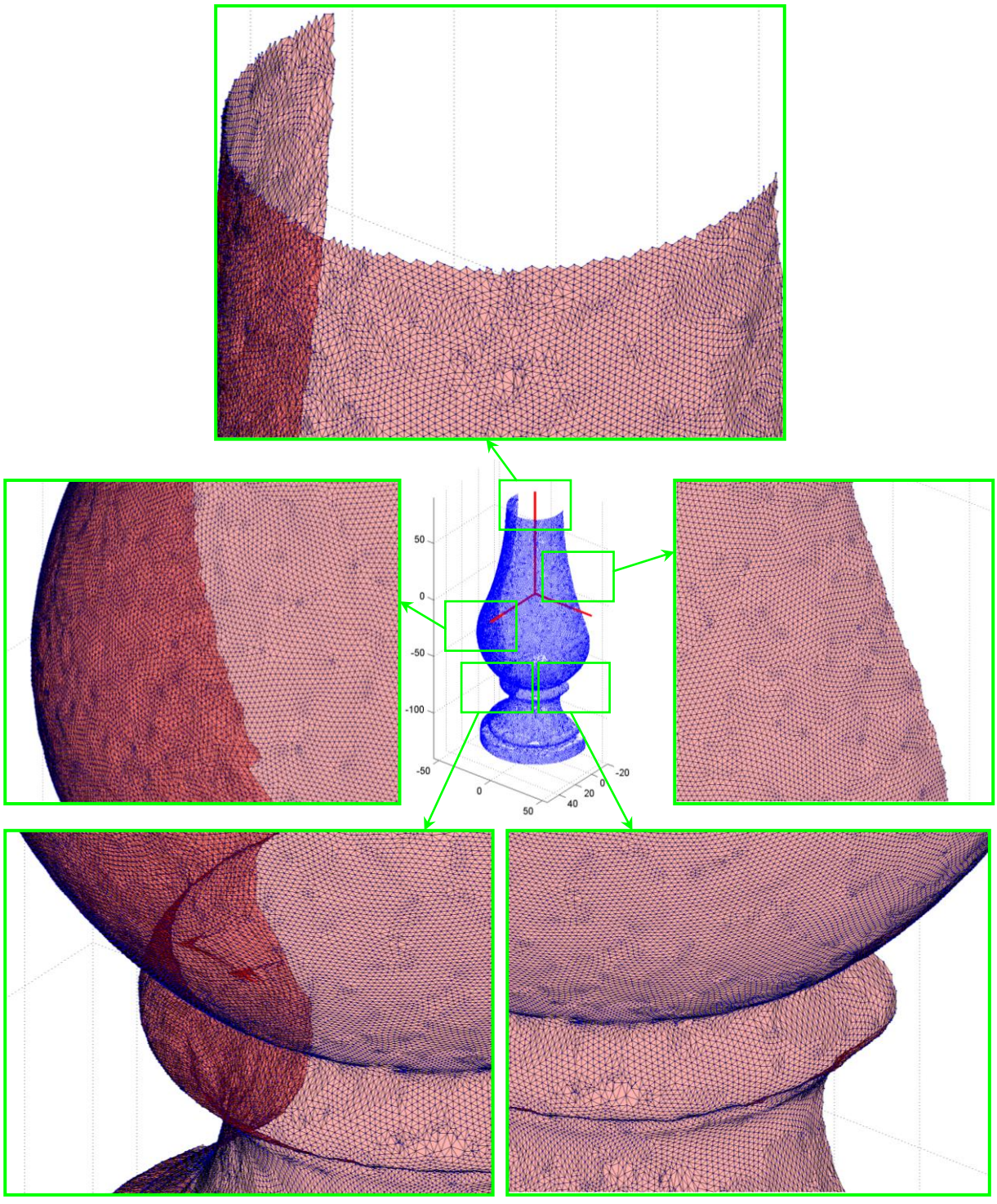
částech naskenována rovnoměrně jako v místech, která byla optickým skenerem zachytitelná. Tyto části ručně odstraníme ve 3D modelovacím softwaru Rhinoceros.



Obrázek 6.8: Transformace kartézské soustavy souřadnic v prostoru, převod do cylindrických souřadnic, Delaunayova triangulace na množině půdorysů bodů (detail) a výšková mapa (detail) – s opakujícími se body na okrajích

Hledání povrchové triangulace takto upravené bodové množiny provádíme metodou převodu do cylindrických souřadnic navrženou v oddíle 5.3 (*Triangulace povrchu pomocí transformace*). K převodu bodů do cylindrických souřadnic užíváme vzorců (5.12) předložených v oddíle 5.3. Obrázek 6.8 znázorňuje vykreslené body $\{X_i'\}_{i=1}^n$ (a kladné části poloos x', y', z') a body $\{X_i^C\}_{i=1}^n$ v soustavě S', nyní množinu při zobrazení nijak neztenčujeme. Pro lepší výsledek povrchové triangulace přidáváme zopakování částí bodů na všech okrajích

výškové mapy, jak je barevně zvýrazněno. Na obrázku 6.8 je v detailu rovněž zachycen výsledek Delaunayovy triangulace nad půdorysy bodů $\left\{X_i^C\right\}_{i=1}^n$ a v detailu výšková mapa, která vznikne zpětným přidáním třetí souřadnice. Na obrázku 6.9 jsou v detailech zobrazeny výřezy výsledné triangulace povrchu již bez trojúhelníků, které vznikly přidáním okrajů.



Obrázek 6.9: Výsledná povrchová triangulace po odstranění nadbytečných trojúhelníků – detailní pohledy

6.1.2 Model balustrády – dotykový skener MicroScribe G2X

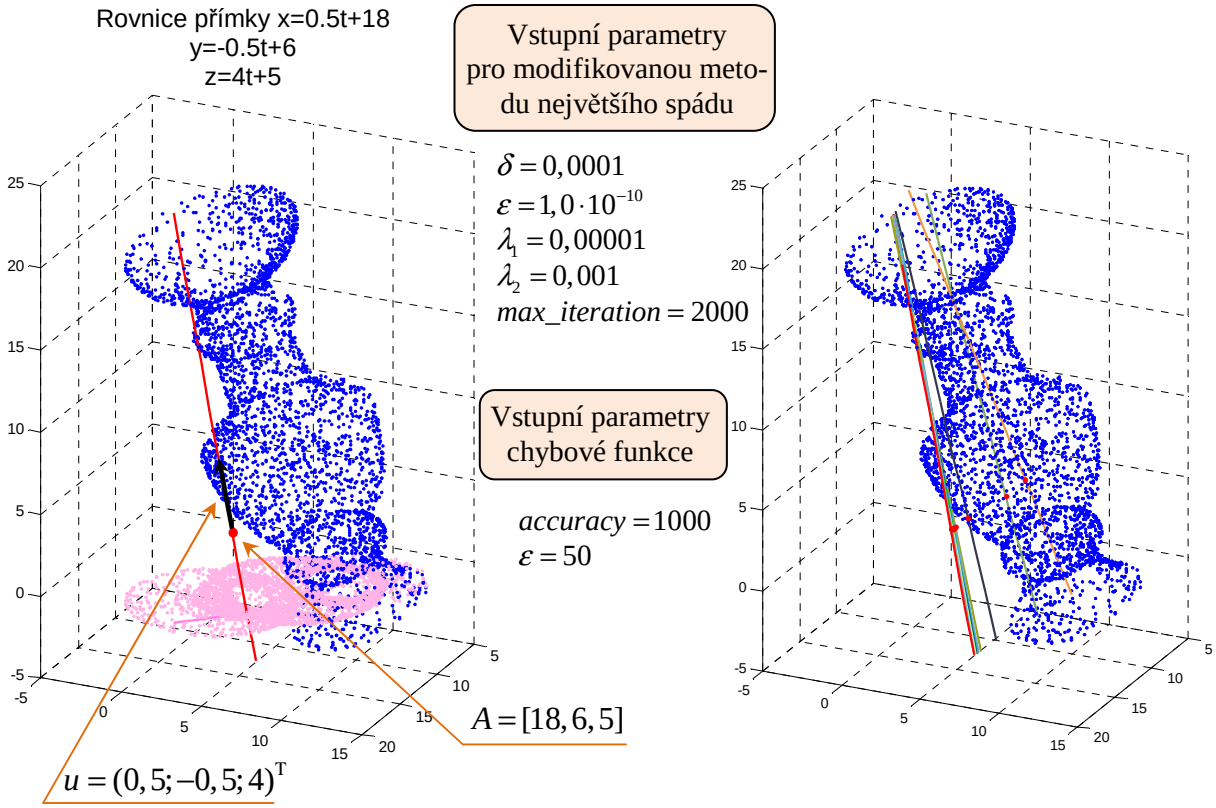
Další bodovou množinu jsme získali naskenováním balustrády, viz obrázek 6.10, dotykovým skenerem MicroScribe G2X. Balustráda je tvořena opět elementárními souosými rotačními plochami. Na obrázku 6.11 je zobrazeno toto mračno bez jakýchkoliv úprav, množina obsahuje 3 346 bodů. Tentokrát je množina řidší, což je dáno ručním měřením bodů povrchu. Opět vycházíme ze znalosti, že bodová množina reprezentuje povrch osově symetrický a tuto osu symetrie hledáme jednak naším postupem navrženým v oddíle 4.3 (*Hledání osy obecné rotační plochy*), přičemž volíme modifikovanou metodu největšího spádu, a jednak metodou ortogonálního prokládání dat popsanou v oddíle 3.3 (*Ortogonální prokládání dat přímkou*).

Výstupy hledání osy symetrie bodového mračna prezentujeme obdobným způsobem jako v předchozím příkladě. Na obrázku 6.12 je společně s bodovým mračnem zakreslena počáteční poloha přímky (přidáváme pro lepší názornost také půdorysy bodů a přímky), kterou postupně optimalizujeme, a předloženy jsou její parametrické rovnice. Na témže obrázku můžeme vidět postupné vylepšování polohy přímky, opět přímku vykreslujeme pouze v několika pozicích. Zvlášť je vykreslena výsledná poloha osy a vypsány jsou její parametrické rovnice, viz obrázek 6.13. Na obrázku 6.14 je zobrazena situace v pomocné soustavě souřadnic v rovině v prvním a posledním kroku iteračního procesu. Dostatečně dobrý odhad osy rotační plochy dostáváme pro uvedenou volbu vstupních parametrů metody (2000 iterací). Délka kroku pro optimalizaci směrového vektoru a určujícího bodu osy je zvolena opět rozdílně. Pokud bychom vyžadovali přesnější výsledek, je možné zvýšit počet iterací nebo požadovanou přesnost či změnit velikost délky kroků λ_1 nebo λ_2.

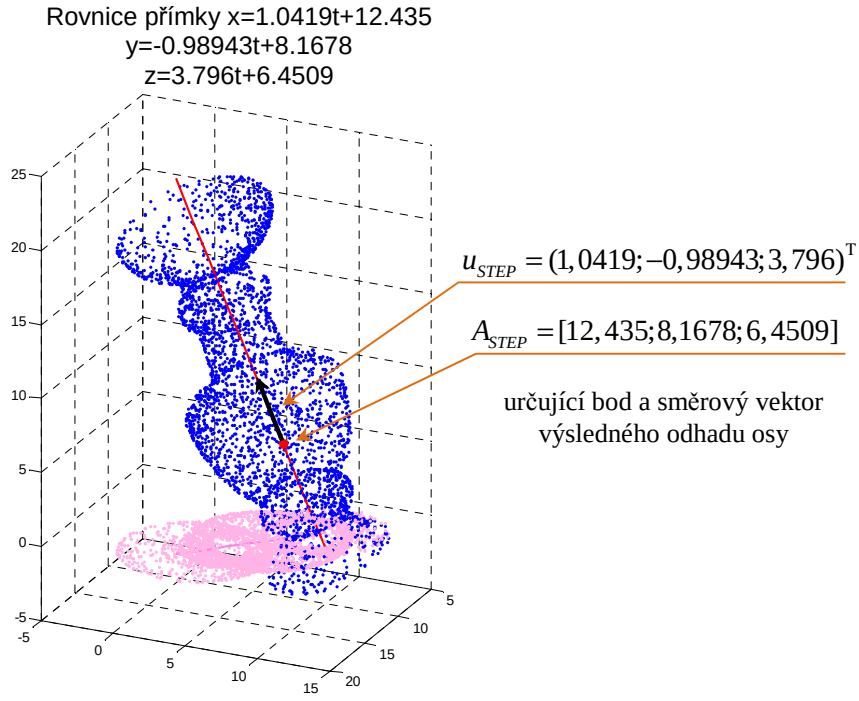
Hledání osy symetrie u této bodové množiny je provedeno také způsobem ortogonálního prokládání dat přímkou. Výsledek této metody již nezobrazujeme, pouze uvádíme určující prvky výsledné přímky.

