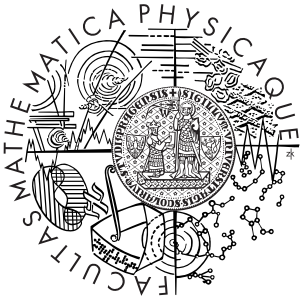
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Peter Majer

Tvorba programu pro zpracování dat tepelné kapacity

Katedra fyziky kondenzovaných látek

Vedoucí bakalářské práce: doc. Mgr. Pavel Javorský, Dr.

Studijní program: Informatika, obecná informatika

2009

Moje poďakovanie patrí predovšetkým vedúcemu bakalárskej práce za jeho trpezlivosť a rady.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 11.12.2009 Peter Majer

Obsah

Název práce: Tvorba programu pro zpracování dat tepelné kapacity
Autor: Peter Majer
Katedra (ústav): Katedra fyziky kondenzovaných látek
Vedoucí bakalářské práce: doc. Mgr. Pavel Javorský, Dr.
e-mail vedoucího: javor@mag.mff.cuni.cz

Abstrakt: Práca popisuje návrh a riešenie programu určeného na spracovanie dát, získaných meraním tepelnej kapacity skúmaných vzoriek. Program má dve časti: spracovanie dát a vykresľovanie grafov. Časť spracúvajúca dáta je vzhľadovo podobná tabuľkovému procesoru - každý stĺpec v tabuľke reprezentuje jedno meranie. Zo stĺpcov reprezentujúcich meranie teplôt je možné spočítať tepelnú kapacitu podľa niekoľkých modelov (uvažujúcich: merné teplo vodivostných elektrónov, fonónový príspevok, Schottkyho príspevok), príp. zo základných matematických operácií zostaviť vlastný výpočet. Spätne je možné fitovať namerané tepelné kapacity na zvolený model. Na nelineárne fitovanie sa používajú simplexová a Marquardtova metóda. Súčasťou programu je aj časť počítajúca Hamiltonián využiteľný pri výpočte energetických hladín. Program je napísaný v jazyku C#.

Klíčová slova: tepelná kapacita, nelineárne fitovanie, C#

Title: Generation of computer program for specific-heat analysis
Author: Peter Majer
Department: Department of Condensed Matter Physics
Supervisor: doc. Mgr. Pavel Javorský, Dr.
Supervisor's e-mail address: javor@mag.mff.cuni.cz

Abstract: This work is about to design and implement the specific-heat analysis application. The application consists of two parts: the first one provides for data manipulation, the second one plots a graph. The data manipulation part resembles a spreadsheet program - each column in a dataset stands for one measurement. Columns containing a temperature data could be used for a specific-heat calculation. This calculation is based on this model functions: conduction electrons consideration, phonon contribution and Schottky contribution. It is also possible to arrange your own formulas by using basic mathematical operations (namely summation, substraction, multiplication and division). Fitting data to the selected model is also possible. Levenberg-Marquardt and simplex method are being used for it. Hamiltonian matrix used for energy levels calculations of magnetic ions in crystalline electric fields could also be calculated. The application is written in C# programming language.

Keywords: specific heat, non-linear fitting, C#

Kapitola 1

Úvod

Cieľom tejto bakalárskej práce bolo vytvoriť použiteľný počítačový program umožňujúci analyzovať dáta experimentov zameraných na skúmanie merného tepla vzoriek.

Základné požadované vlastnosti programu boli

- schopnosť načítať dáta z jednoduchých textových súborov, ktoré sú výstupom experimentálnych aparatúr
- možnosť spravovať dáta a vytvárať množiny vlastných dát
- jednoduché matematické operácie nad dátami
- vynesenie dát do grafu
- výpočet predpokladaného merného tepla na základe zadanej teploty podľa odhadovaných typov príspevkov a ich veľkostí
- fitovanie počítanej krivky k nameraným dátam – teda na základe zmeraných údajov (teplota a merné teplo) určiť veľkosť jednotlivých fyzikálnych príspevkov predpokladaného fyzikálneho modelu
- uloženie rozpracovanej úlohy na disk a jej opätovné načítanie

V tejto práci som sa pokúsil program na základe predložených požiadaviek vytvoriť, pričom súčasťou návrhu bol aj výber vhodných algoritmov nelineárneho fitovania.