Obrázek 6.10: Model balustrády tvořené souosými rotačními plochami

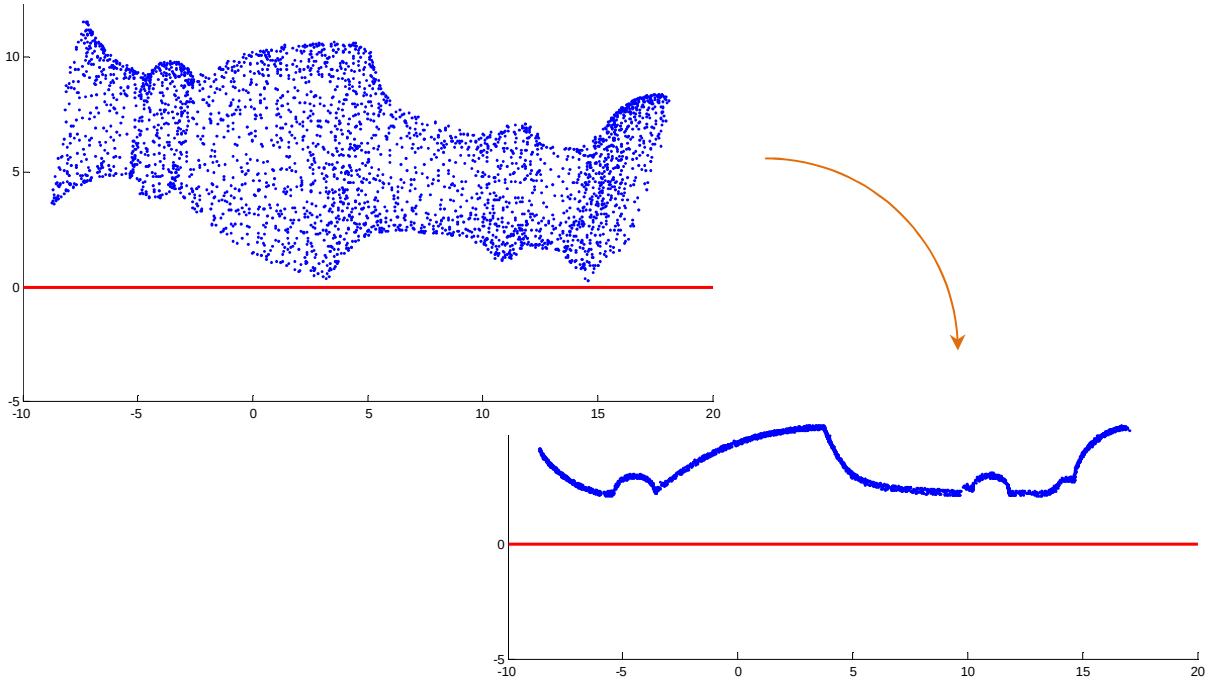
Obrázek 6.11: Bodová množina bez úprav reprezentující povrch balustrády tvořené souosými rotačními plochami – výstup z dotykového skeneru MicroScribe G2X



Obrázek 6.12: Počáteční poloha přímky, vstupní parametry modifikované metody největšího spádu a chybové funkce a postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu



Obrázek 6.13: Výsledný odhad osy

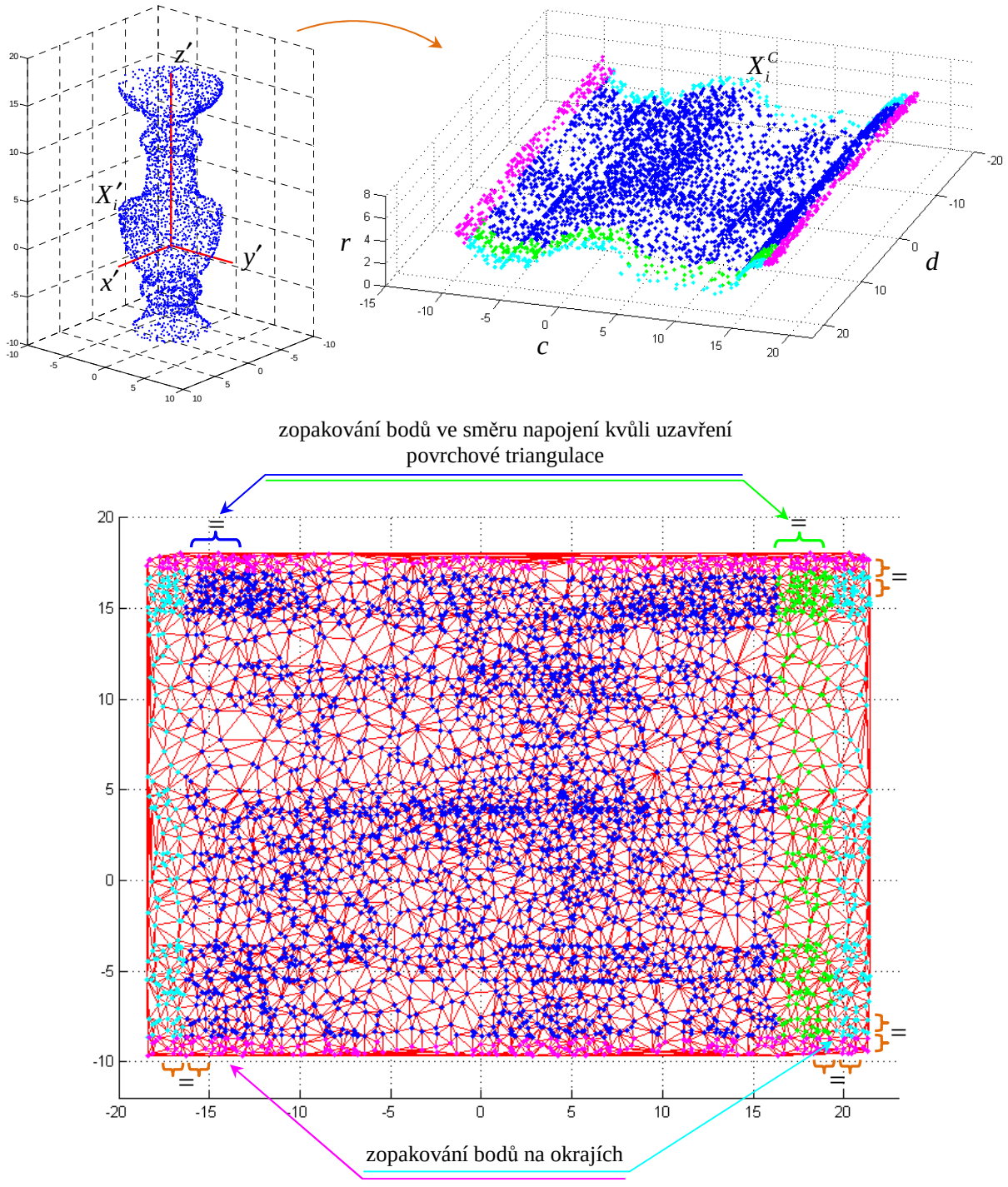


Obrázek 6.14: Situace v pomocné soustavě souřadnic v rovině pro počáteční polohu optimalizované přímky a pro výsledný odhad osy

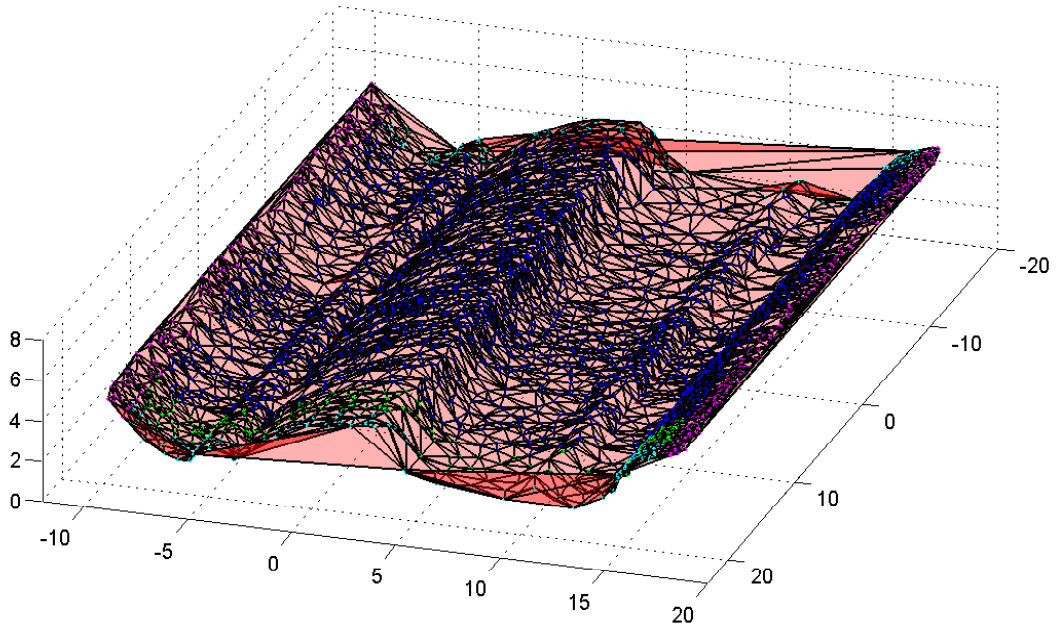
Směrový vektor u_{STEP} odhadu osy, který jsme určili naší metodou, není jednotkový. Po znormování je $u_{STEP} = (0,2567; -0,24377; 0,93524)^{\mathrm{T}}$. Ortogonálním prokládáním dat přímkou vyšel odhad osy se směrovým vektorem $u = (0,25106; -0,2226; 0,94203)^{\mathrm{T}}$ a určujícím bodem $A = [13,91282; 7,18454; 10,67621]$. Porovnáním zjišťujeme, že metody dávají velmi podobný výsledek. V dalších experimentech jsme vyzkoušeli, že v následném zpracování, kdy odhad osy použijeme pro výpočet polygonálního povrchu, nezáleží, který z těchto odhadů použijeme. Kvalita výsledné povrchové triangulace bude stejná.

Hledání povrchové triangulace provádíme opět metodou převodu do cylindrických souřadnic navrženou v oddíle 5.3 (*Triangulace povrchu pomocí transformace*). Na obrázku 6.15 uvažujeme situaci v kartézské soustavě souřadnic S'. Zobrazujeme tedy body $\{X_i'\}_{i=1}^n$ a kladné části poloos x', y', z'. Dále jsou body $\{X_i'\}_{i=1}^n$ převedeny do cylindrických souřadnic podle vzorců (5.12) a zobrazeny v soustavě S' jako body $\{X_i^C\}_{i=1}^n$. Je nutné uvažovat zopakování částí bodů ve směru napojení kvůli uzavření výsledné povrchové triangulace a rovněž zopakování bodů na všech okrajích, abychom se vyhnuli nevhodným tvarům trojúhelníků. Ve výsledné povrchové triangulaci překryv a navíc přidané trojúhelníky odstraňujeme. Na obrázku 6.15 je předložen také výsledek Delaunayovy triangulace nad půdorysy bodů $\{X_i^C\}_{i=1}^n$ a na obrázku 6.16 výšková mapa, která vznikne zpětným přidáním třetí souřadnice. Výslednou povrchovou triangulaci a její detail můžeme vidět na obrázku 6.17.

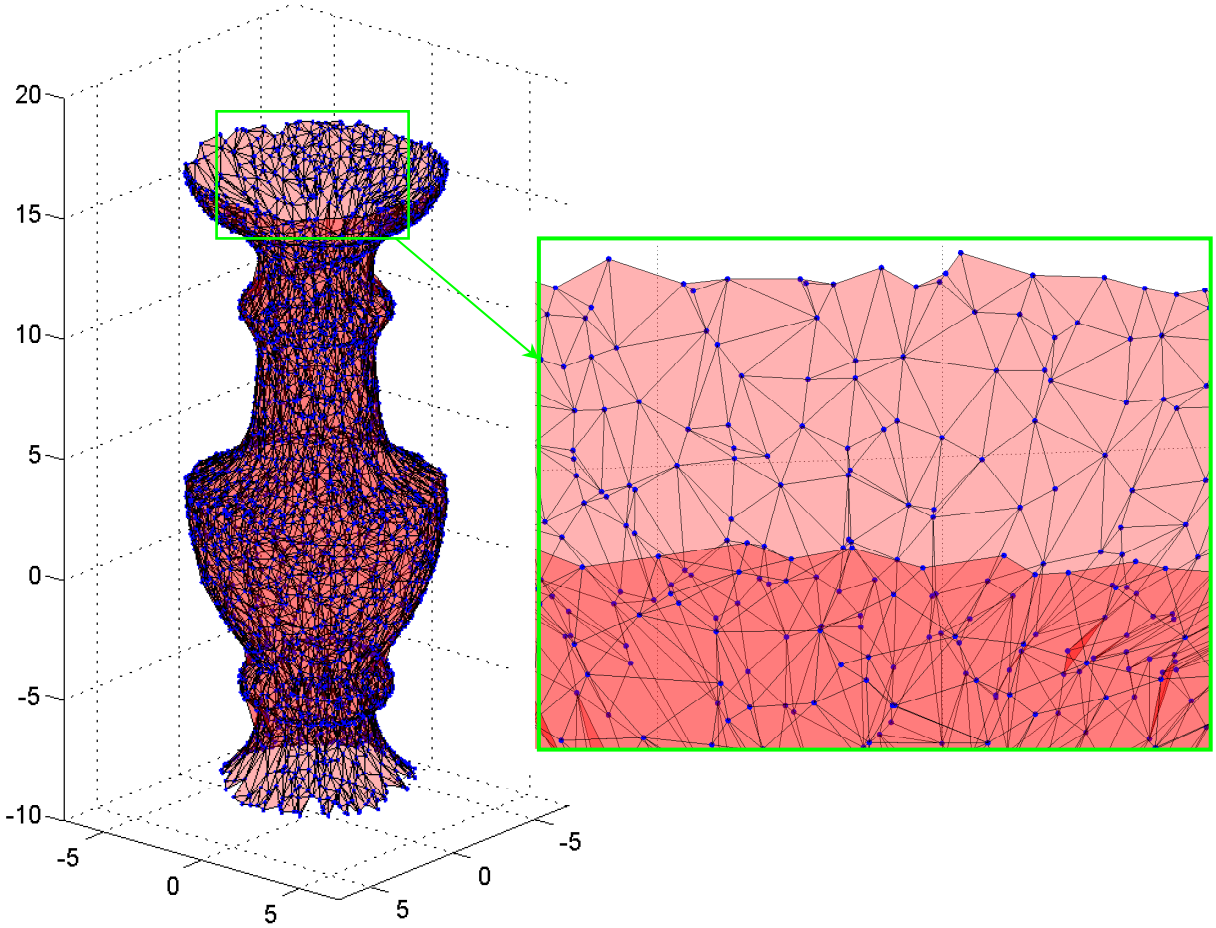
Je vidět, že na rozdíl od optického skeneru je vstupní bodová množina získaná ručním skenováním povrchu velice nepravidelná. Z toho důvodu je i výsledná povrchová triangulace velmi nerovnoměrná.



Obrázek 6.15: Transformace kartézské soustavy souřadnic v prostoru, převod do cylindrických souřadnic a Delaunayova triangulace na množině půdorysů bodů – s opakujícími se body na okrajích



Obrázek 6.16: Výšková mapa s opakujícími se body na okrajích



Obrázek 6.17: Výsledná povrchová triangulace bez chyb a část v detailu

6.1.3 Model balustrády – laserový skener Roland Picza LPX-1200

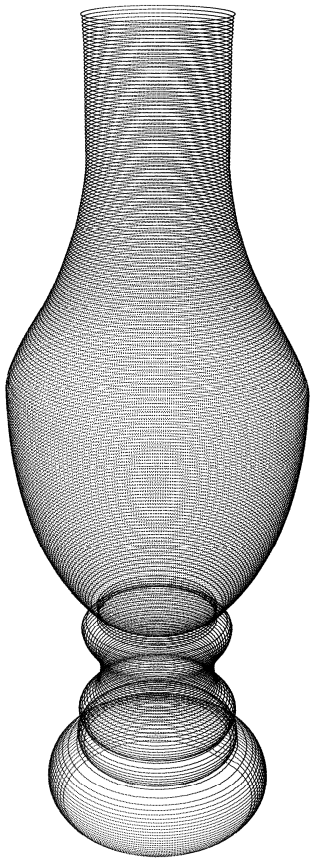
Bodová množina, kterou jsme dále zpracovávali, vznikla skenováním modelu balustrády opět tvořené elementárními souosými rotačními plochami, viz obrázek 6.18, laserovým skenerem Roland Picza LPX-1200. Na obrázku 6.19 je zobrazeno toto mračno bez jakýchkoliv úprav, množina obsahuje 105 200 bodů. Tento laserový skener pracuje tak, že snímání bodů provádí při současném otáčení modelu stojícím na pevné podložce. Díky tomu máme naskenovaný model ze všech stran, není tedy nutná dodatečná registrace. Navíc jsou body na modelu plochy nasnímány tak, že téměř přesně leží na rovnoběžkových kružnicích.

Obrázek 6.18: Model balustrády tvořené souosými rotačními plochami

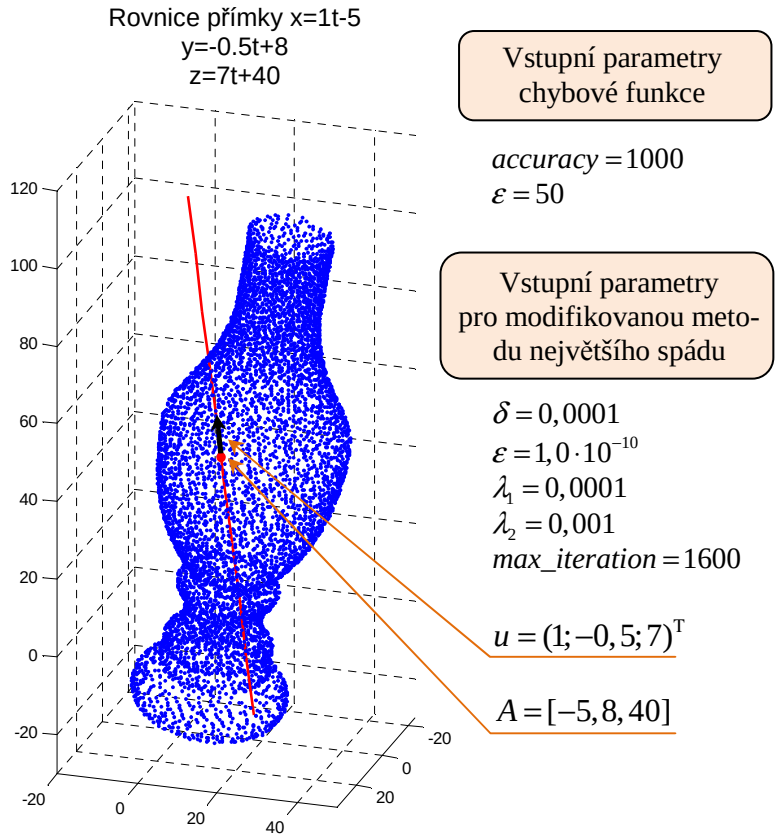
Osu symetrie bodové množiny hledáme jednak vlastním postupem navrženým v oddíle 4.3 (*Hledání osy obecné rotační plochy*), přičemž volíme modifikovanou metodu největšího spádu, a jednak metodou ortogonálního prokládání popsanou v oddíle 3.3 (*Ortogonální prokládání dat přímkou*).

Výstupy hledání osy symetrie bodového mračna prezentujeme analogicky jako v předchozích dvou příkladech. Na obrázku 6.20 je společně s bodovým mračnem zakreslena počáteční poloha přímky (přidáváme pro lepší názornost také půdorysy bodů a přímky), kterou postupně optimalizujeme, a předloženy jsou její parametrické rovnice. Na obrázku 6.21 je zobrazena situace v pomocné soustavě souřadnic v rovině v prvním a posledním kroku iteračního procesu. Obrázek 6.22 ilustruje postupné vylepšování polohy přímky, přičemž přímku vykreslujeme pouze v několika pozicích. Výpočty probíhají se všemi body vstupní množiny, ovšem ve všech obrázcích zobrazujeme pouze ztenčenou množinu bodů (zde je použita šestnáctina) kvůli větší přehlednosti. Zvlášť zobrazujeme výsledný odhad osy a vypisujeme její parametrické rovnice. Dostatečně dobrý odhad osy rotační plochy dostáváme pro uvedenou volbu vstupních parametrů metody (1600 iterací). Pro optimalizaci směrového vektoru a určujícího bodu osy volíme opět různou délku kroku, neboť u takto rozsáhlé množiny se již výrazně projeví rozdílnost spádu těchto parametrů. Hodnota λ_1 je délka kroku v optimalizaci směrového vektoru a hodnota λ_2 v optimalizaci určujícího bodu přímky. Opět lze měnit počet iterací nebo požadovanou přesnost či změnit velikost délky kroků λ_1 nebo λ_2.

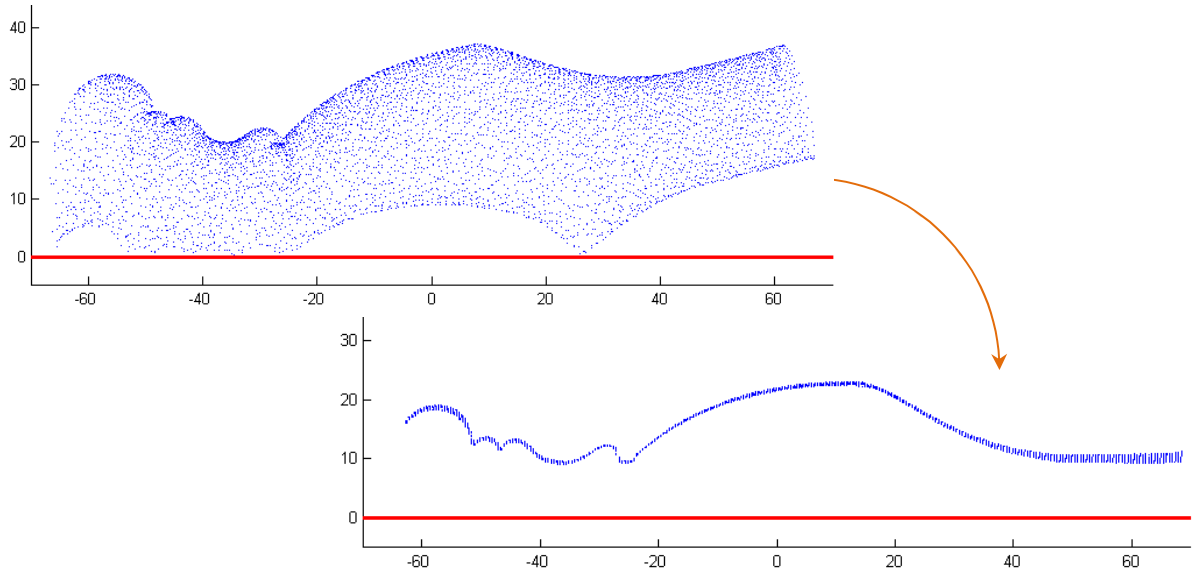
Hledání osy symetrie u této bodové množiny je provedeno také způsobem ortogonálního prokládání dat přímkou. Výsledek této metody již nezobrazujeme, pouze uvádíme určující prvky výsledné přímky.



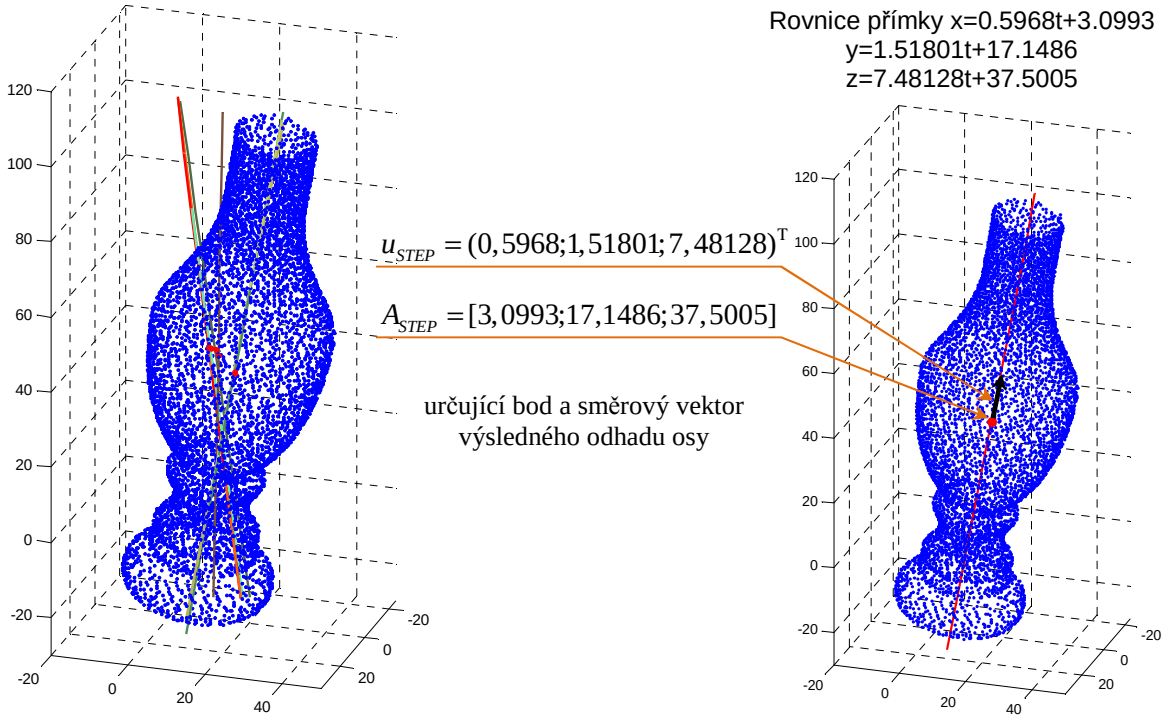
Obrázek 6.19: Bodová množina bez úprav reprezentující část povrchu balustrády tvořené souosými rotačními plochami – výstup z laserového skeneru Roland Picza LPX-1200



Obrázek 6.20: Počáteční poloha přímky a vstupní parametry modifikované metody největšího spádu a chybové funkce