Kapitola 2

Fyzikálne pozadie

V teplotnej závislosti merného tepla sa odrážajú prakticky všetky deje v danej látke. Do celkovej hodnoty zistenej experimentom tak prispieva nezávisle niekoľko dejov, napr. merné teplo kmitov kryštálovej mriežky, merné teplo fázových prechodov (supravodiče, magnetické usporiadanie, zmeny kryštálovej štruktúry, ...), merné teplo vodivostných elektrónov a ďalšie. Jednotlivé príspevky sú aditívne a sú obvykle popísané v rámci príslušnej teórie, celková analýza je však pomerne komplikovaná.

Príspevky merného tepla počítané vypracovaným programom sú

- príspevok kmitov mriežky
- merné teplo vodivostných elektrónov
- magnetický príspevok spôsobený postupných obsadzovaním energetických hladín so zvyšujúcou sa teplotou (tzv. Schottkyho príspevok)

Príspevok kmitov kryštálovej mriežky

Tento príspevok, tiež nazývaný fonónový príspevok, je v kryštalických pevných látkach dominantný nad $\sim 10\,\mathrm{K}$. Jeho analýza je preto veľmi dôležitá aj pre vyhodnocovanie ostatných príspevkov. Všeobecne je vibračné spektrum v 3-dimenzionálnej kryštalickej pevnej látke tvorené 3 akustickými a $3(p-1)$ optickými vetvami, pričom p je počet atómov v primitívnej bunke. Správanie akustických kmitov je pomerne dobre popísané v rámci Debyeovho modelu, kde predpokladáme lineárnu závislosť frekvencie kmitov na veľkosti recipročného vektora. Merné teplo je v rámci tohto modelu dané vzťahom [2]

$$C_p = 9k_B N_A \left(T/\theta_D\right)^3 \int_0^{\theta_D/T} \frac{x^4 \mathrm{e}^x}{\left(\mathrm{e}^x - 1\right)^2}\,\mathrm{d}x$$

k_B je Boltzmannova konštanta, N_A je Avogadrova konštanta, T je teplota a θ_D je tzv. Debeyeho teplota, ktorá je určitou charakteristikou kmitov mriežky a predstavuje jediný variabilný parameter.

Optické kmity sú zas dobre popísané v rámci jednoduchšieho Einstenovho modelu, v ktorom predpokladáme frekvenciu kmitov nezávislú na vlnovom vektore. Merné teplo je v rámci tohto modelu dané vzťahom

$$C_p = k_B N_A \sum_i \left(\theta_{E_i}/T\right)^2 \frac{\mathrm{e}^{\theta_{E_i}/T}}{\left(\mathrm{e}^{\theta_{E_i}/T} - 1\right)^2}$$

θ_E je tzv. Einsteinova teplota, ktorá je opäť určitou charakteristikou kmitov mriežky, konkrétne optických vetiev. Podľa počtu vetiev dostávame príslušný počet variabilných parametrov. Kvôli zjednodušeniu modelu je často výhodné popísať viacej vetiev jediným parametrom θ_E.

Merné teplo vodivostných elektrónov

Merné teplo vodivostných elektrónov je spravidla malé, v niektorých látkach však môže najmä pri nízkych teplotách nadobúdať význam a jeho veľkosť je významnou charakteristikou elektrónových vlastností daného materiálu. Závisí lineárne na teplote s koeficientom úmernosti γ (tzv. Sommerfeldov koeficient) [2]

$$C_{el} = \frac{1}{3}\pi^2 n(E_F) k_B^2 T = \gamma T$$

$n(E_F)$ je hustota elektrónových stavov na Fermiho medzi. Pre väčšinu látok je parameter γ teplotne nezávislý a je ho možné pomerne dobre zistiť analýzou nízkoteplotných dát merného tepla. Problematická môže byť analýza u tzv. ťažkofermiónových zlúčenín, kde γ teplotne závislé je.