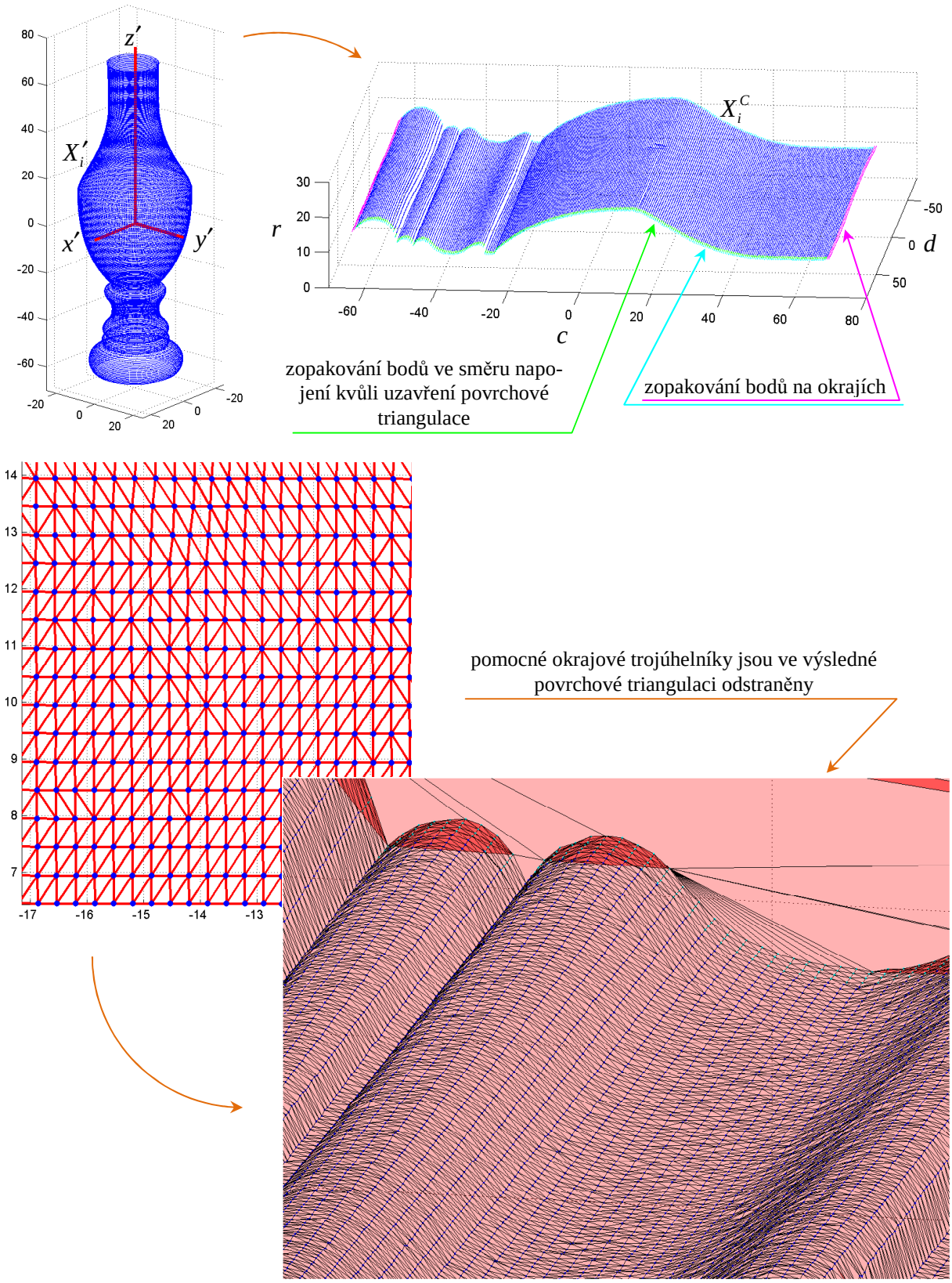
Obrázek 6.21: Situace v pomocné soustavě souřadnic v rovině pro počáteční polohu optimalizované přímky a pro výsledný odhad osy



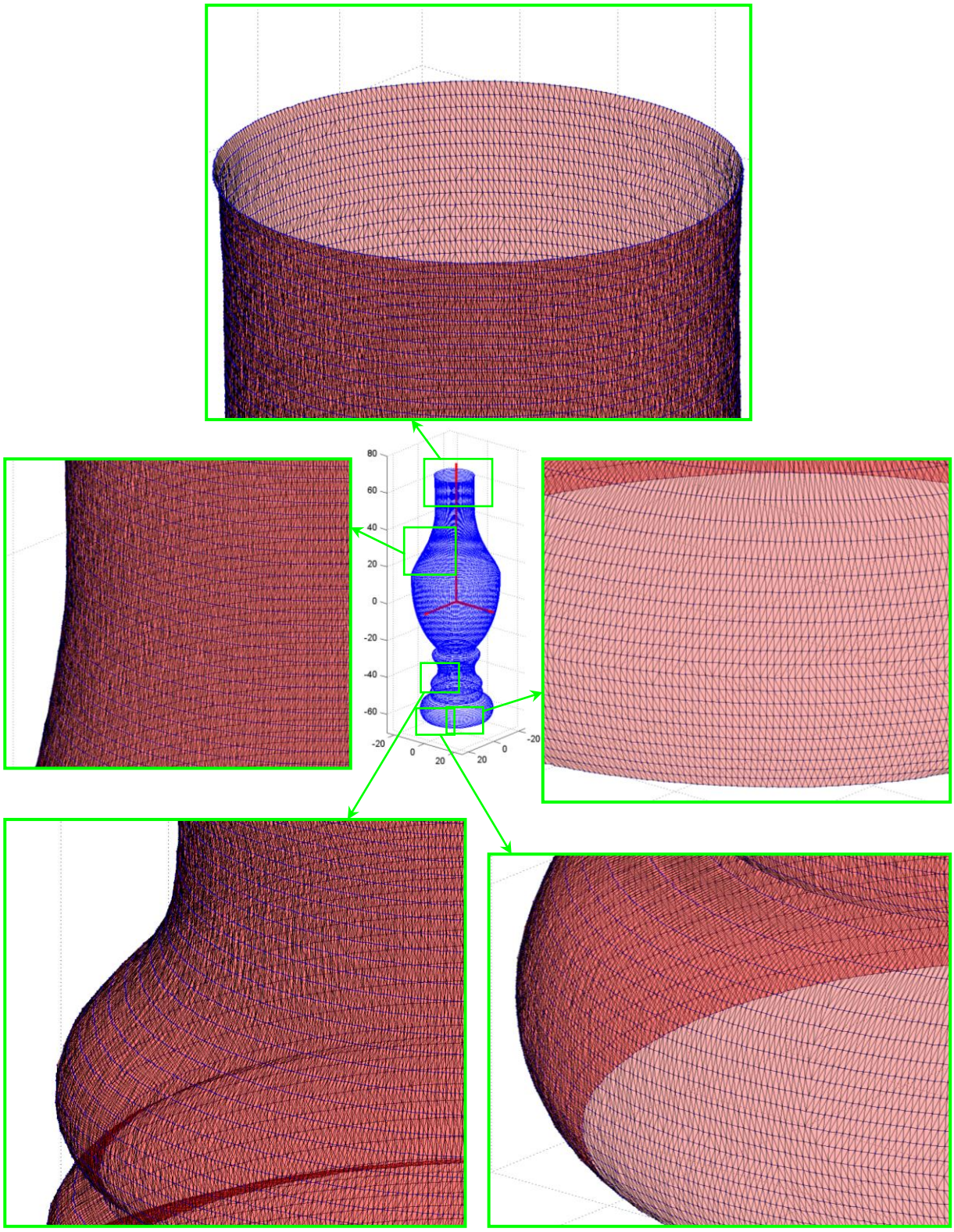
Obrázek 6.22: Postupná optimalizace přímky minimalizací chybové funkce modifikovanou metodou největšího spádu a výsledný odhad osy

Pokud porovnáme po znormování směrový vektor $u_{STEP} = (0,07794;0,19825;0,97705)^T$ odhadu osy, který jsme určili naší metodou, se směrovým vektorem určeným ortogonálním prokládáním dat přímkou, zjišťujeme, že metody dávají velmi podobný výsledek. Ortogonálním prokládáním dat přímkou vyšel odhad osy se směrovým vektorem $u = (0,07943;0,19563;0,97746)^T$ a určujícím bodem $A = [5,22315;18,07676;40,07772]$. Kvalita výsledné povrchové triangulace je v obou případech stejná.

Hledání povrchové triangulace provádíme opět metodou převodu do cylindrických souřadnic navrženou v oddíle 5.3 (*Triangulace povrchu pomocí transformace*) a výsledky prezentujeme zcela analogicky jako v předchozích případech. Nyní množinu bodů při zobrazení nijak neztenčujeme. Na obrázku 6.23 uvažujeme situaci v kartézské soustavě souřadnic S'. Dále jsou body $\{X_i\}_{i=1}^n$ převedeny do cylindrických souřadnic podle vzorců (5.12) a zobrazeny v soustavě S' jako body $\{X_i^C\}_{i=1}^n$. Je nutné uvažovat zopakování částí bodů ve směru napojení kvůli uzavření výsledné povrchové triangulace a rovněž zopakování bodů na všech okrajích, abychom se vyhnuli nevhodným tvarům trojúhelníků v triangulaci. Pomocné trojúhelníky ve výsledné povrchové triangulaci odstraňujeme. Na obrázku 6.23 můžeme také vidět Delaunayovu triangulaci nad půdorysy bodů $\{X_i^C\}_{i=1}^n$ a výškovou mapu. Výsledná povrchová triangulace je v detailech znázorněna na obrázku 6.24.



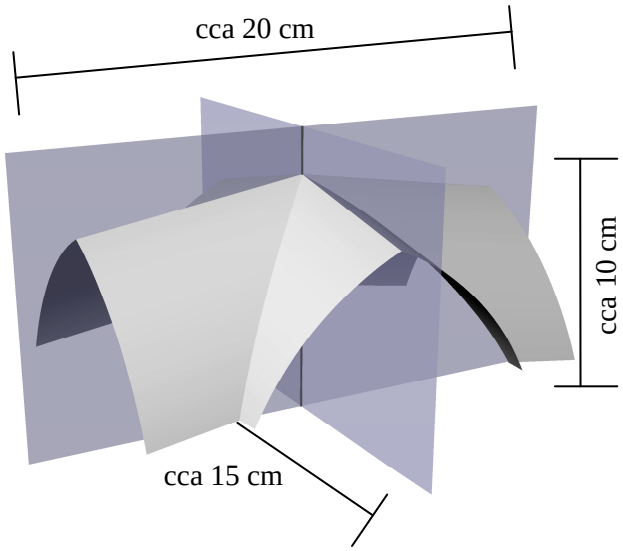
Obrázek 6.23: Transformace kartézské soustavy souřadnic v prostoru, převod do cylindrických souřadnic, Delaunayova triangulace na množině půdorysů bodů (detail) a výšková mapa (detail) – s opakujícími se body na okrajích



Obrázek 6.24: Výsledná povrchová triangulace po odstranění nadbytečných trojúhelníků – detailní pohledy

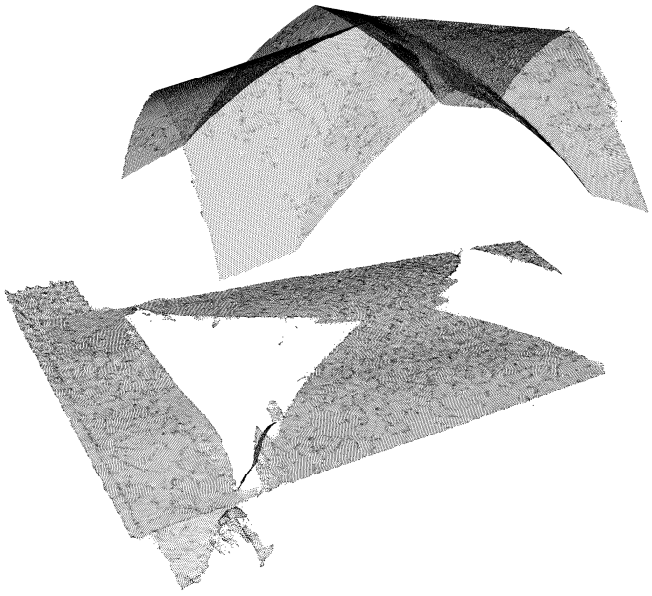
6.1.4 Model křížové klenby – optický systém Vectra3D

Bodová množina, kterou dále zkoumáme, je získána skenováním modelu křížové klenby, viz obrázek 6.25 optickým systémem Vectra3D. Model křížové klenby je tvořen osmi částmi kosé kruhové válcové plochy a je symetrický podle dvou navzájem kolmých rovin, viz obrázek 6.25. Na obrázku 6.26 je vykresleno bodové mračno bez jakýchkoliv úprav, množina obsahuje 154 175 bodů. Jak je vidět, bodová množina zahrnuje také části naskenovaného okolí. Tyto nadbytečné body odstraňujeme ručně v modelovacím softwaru Rhinoceros.

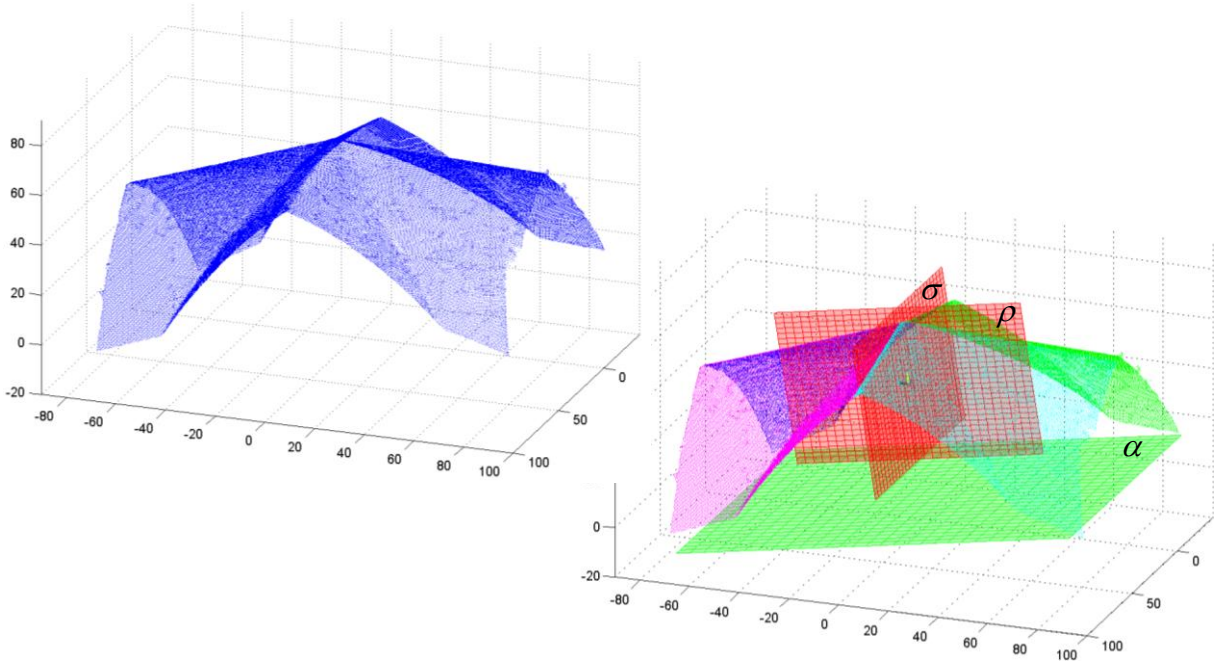


Obrázek 6.25: Model křížové klenby tvořené částmi kosých kruhových válcových ploch