Schottkyho príspevok

Tento príspevok sa objavuje pri zlúčeninách obsahujúcich najmä ióny vzácnych zemín. Základný elektrónový stav týchto iónov, tzv. multiplet, je pod vplyvom ostatných iónov v kryštálovom okolí rozštiepený na viacero energetických hladín. Ich počet závisí na type daného iónu a veľkosti rozštiepenia. Prípadná degenerácia jednotlivých energetických hladín závisí ešte aj na symetrii kryštálového okolia. Ak poznáme energetické vzdialenosti medzi jednotlivými hladinami, potom entropia (či merné teplo) je daná postupnou zmenou pravdepodobnosti obsadzovania týchto hladín s premennou teplotou.

Pre merné teplo je možné z termodynamiky odvodiť vzťah

$$C_{Sch} = -\frac{\partial^2 \left(k_B T \ln Z\right)}{\partial T^2} = \frac{k_B}{T^2}\left[\sum_i \frac{E_i^2 \mathrm{e}^{-E_i/T}}{Z} - \left(\sum_i \frac{E_i \mathrm{e}^{-E_i/T}}{Z}\right)^2\right]$$

E_i sú jednotlivé energetické hladiny a Z je štatistická suma

$$Z = \sum_i \mathrm{e}^{-E_i/T}$$

K tomuto príspevku sa viaže aj výpočet energetických hladín na základe znalosti symetrie kryštálovej štruktúry a parametrov charakterizujúcich vplyv kryštálového okolia. Vo fyzike pevných látok je tento vplyv popísaný Hamiltoniánom

$$\widehat{H}_{CF} = \sum_{n,m} A_n^m \langle r^n \rangle \, \theta_n \widehat{O}_n^m = \sum_{n,m} B_n^m \widehat{O}_n^m$$

A_n^m, resp. B_n^m sú parametre kryštálového poľa. Počet nezávislých parametrov B_n^m je daný symetriou štruktúry a pohybuje sa medzi 2 (kubická symetria) až 9 (nízke symetrie, napr. ortorombická). $\widehat{O}_n^m$ sú tabelované operátorové ekvivalenty [1], θ_n sú číselné faktory tabelované pre dané ióny vzácnych zemín.

Veľkosť rozštiepenia energetických hladín môže byť ovplyvnená aj ďalšími vonkajšími vplyvmi, napr. pôsobením vonkajšieho magnetického poľa. To je možné započítať do uvedeného Hamiltoniánu, ktorého výpočet je súčasťou vypracovaného programu.

Kapitola 3

Požadované výpočty

Z predložených požiadaviek vyplynula nutnosť implementovať výpočty, ktoré neposkytuje štandardná knižnica tried `Math .NET Framework`. Komplikovanejšie výpočty zahŕňali integráciu krivky v určenom intervale a najmä fitovanie krivky k dátam.

3.1 Integrovanie

Presnou a asi najjednoduchšou metódou integrácie danej funkcie, je výpočet rozdielu jej primitívnej funkcie v zadaných okrajoch integračného intervalu. Tento postup (*Newtonov integrál*) ale predpokladá existenciu primitívnej funkcie a jej nájdenie (analytický výpočet integrálu) nie je vždy jednoduché, či možné.

Existuje ale rad numerických integračných metód. Základným typom sú metódy deliace integračný interval na menšie časti a odhadujúce celkový integrál ako sumu integrálov týchto častí [6].

Nech je interval rozdelený bodmi x_i, pričom platí $x_i = x_0 + ih$, kde h je veľkosť každého podintervalu.