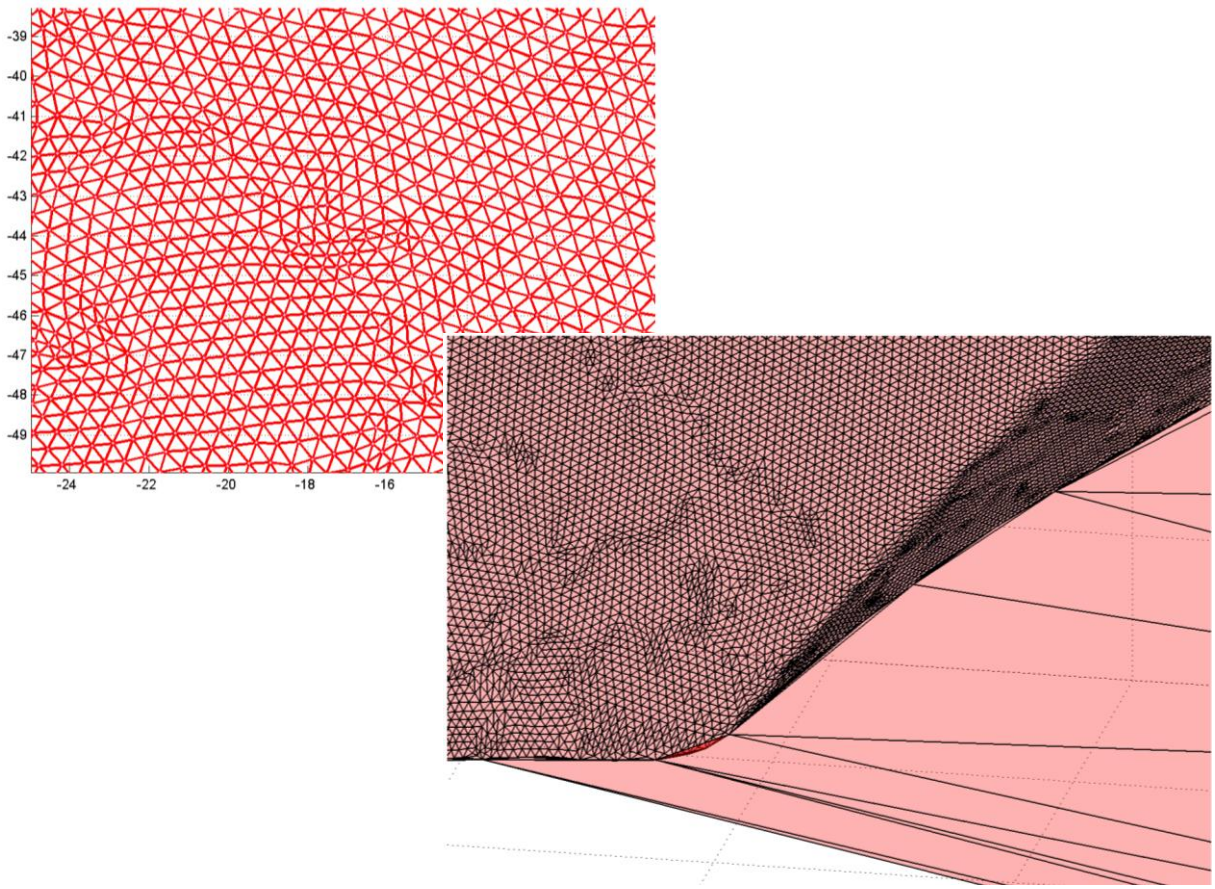
Ke konstrukci povrchové triangulace užijeme rovinových symetrií bodové množiny. Nejdříve nalezneme přibližnou polohu navzájem kolmých rovin symetrií ρ a σ pomocí iteračního algoritmu navrženého v oddíle 4.4 (*Hledání rovinových symetrií*). Dále dourčíme rovinu α, která je kolmá k oběma rovinám symetrií a v prostoru je umístěna libovolně, viz obrázek 6.27. Následně postupujeme stejně jako v příkladě 5.4. Body pravoúhle promítneme do roviny α a kartézskou soustavu souřadnic v prostoru přetransformujeme tak, aby přešla do roviny (x, y). Dále nad pravoúhlými průměty (nyní přetransformované do roviny (x, y)) provedeme Delaunayovu triangulaci, viz obrázek 6.28. Na závěr přejdeme zpět k bodům vstupní množiny a dostáváme povrchovou triangulaci, jak můžeme vidět na obrázcích 6.28 a 6.29. Výslednou triangulaci povrchu na obrázku 6.29 zobrazujeme po odstranění nevhodných trojúhelníků v několika detailech.



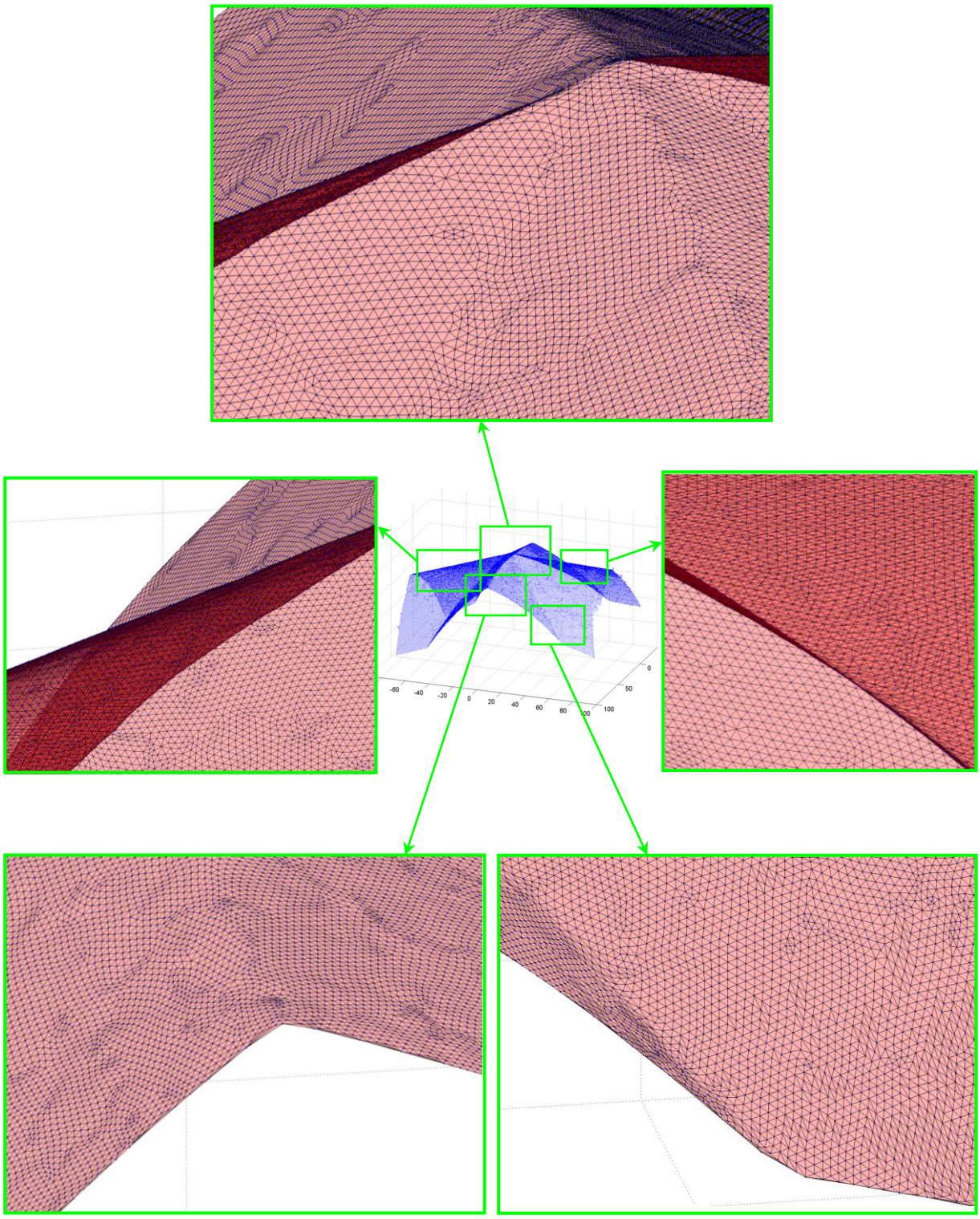
Obrázek 6.26: Bodová množina bez úprav reprezentující povrch křížové klenby – výstup z optického systému Vectra3D



Obrázek 6.27: Vstupní bodová množina po odstranění bodů nepatřících skenovanému objektu a dvě navzájem kolmé roviny symetrií ρ a σ a třetí na ně kolmá pomocná rovina α



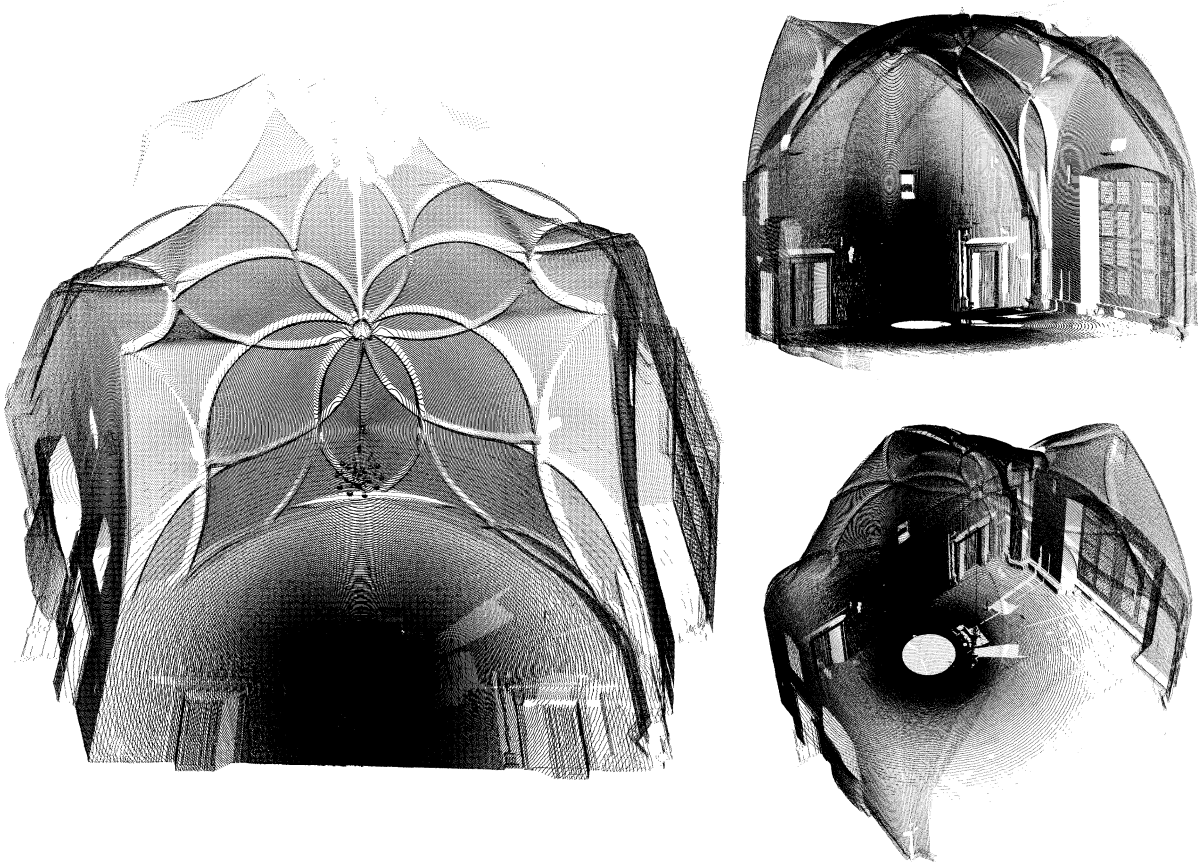
Obrázek 6.28: Delaunayova triangulace nad pravoúhlými průměty bodů v přetransformované rovině α a v detailu povrchová triangulace před odstraněním nevhodných trojúhelníků



Obrázek 6.29: Výsledná povrchová triangulace po odstranění nadbytečných trojúhelníků – detailní pohledy

6.2 Reálná data reprezentující povrchy skutečných staveb

V tomto závěrečném oddíle prezentujeme zpracování bodového mračna získaného skenováním interiéru Vladislavského sálu na Pražském hradě, viz obrázek 6.30. Množina bodů byla naskenována panoramatickým laserovým skenerem Callidus 1.1. Bodové mračno nám poskytl Ing. Jan Řezníček z Katedry mapování a kartografie, Fakulta stavební, České vysoké učení technické v Praze.