Lichobežníková metóda

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x = h\left[\frac{1}{2}f(x_0) + \frac{1}{2}f(x_1)\right] + O\left(h^3 f''\right)$$

f'' je druhá derivácia f vo vhodnom mieste intervalu, cez ktorý sa integruje. Táto metóda dáva presný výsledok pre polynómy do 1. stupňa. Odhad je možné v grafe interpretovať ako výpočet obsahu plochy pod spojnicou funkčných hodnôt krajných bodov intervalu – výpočet obsahu lichobežníka. Odhad je teda počítaný z dvoch hodnôt. Ďalšia metóda využíva hodnoty tri.

Simpsonova metóda

$$\int_{x_0}^{x_2} f(x)\,\mathrm{d}x = h\left[\frac{1}{3}f(x_0) + \frac{4}{3}f(x_1) + \frac{1}{3}f(x_2)\right] + O\left(h^5 f^{(4)}\right)$$

$f^{(4)}$ je 4. derivácia f vo vhodnom mieste intervalu. Počítaná je plocha pod parabolou. Táto metóda dáva vďaka vyrušeniu koeficientov (ľavo-pravá symetria) presný výsledok pre polynómy až do 3. stupňa.

Podobne päťbodová metóda (tzv. *Bodeho*) je presná pre polynómy do 5. stupňa.

Rozšírená lichobežníková metóda

Ak lichobežníkovú metódu aplikujeme na jednotlivé intervaly

$$\langle x_0, x_1\rangle \text{ až } \langle x_{N-1}, x_N\rangle$$

a výsledky sčítame, dostaneme tzv. rozšírené lichobežníkové pravidlo

$$\int_{x_0}^{x_N} f(x)\,\mathrm{d}x = h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N)\right] + O\left(\frac{(x_N - x_0)^3 f''}{N^2}\right)$$

Odhad chyby je v takomto zápise závislý na N a nie na h, čo je pre výpočty pohodlnejšie – vo zvolenom intervale $\langle x_0, x_N\rangle$ sa z požadovanej chyby určí počet deliacich bodov a dopočíta h.

Rombergova metóda

Táto metóda opakovane aplikuje tzv. *Richardsonovu extrapoláciu* na pravidlo lichobežníkovej metódy. Richardsonova extrapolácia je uskutočňovanie zvoleného výpočtu pre rôzne parametre h a následná extrapolácia výsledkov pre $h = 0$. Algoritmus končí, ak je odhadovaná chyba extrapolácie menšia než požadovaná hodnota.

```
h[0] ← 1
for i ← 1 to steps do
  s[i − 1] ← trapez(function, a, b, i)
  if i ≥ points then
    for j ← 0 to points − 1 do
      h_t[j] ← h[i − steps + j]
      s_t[j] ← s[i − steps + j]
    end for
    extrapol(h_t, s_t, 0, out ss, out dss)
    if |dss| ≤ EPS ∗ |ss| then
      return ss
    end if
  end if
  h[i] ← 0,25 ∗ h[i − 1]
end for
```

function je integrovaná funkcia, *steps* je maximálny počet iterácií, *points* je počet bodov v extrapolácii (napr. pre *points* = 2 je výsledkom Simpsonova metóda), *EPS* je požadovaná čiastočná presnosť. Press a kol. [6] volia *points* = 5 a $EPS = 10^{-10}$.

Funkcia `trapez` počíta integrál rozšírenou lichobežníkovou metódou a to pre zadanú funkciu, interval a počet delení. Procedúra `extrapol` polynomicky extrapoluje funkciu $S(h) = \int_{x_0}^{x_N} f(x)\,\mathrm{d}x$ v bode $h = 0$. Pokles h je zámerne volený dvojnásobne oproti skutočnému zvyšovaniu počtu deliacich bodov, vďaka čomu je extrapolácia polynómom nielen v h ale aj v h^2 [6]. `extrapol` vracia extrapoláciu v danom bode a jej chybu.

Výpočet integrálu v Debyeovom modeli je vo vypracovanej aplikácii vykonávaný pomocou tejto metódy.