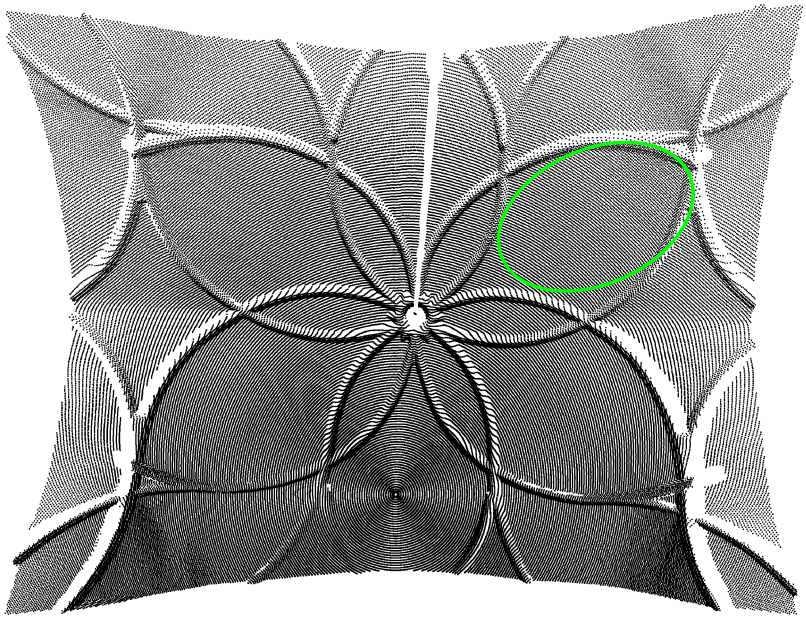


Obrázek 6.30: Bodová množina bez úprav reprezentující interiér Vladislavského sálu na Pražském hradě – výstup z laserového skeneru Callidus 1.1

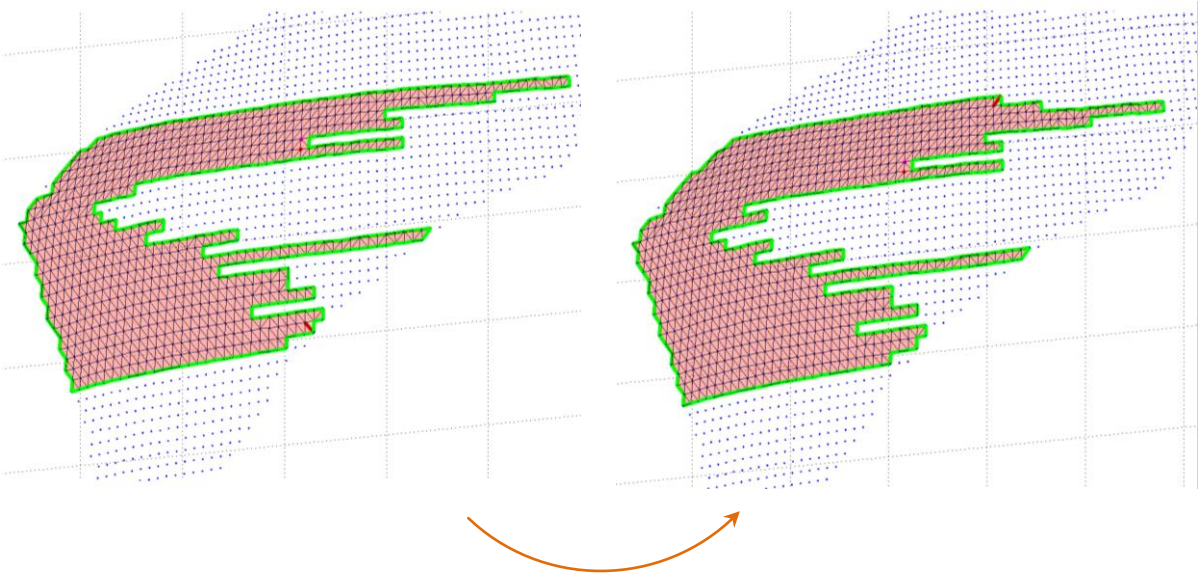
Před dalším zpracováním vstupní bodové mračno ručně upravíme v modelovacím softwaru Rhinoceros. Věnovat se dále budeme povrchové triangulaci části klenby, ostatní body mračna proto odstraníme. Na obrázku 6.31 je zobrazena část bodové množiny po této redukci a vyznačen jen ten úsek, jehož triangulaci dále řešíme. Takto po částech řešíme výpočet triangulace povrchu kvůli jeho velké členitosti a velkému množství bodů množiny. Nelze zpracovávat větší celek také z toho důvodu, že především v místech, která odpovídají žebrům klenby, obsahuje mračno bodů velké chyby a nepřesnosti.

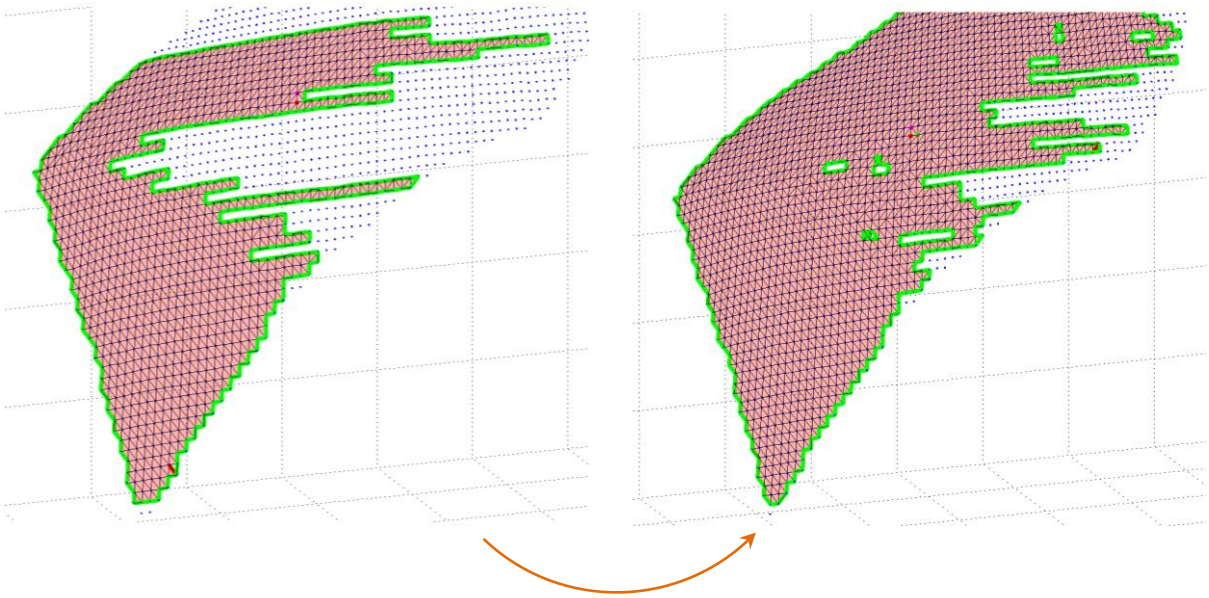
Povrchovou triangulaci řešíme algoritmem navrženým v oddíle 5.2 (*Inkrementální konstrukce polygonální sítě*). Na obrázku 6.32 můžeme sledovat v detailu postupný výpočet

trojúhelníkové sítě jedné takové části bodové množiny, přičemž je vždy zvýrazněna hrana, ke které se v daném kroku hledá vhodný vrchol nového trojúhelníka v síti. Jedná se o množinu bodů, kdy je nutné použít restartování procesu konstrukce sítě. Na obrázku 6.32 je zachyceno několik okamžiků, kdy se konstrukce sítě zastaví, neboť je již prázdná fronta hran, které čekají na zpracování. Výsledek povrchové triangulace uvažované části bodové množiny je vidět na obrázku 6.33.

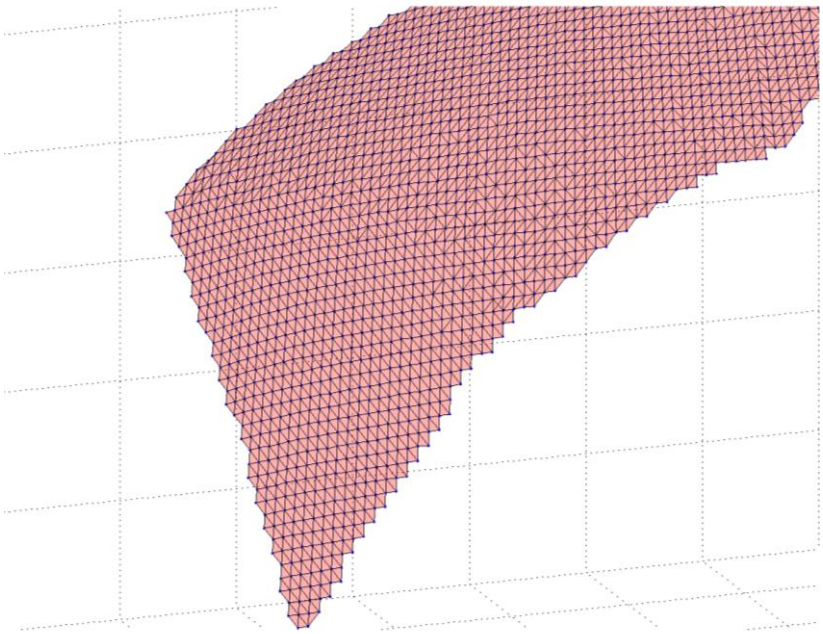


Obrázek 6.31: Část bodové množiny reprezentující interiér Vladislavského sálu na Pražském hradě a vyznačený úsek, který dále zpracováváme





Obrázek 6.32: Zastavení konstrukce povrchové triangulace, výpočet okrajových částí a děr v síti a restartování procesu tvorby sítě – několik okamžiků



Obrázek 6.33: Výsledná povrchová triangulace části bodové množiny

Na závěr poznamenejme, že jsme testovali ještě několik dalších bodových množin, které jsme získali skenováním geometrických modelů. Tyto a další bodové množiny jsou k dispozici na přiloženém výměnném médiu.

Kapitola 7

Závěr a budoucí práce

Disertační práce představuje přínos v oblasti digitální rekonstrukce povrchů technické a stavební praxe z mračen bodů. Hlavním výsledkem práce je předložení nových výsledků a metod přispívajících k jednotlivým fázím v současnosti používaném rekonstrukčním procesu ze vstupních množin bodů, které reprezentují speciální typy reálných povrchů. Nejvíce pozornosti bylo věnováno analýze vstupního mračna bodů popisujícího nějaký objekt.

V práci jsem prezentovala několik zcela nových algoritmů (*hledání osy rotační válcové plochy, hledání osy obecné rotační plochy, hledání rovinových symetrií*, *konstrukce povrchové triangulace pomocí rovinových symetrií*) a vylepšení některých algoritmů existujících (*odhady tečných rovin a normál bodové množiny, inkrementální konstrukce polygonální sítě, triangulace povrchu pomocí transformace*), které řeší dílčí fáze rekonstrukce povrchů. Veškeré navržené řešící postupy jsou založeny na geometrických myšlenkách, které lze použít díky znalostem speciálních vlastností rekonstruovaného objektu. Tento aspekt nebývá v obecných řešících postupech často brán v úvahu, navíc mezi jeho výhody patří snadná implementace a názornost. Některé metody dovolují rovněž interaktivní zásah uživatele. V problematice digitální rekonstrukce povrchů se jedná o inovativní přístupy.

Významným výsledkem disertační práce jsou rovněž analýzy a zpracování nejen syntetických bodových množin ale především reálných bodových množin, které byly získané vlastnoručním měřením. Veškerá použitá data jsou také k dispozici na přiloženém výměnném médiu.

V úvodu práce je podrobně uveden také současný stav poznání rekonstrukce povrchů z mračen bodů a popsány jsou jednotlivé fáze řešení dané problematiky. Nové řešící postupy, které vycházejí více z geometrických znalostí rekonstruovaného objektu, přispívají k dílčím krokům tohoto celkově velmi náročného procesu rekonstrukce. Při vývoji nových technik jsem přitom vycházela z numerických diferenciálních metod, proto je v práci také v přiměřené míře uvedena nezbytná teorie vztahující se k těmto oblastem. Ke každému zpracovávanému tématu přikládám velké množství příkladů, na kterých demonstruji správnost postupů. Všechny příklady ilustruji vlastními obrázky, které jsou především z didaktického hlediska také hodnotným přispěním. Nově navržené metody řešení a algoritmy a modifikované existující postupy aplikuji na počítačově generovaná data, data popisující modely reálných objektů nebo i části reálných staveb.

Jedním z nejvýznamnějších výsledků disertační práce jsou implementace navržených algoritmů v moderním programovacím jazyku a interaktivním prostředí MATLAB. Pro snadnější orientaci čtenáře v přiložených programech uvádím následující tabulku 7.1. Tabulka obsahuje vždy název programu, odkaz na oddíl v disertační práci, popis cesty k pro-

gramu na přiloženém výměnném médiu, a stručný popis programu. V příslušných adresářích je umístěna také konkrétní bodová množina, pro kterou jsou upraveny vstupní parametry daného programu.

Přehled programů z výpočetního prostředí MATLAB

Název	*Oddíl*	*Adresář* programy/..	*Stručný popis*
mnc_primka.m	3.2	mnc2d	Prokládání dat v rovině přímkou metodou nejmenších čtverců.
mnc_polynom.m	3.2	mnc2d	Prokládání dat v rovině polynomem libovolného stupně metodou nejmenších čtverců.
primka_orto_2d.m	3.3	orto_primka2d	Ortogonální prokládání dat v rovině přímkou.
primka_orto_3d.m	3.3	orto_primka3d	Ortogonální prokládání dat v prostoru přímkou.
mnc_rovina.m	3.4	mnc3d	Prokládání dat v prostoru rovinou metodou nejmenších čtverců.
mnc_polynom_3d.m	3.4	mnc3d	Prokládání dat v prostoru polynomem dvou proměnných libovolného stupně metodou nejmenších čtverců.
rovina_orto.m	3.5	orto_rovina	Ortogonální prokládání dat v prostoru rovinou.
pca_2d.m	3.6	pca	Analýza hlavních komponent v rovině.
pca_3d.m	3.6	pca	Analýza hlavních komponent v prostoru.
nejvetsi_spad_modifik.m	4.1.1	minimalizace	Modifikovaná metoda největšího spádu k hledání minima funkce více proměnných.
nejvetsi_spad.m	4.1.1	minimalizace	Metoda největšího spádu k hledání minima funkce více proměnných.
osa_rot_valce_spad_modifik.m	4.2	osa_rot_valce	Hledání osy rotační válcové plochy modifikovanou metodou největšího spádu.
osa_rot_valce_spad.m	4.2	osa_rot_valce	Hledání osy rotační válcové plochy metodou největšího spádu.
osa_rovnobezek_spad_modifik.m	4.2	osa_rovnobezek	Hledání osy rovnoběžek modifikovanou metodou největšího spádu.
osa_rovnobezek_spad.m	4.2	osa_rovnobezek	Hledání osy rovnoběžek metodou největšího spádu.
osa_rot_plochy_spad_modifik.m	4.3	osa_rot_plochy	Hledání osy obecné rotační plochy modifikovanou metodou největšího spádu.

osa_rot_plochy_spad.m	4.3	osa_rot_plochy	Hledání osy obecné rotační plochy metodou největšího spádu.
symetrie_jedna_rovina.m	4.4	symetrie	Hledání roviny symetrie bodové množiny.
symetrie_dve_roviny.m	4.4	symetrie	Hledání dvou navzájem kolmých rovin symetrií.
symetrie_tri_roviny.m	4.4	symetrie	Hledání tří navzájem kolmých rovin symetrií.
odhad_tecnych_rovin.m	5.1	tecne_roviny	Odhady tečných rovin a normál bodové množiny.
inkrementalni_povrch.m	5.2	povrch/inkrement	Inkrementální konstrukce povrchové triangulace.
cylindricke_souradnice.m	5.3	povrch/cylindr	Konstrukce povrchové triangulace pomocí transformace.
triangulace_symetrie.m	5.4	povrch/symetrie	Konstrukce povrchové triangulace pomocí rovinových symetrií.

Tabulka 7.1: Přehled přiložených programů naimplementovaných ve výpočetním prostředí MATLAB

I přes množství uvedených postupů a metod a rozsáhlost práce nepovažuji téma rekonstrukce povrchů v žádném případě za vyčerpané. Již nyní pracuji na dalších fázích rekonstrukčního procesu a věnuji se především analytické deskripci povrchu reprezentovaného bodovým mračnem. Pro tyto účely vyvíjím metody evolucí hladkých povrchů, jejichž principem je postupné vylepšování tvaru a polohy daného typu plochy, která ve výsledku co nejlépe vyjadřuje vstupní mračno bodů. Použití evoluce vyžaduje jednak předchozí orientaci vstupní bodové množiny pomocí osových nebo rovinových symetrií, jednak konstrukci povrchové triangulace. Obojí bylo zpracováno v předložené disertační práci. Na navržené metody řešení v dalším výzkumu navazuji.

Disertační práce *Analýza bodových množin reprezentujících povrchy technické praxe* je ukázkou uplatnění znalostí klasické a aplikované geometrie v praktické problematice digitální rekonstrukce, tedy v oblasti počítačové grafiky. Práce je jednou z prvních rozsáhlejších českých publikací věnujících se tomuto okruhu témat.

Výsledky práce mohou napomoci ke zvýšení zájmu o studium matematiky a převážně geometrie na vysokých školách. Rovněž může být práce inspirujícím zdrojem pro další výzkum těchto oblastí.

Literatura

Karim M. **Abadir**, Jan R. **Magnus** (2005). *Matrix Algebra*. Cambridge University Press, USA.

Elsa **Abbena**, Alfred **Gray**, Simon **Salamon** (2006). *Modern Differential Geometry of Curves and Surfaces with MATHEMATICA*. Taylor & Francis Group, USA.

Sung Joon **Ahn** (2004). *Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space*. Springer-Verlag Berlin Heidelberg, Germany.

Martin **Aigner**, Zbyněk **Šír**, Bert **Jüttler** (2006). *Least-squares Approximation by Pythagorean Hodograph Spline Curves via an Evolution Process.* Journal, Geometric Modelling and Processing, Volume 4077, pp. 45-58, Springer.

Maria-Elena **Algorri**, Francis **Schmitt** (1996). *Surface Reconstruction from Unstructured 3D Data*. Journal, Computer Graphics Forum, Volume 15 (1), pp. 47-60, John Wiley & Sons.

Jiří **Anděl** (1985). *Matematická statistika*. SNTL – Nakladatelství technické literatury, Praha.

Dominique **Attali** (1997). *r-Regular Shape Reconstruction from Unorganized Points.* Proceedings of the 13th annual symposium on Computational Geometry, Nice, France, pp. 248-253, Association for Computing Machinery.

Marco **Attene**, Bianca **Falcidieno**, Michela **Spagnuolo** (2006). *Hierarchical Mesh Segmentation Based on Fitting Primitives*. Journal, The Visual Computer, Volume 22, pp. 181-193, Springer.

Chandrajit L. **Bajaj**, Fausto **Bernardini**, Guoliang **Xu** (1997). *Reconstructing Surfaces and Functions on Surfaces from Unorganized 3d Data*. Journal, Algorithmica, Volume 19, pp. 243-261, Springer.

Jindřich **Bečvář** (2002). *Lineární algebra*. MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Fausto **Bernardini**, Chandrajit L. **Bajaj** (1997). *Sampling and Reconstructing Manifolds Using Alpha-Shapes*. Proceedings of the 9th Canadian Conference on Computational Geometry, Kingston, Ontario, Canada, pp. 193-198, Queen's University.

Radek **Bien** (2008). *Rekonstrukce terénu z vrstevnicových dat*. Bakalářská práce, České vysoké učení technické, Fakulta elektrotechnická, Česká republika.

Eric **Bittar**, Nicolas **Tsingos** , Marie-Paule **Gascuel** (1995). *Automatic Reconstruction of Unstructured 3D Data: Combining a Medial Axis and Implicit Surfaces*. Journal, Computer Graphics Forum, Volume 14 (3), pp. 457-468, John Wiley & Sons.

Leo **Boček**, Václav **Kubát** (1983). *Diferenciální geometrie křivek a ploch*. SPN – Státní pedagogické nakladatelství, Praha.

Jean-Daniel **Boissonnat** (1984). *Geometric Structures for Three-dimensional Shape Representation*. Journal, ACM Transactions on Graphics, Volume 3 (4), pp. 266-286, Association for Computing Machinery.

Alexandre **Boulch**, Renaud **Marlet** (2012). *Fast and Robust Normal Estimation for Point Clouds with Sharp Features*. Journal, Computer Graphics Forum, Volume 31 (5), pp. 1765-1774, John Wiley & Sons.

Richard **Bronson** (1991). *Matrix Methods: An Introduction*. Academic Press, USA.

Bruno **Budinský** (1983). *Analytická a diferenciální geometrie*. SNTL – Nakladatelství technické literatury, Praha.

Bruno **Budinský**, Bořivoj **Kepr** (1970). *Základy diferenciální geometrie s technickými aplikacemi*. SNTL – Nakladatelství technické literatury, Praha.

Xingping **Cao**, Neelima **Shrikhande** (1991). *Quadric Surface Fitting for Sparse Range Data*. Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, USA, pp. 123-128, IEEE Computer Society.

Manfredo Perdiago do **Carmo** (1976). *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, USA.

David **Cohen-Steiner**, Pierre **Alliez**, Mathieu **Desbrun** (2004). *Variational Shape Approximation*. Journal, ACM Transactions on Graphics, Volume 23 (3), pp. 905-914, Association for Computing Machinery.

Creaform (2012). Komerční internetové stránky [online], http://www.creaform3d.com, (březen 2012).

Mark **de Berg**, Otfried **Cheong**, Marc **van Kreveld**, Mark **Overmars** (2010). *Computational Geometry: Algorithms and Applications*. Springer-Verlag Berlin Heidelberg, Germany.

James W. **Demmel** (1997). *Applied Numerical Linear Algebra.* SIAM – Society for Industrial and Applied Mathematics, USA.

Tamal Krishna **Dey** (2007). *Curve and Surface Reconstruction*. Cambridge University Press, USA.

Tamal Krishna **Dey**, Gang **Li,** Jian **Sun** (2005). *Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method.* Eurographics/IEEE VGTC Symposium Proceedings, Point-Based Graphics (PBG 2005), Stony Brook, USA, pp. 39-46, Eurographics Association.

Ondřej **Došlý** (2005). *Základy konvexní analýzy a optimalizace v* $\mathbb{R}^n$. Munipress - nakladatelství Masarykovy univerzity v Brně, Přírodovědecká fakulta.

Harry **Dym** (2007). *Linear Algebra in Action*. American Mathematical Society, USA.

Herbert **Edelsbrunner** (2001). *Geometry and Topology for Mesh Generation*. Cambridge University Press, United Kingdom.

Herbert **Edelsbrunner,** Ernst P. **Mücke** (1994). *Three-dimensional Alpha Shapes*. Journal, ACM Transactions on Graphics, Volume 13 (1), pp. 43-72, Association for Computing Machinery.

Emicroscribe (2012). Komerční internetové stránky [online], http://www.emicroscribe.com, (březen 2012).

František **Fabian**, Zdeněk **Kluiber** (1998). *Metoda Monte Carlo a možnosti jejího uplatnění*. Prospektum, Praha.

Olivier D. **Faugeras**, Martial **Hebert**, Eric **Pauchon** (1983). *Segmentation of Range Data into Planar and Quadric Patches*. Proceedings of IEEE conference on Computer Vision and Pattern Recognition, Arligton, USA, pp. 8-13, IEEE Computer Society.

James D. **Foley**, Andries **van Dam**, Steven K. **Feiner**, John F. **Hughes** (1995). *Computer Graphics: principles and practice*. Addison-Wesley Publishing Company, USA.

Thomas A. **Foley** (1987). *Interpolation and Approximation of 3-D and 4-D Scattered Data*. Journal, Computers & Mathematics with Applications, Volume 13 (8), pp. 711-740, Elsevier.

Pascal **Fua**, Peter T. **Sander** (1992). *Reconstructing Surfaces from Unstructured 3D Points*. Proceedings Image Understanding Workshop, San Diego, USA, pp. 615-625.

Geometric Tools (2012). *Books, Source Code, and Documentation for Computer Graphics, Mathematics, Physics, Numerical Methods, and Image Analysis* [online], http://www.geometrictools.com/, (červen 2012).

Geovap (2011). Komerční internetové stránky [online], http://www.geovap.cz, (březen 2011).

Gom (2012). *Optical Measuring Techniques*. Komerční internetové stránky [online], http://www.gom.com, (březen 2012).

Günther **Hämmerlin,** Karl-Heinz **Hoffmann** (1991). *Numerical Mathematics*. Springer-Verlag New York, USA.

David A. **Harville** (2008). *Matrix Algebra From a Statistician's Perspective*. Springer Science+Business Media, USA.

Hugues **Hoppe**, Tony **DeRose**, Tom **Duchamp,** John **McDonald**, Werner **Stuetzle** (1992). *Surface Reconstruction from Unorganized Points.* SIGGRAPH '92 Proceedings of the 19th annual conference on Computer graphics and interactive techniques, Chicago, USA, pp. 71-78, Association for Computing Machinery.

Josef **Hoschek**, Dieter **Lasser** (1993). *Fundamentals of Computer Aided Geometric Design*. A K Peters, USA.

Yong-Qing **Cheng**, Victor **Wu**, Robert **Collins**, Allen R. **Hanson**, Edward M. **Riseman** (1996). *Maximum-weight Bipartite Matching Technique and its Application in Image Feature Matching*. Proceedings of the SPIE 2727 Conference on Visual Communication and Image Processing, Orlando, USA, pp. 453-462, SPIE Digital Library.

Katsushi **Ikeuchi**, Daisuke **Miyazaki** (2008). *Digitally Archiving Cultural Objects*. Springer Science+Business Media, USA.

Armin **Iske** (2004). *Multiresolution Methods in Scattered Data Modelling.* Springer-Verlag Berlin Heidelberg, Germany.

Frank **Isselhard**, Guido **Brunnett**, Thomas **Schreiber** (1997). *Polyhedral Reconstruction of 3D Objects by Tetrahedra Removal*. Technical report, Fachbereich Informatik, TU-Kaiserslautern, Germany.

J. Edward **Jackson** (2003). *A User's Guide To Principal Components*. John Wiley & Sons, USA.

Alan **Jeffrey** (2010). *Matrix Operations for Engineers and Scientists: An Essential Guide in Linear Algebra*. Springer Science+Business Media B. V., United Kingdom.

Alexandr **Jeměljanov** (2004). *Surface Reconstruction from Clouds of Points*. Doctoral Thesis, Universiry of West Bohemia in Pilsen, Faculty of Applied Sciences, Czech Republic.

Ian T. **Jolliffe** (2002). *Principal Component Analysis*. Springer-Verlag New York, USA.

Brian Wilson **Kernighan,** Dennis MacAlistair **Ritchie** (1988). *The C Programming Language*. Prentice Hall, USA.

Konica Minolta (2012). Komerční internetové stránky [online], http://www.konicaminolta.com, (březen 2012).

Jiří **Kopáček** (2003). *Matematická analýza pro fyziky*. MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Laboratoř 3D zobrazovacích a analytických metod (2013). Fakultní internetové stránky [online], http://www.natur.cuni.cz/biologie/antropologie/pracoviste-katedry/3dlab, (květen 2013).

Alan J. **Laub** (2005). *Matrix Analysis for Scientists and Engineers*. SIAM – Society for Industrial and Applied Mathematics, USA.

Alan M. **McIvor**, Robert J. **Valkenburg** (1997). *A Comparison of Local Surface Geometry Estimation Methods*. Journal, Machine Vision and Applications, Volume 10 (1), pp. 17-26, Springer-Verlag New York.

Milan **Meloun**, Jiří **Militký** (2004). *Statistická analýza experimentálních dat*. Academia, nakladatelství Akademie věd České republiky, Praha.

Robert **Mencl**, Heinrich **Müller** (1998). *Graph-based Surface Reconstruction Using Structures in Scattered Point Sets*. Proceedings of the Computer Graphics International, Hannover, Germany, pp. 298-311, IEEE Computer Society.

Niloy J. **Mitra**, An **Nguyen** (2003). *Estimating Surface Normals in Noisy Point Cloud Data.* Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, USA, pp. 322-328, Association for Computing Machinery.

Gregory M. **Nielson** (1993). *Scattered Data Modeling*. Journal, IEEE Computer Graphics and Applications - Special issue on computer-aided geometric design, Volume 13 (1), pp. 60-70, IEEE Computer Society Press.

Mark **Pauly**, Markus **Gross**, Leif P. **Kobbelt** (2002). *Efficient Simplification of Point-Sampled Surfaces*. Proceedings of the IEEE Visualization 2002 (VIS 2002), Boston, USA, pp. 163-170, IEEE Computer Society.

Karel **Pavelka** (2006). *Laserové skenování – nová technologie sběru prostorových dat.* Habilitační přednášky, České vysoké učení technické v Praze, Fakulta stavební, Česká republika.

Irena **Pešková** (2011). *Povrchové triangulace*. Bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Česká republika.

Sylvain **Petitjean** (2002). *A Survey of Methods for Recovering Quadrics in Triangle Meshes*. Journal, ACM Computing Surveys, Volume 34 (2), pp. 211-262, Association for Computing Machinery.

David **Poole** (2006). *Linear Algebra: A Modern Introduction*. Thomson Brooks/Cole, Canada.

Helmut **Pottmann**, Andreas **Asperl**, Michael **Hofer**, Axel **Kilian** (2007). *Architectural Geometry*. Bentley Instute Press, USA.

Franco P. **Preparata**, Michael Ian **Shamos** (1985). *Computational Geometry: An Introduction*. Springer-Verlag New York, USA.

William H. **Press**, Saul A. **Teukolsky**, Brian P. **Flannery**, William T. **Vetterling** (1992). *Numerical Recipes in C – The Art of Scientific Computing*. Cambridge University Press.

Petr **Přikryl** (1988). *Numerické metody matematické analýzy*. SNTL – Nakladatelství technické literatury, Praha.

Anthony **Ralston** (1965). *A First Course in Numerical Analysis.* McGraw-Hill Inc., USA.

Gerhard **Roth**, Eko **Wibowoo** (1997). *An Efficient Volumetric Method for Building Closed Triangular Meshes from 3-D Image and Point Data*. Proceedings of the conference on Graphics Interface ‘97, Kelowna, British Columbia, Canada, pp. 173-180, Canadian Information Processing Society Toronto.

Otakar **Rožánek** (2008). *Nové metody dokumentace historického objektu pomocí laserového skenování.* Diplomová práce, České vysoké učení technické v Praze, Fakulta stavební, Česká republika.

Detlef **Ruprecht**, Ralf **Nagel**, Heinrich **Müller** (1995). *Spatial Free Form Deformation with Scattered Data Interpolation Methods*. Journal, Computers & Graphics, Volume 19 (1), pp. 63-71, Elsevier.

Stuart J. **Russell**, Peter **Norvig** (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, USA.

Jitka **Segethová** (1998). *Základy numerické matematiky*. Karolinum – Nakladatelství Univerzity Karlovy v Praze.

Peter **Shirley**, Steve **Marschner** (2009). *Fundamentals of Computer Graphics*. A K Peters, USA.

Simple3D (2006). *3D Scanners, Digitizers, and Software for Making 3D Models and 3D Measurements*. Internetový portál [online], http://www.simple3d.com/, (září 2011).

Pavel **Skoupý** (2007). *3D optické měřící a skenovací systémy pro strojírenství*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Česká republika.

SolidVision (2011). Komerční internetové stránky [online], http://www.solidvision.cz/, (březen 2011).

Petra **Surynková** (2008a). *Plochy stavební praxe*. Diplomová práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Česká republika.

Petra **Surynková** (2008b). *3D modely v programu Rhinoceros*. Sborník příspěvků 28. konference o geometrii a počítačové grafice, Lednice, Česká republika, str. 293-300, Mendelova zemědělská a lesnická univerzita, Brno.

Petra **Surynková** (2009a). *Surface Reconstruction*. Proceedings of the 18th Annual Conference of Doctoral Students (WDS 2009), Prague, Czech Republic, pp. 204-209, MATFYZPRESS, Faculty of Mathematics and Physics, Charles University, Prague.

Petra **Surynková** (2009b). *Vývoj výpočetní geometrie*. Sborník 30. mezinárodní konference Historie matematiky, Jevíčko, Česká republika, str. 217-220, MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Petra **Surynková** (2010a). *Curve and Surface Reconstruction Based on the Method of Evolution*. Proceedings of the 19th Annual Conference of Doctoral Students (WDS 2010), Prague, Czech Republic, pp. 139-144, MATFYZPRESS, Faculty of Mathematics and Physics, Charles University, Prague.

Petra **Surynková** (2010b). *Vývoj lineární perspektivy ve výtvarném umění*. Sborník 31. mezinárodní konference Historie matematiky, Velké Meziříčí, Česká republika, str. 225-230, MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Petra **Surynková** (2010c). *Počítačové modelování*. Sborník celostátní konference Jak připravit učitele matematiky a další texty, Praha, Česká republika, str. 127-139, MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Petra **Surynková** (2011a). *Classical Geometry with GeoGebra*. Proceedings of the Second North American GeoGebra Conference, Toronto, ON, Canada, pp. 77-94, University of Toronto.

Petra **Surynková** (2011b). *3D Modeling in Teaching and Learning Geometry*. Proceedings of the Tenth International Conference on Technology in Mathematics Teaching (ICTMT10), Portsmouth, Great Britain, pp. 279-284, University of Portsmouth.

Petra **Surynková** (2012a). *3D Geometric Modeling*. Proceedings of the International Conference on Communication, Media, Technology and Design (ICCMTD), Istanbul, Turkey, pp. 11-19, Anadolu University, Institute of Communication Sciences.

Petra **Surynková** (2012b). *Analýza bodové množiny*. Sborník příspěvků 32. konference o geometrii a grafice, Železná Ruda – Špičák, Česká republika, str. 209-224, Fakulta aplikovaných věd, Západočeská univerzita v Plzni.

Petra **Surynková**, Radka **Matěková**, Jana **Vlachová** (2012). *Geometrické modelování*. Sborník příspěvků konference Matematika a reálný svět, Praha, Česká republika, str. 82-92, MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Petra **Surynková**, Zbyněk **Šír** (2010). *Shape Fitting and non Convex Data Analysis*. Sborník příspěvků 30. konference o geometrii a grafice, Zlenice, Česká republika, str. 219-229, Matfyzpress, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Petra **Surynková**, Zbyněk **Šír** (2011). *Shape Fitting and non Convex Data Analysis*. Journal**,** South Bohemia Mathematical Letters, Volume 18 (1), pp. 29-37, University of South Bohemia.

Petra **Surynková**, Zbyněk **Šír**, Šárka **Voráčová** (2009). *Statistical Application of Curve Evolution*. Sborník příspěvků 29. konference o geometrii a počítačové grafice, Doubice, Česká republika, str. 325-330, Nakladatelství POLYGLOT, spol. s r. o., Liberec.

Petra **Surynková**, Šárka **Voráčová** (2011). *Digitální rekonstrukce povrchů*. Sborník příspěvků 31. konference o geometrii a grafice, Malá Morávka, Česká republika, str. 233-239, Vysoká škola báňská, Technická univerzita Ostrava.

Emanuele **Trucco**, Robert B. **Fisher** (1995). *Experiments in Curvature-Based Segmentation of Range Data*. Journal, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 17 (2), pp. 177-182, IEEE Computer Society.

Aleš **Urbánek** (2008). *Kontrola součástí pomocí metod reverzního inženýrství*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Česká republika.

Michal **Varnuška** (2002). *Rekonstrukce povrchů geometrických objektů z roztroušených bodů*. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Česká republika.

Remco C. **Veltkamp** (1994). *Closed Object Boundaries from Scattered Points*. Lecture Notes in Computer Science 885, Springer-Verlag Heidelberg, Germany.

Stewart **Venit**, Wayne **Bishop**, Jason Ira **Brown** (2008). *Elementary Linear Algebra*. Nelson Education, Canada.

Šárka **Voráčová,** Petra **Surynková** (2011). *Rekonstrukce kvadratických plátů z mračna bodů*. Sborník příspěvků 31. konference o geometrii a grafice, Malá Morávka, Česká republika, str. 291-298, Vysoká škola báňská, Technická univerzita Ostrava.

Naoufel **Werghi**, Bob **Fisher**, Anthony **Ashbrook**, Craig **Robertson** (2000). *Faithful Recovering of Quadric Surfaces from 3D Range Data*. Journal, International Journal of Shape Modelling, Volume 6 (1), pp. 65-78, World Scientific.

Xin-She **Yang** (2008). *Introduction to Computational Mathematics*. World Scientific. Singapore.

Shuchun **Yu**, Yanhui **Wei**, Kunpeng **He**, Xiaoyang **Yu** (2010). *A New Direct Triangulation Method for Surface Unorganized Points.* Information Technology Journal, Volume 9 (7), pp. 1421-1425, Asian Network for Scientific Information.

Jaromír **Zábranský** (2005). *Triangulace povrchů a úlohy na nich*. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Česká republika.

Karel **Zvára**, Josef **Štěpán** (2001). *Pravděpodobnost a matematická statistika*. MATFYZPRESS, vydavatelství Matematicko-fyzikální fakulty, Univerzita Karlova v Praze.

Jiří **Žára**, Bedřich **Beneš**, Jiří **Sochor**, Petr **Felkel** (2004). *Moderní počítačová grafika*. Computer Press, Česká republika.

Příloha A

Obsah přiloženého výměnného média

Součástí disertační práci je rovněž přiložené výměnné médium (DVD-ROM), na němž se nacházejí další materiály v elektronické podobě. Médium obsahuje text disertační práce, implementace navržených algoritmů v moderním programovacím jazyku a interaktivním prostředí MATLAB a veškerá použitá data. Obsah přiloženého výměnného média je shrnut v následující tabulce A.1.

Obsah přiloženého výměnného média		
Adresář		*Stručný popis*
bodove_mnoziny		Vstupní bodové množiny programů.
	bodove_mnoziny/orto_primka2d	Experimentální data pro ortogonální prokládání dat v rovině přímkou.
	bodove_mnoziny/orto_primka3d	Experimentální data pro ortogonální prokládání dat v prostoru přímkou.
	bodove_mnoziny/osa_rot_valce	Experimentální data pro hledání osy rotační válcové plochy minimalizačními technikami.
	bodove_mnoziny/osa_rovnobezek	Experimentální data pro hledání osy rovnoběžek minimalizačními technikami.
	bodove_mnoziny/osa_rot_plochy	Experimentální data pro hledání osy obecné rotační plochy minimalizačními technikami.
	bodove_mnoziny/povrch	Experimentální data pro povrchové analýzy.
	bodove_mnoziny/povrch/inkrement	Experimentální data pro inkrementální konstrukci povrchové triangulace.
	bodove_mnoziny/povrch/cylindr	Experimentální data pro konstrukci povrchové triangulace pomocí transformace.
	bodove_mnoziny/povrch/symetrie	Experimentální data pro konstrukci povrchové triangulace pomocí rovinových symetrií.
	bodove_mnoziny/realna_data	Reálné bodové množiny získané 3D skenováním povrchů.
	bodove_mnoziny/realna_data/dotykovy_skener	Reálné bodové množiny získané skenováním dotykovým skenerem MicroScribe G2X.
	bodove_mnoziny/realna_data/laserovy_skener	Reálné bodové množiny získané skenováním laserovým skenerem Roland Picza LPX-1200.
	bodove_mnoziny/realna_data/opticky_skener	Reálné bodové množiny získané skenováním optickým systémem Vectra3D.

programy		Programy naimplementované ve výpočetním prostředí MATLAB (.m) a bodové množiny, pro které jsou upraveny vstupní parametry (.txt).
	programy/mnc_2d	Prokládání dat v rovině přímkou a polynomem libovolného stupně metodou nejmenších čtverců.
	programy/orto_primka2d	Ortogonální prokládání dat v rovině přímkou.
	programy/orto_primka3d	Ortogonální prokládání dat v prostoru přímkou.
	programy/mnc3d	Prokládání dat v prostoru rovinou a polynomem dvou proměnných libovolného stupně metodou nejmenších čtverců.
	programy/orto_rovina	Ortogonální prokládání dat v prostoru rovinou.
	programy/pca	Analýza hlavních komponent v rovině a v prostoru.
	programy/minimalizace	Hledání minima funkce více proměnných.
	programy/osa_rot_valce	Hledání osy rotační válcové plochy.
	programy/osa_rovnobezek	Hledání osy rovnoběžek.
	programy/osa_rot_plochy	Hledání osy obecné rotační plochy.
	programy/symetrie	Hledání rovin symetrií bodové množiny.
	programy/tecne_roviny	Odhady tečných rovin a normál bodové množiny.
	programy/povrch	Povrchové analýzy.
	programy/povrch/inkrement	Inkrementální konstrukce povrchové triangulace.
	programy/povrch/cylindr	Konstrukce povrchové triangulace pomocí transformace.
	programy/povrch/symetrie	Konstrukce povrchové triangulace pomocí rovinových symetrií.
text		Text disertační práce ve formátu pdf.

Tabulka A.1: Obsah přiloženého výměnného média