3.2 Lineárna regresia

Najpoužívanejšou metódou regresnej analýzy je *metóda najmenších štvorcov*. Táto metóda aproximuje dané hodnoty polynómom stupňa 1 – priamkou. Miera súhlasu modelu s dátami je vyjadrená sumou štvorcov odchýlok modelu od dát. Model s najmenšou sumou štvorcov odchýlok je hľadaný model, odtiaľ metóda najmenších štvorcov.

Pre množinu dát s n prvkami (x_i, y_i) hľadáme model tvaru

$$y(x) = ax + b$$

Ohodnocovacia funkcia má tvar

$$\chi^2(a, b) = \sum_{i=1}^{n} \left(\frac{y_i - ax_i - b}{\sigma_i} \right)^2$$

kde σ_i je chyba určenia hodnoty y_i [6].

Keďže parametre hľadaného modelu sú súradnice minima χ^2, postačí toto minimum určiť položením derivácií χ^2 rovným nule

$$0 = \frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{n} \frac{x_i \left(y_i - ax_i - b \right)}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{n} \frac{y_i - ax_i - b}{\sigma_i^2}$$

Týmto spôsobom je v programe na základe vložených dát určovaný Sommerfeldov koeficient γ.

3.3 Nelineárne regresné modely

Algoritmy nelineárnej regresie sa delia na niekoľko základných skupín [3]

- nederivačné optimalizačné metódy
- derivačné metódy pre kritérium metódy najmenších štvorcov

- všeobecné derivačné metódy
- algoritmy pre špeciálne postupy

V programe sú použité dve najznámejšie metódy, každá z jednej z prvých dvoch skupín.

Simplexové metódy

Algoritmy tohto typu patria medzi nederivačné optimalizačné metódy, ktoré sú v praxi obľúbené najmä vďaka svojej jednoduchosti. Všeobecne sa algoritmy z tejto skupiny odporúča používať vtedy, ak nie je kriteriálna podmienka regresie diferencovateľná alebo nie je diferencovateľná regresná funkcia.

Simplexové metódy vytvárajú adaptívne polyédre v priestore kombinácií možných parametrov modelu. Algoritmus [4] pozostáva z troch fáz

1. konštrukcia prvotného simplexu
2. iteratívny postup k minimu
3. identifikácia ukončenia hľadania

Konštrukcia prvotného simplexu

Súradnice vrcholov polyédra (simplexu) môžeme zapísať do matice s rozmermi $(m+1) \times m$, kde m je počet hľadaných parametrov, pričom každý riadok obsahuje súradnice jedného vrcholu. Ak na začiatku zvolíme pravidelný simplex s dĺžkou hrany t a jedným z vrcholov v počiatku súradníc, matica bude vyzerať takto

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ c & a & a & \cdots & a \\ a & c & a & \cdots & a \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a & a & a & \cdots & c \end{bmatrix}$$

$$a = \frac{t\left(\sqrt{m+1}-1\right)}{m\sqrt{2}} \qquad c = \frac{t\,(m-1)+\sqrt{m+1}}{m\sqrt{2}}$$

V prípade zadaných počiatočných parametrov je matica konštruovaná tak, že jej prvý riadok obsahuje tento počiatočný odhad. Prvky riadka i sa vypočítajú na základe

$$P_{ij} = \beta_j^{(0)} + \frac{\beta_j^{(0)}}{m\,2\sqrt{2}}\left(m-1+\sqrt{m+1}\right), \qquad i \neq j$$

$$P_{ij} = \beta_j^{(0)} + \frac{\beta_j^{(0)}}{m\,2\sqrt{2}}\left(\sqrt{m+1}-1\right), \qquad i = j$$

$\vec{\beta}^{(0)}$ je vektor parametrov počiatočného odhadu [3], čo je implementované aj vo vypracovanej aplikácii. $\vec{\beta}^{(0)}$ je ale možné umiestniť aj do ťažiska počiatočného simplexu, či simplex natočiť do smeru najväčšieho gradientu ohodnocovacej funkcie.

Iteračný postup

Každý krok iteračného hľadania vyberá medzi troma základnými operáciami: *reflexia*, *kontrakcia* a *expanzia*.

Reflexia sa vykonáva na začiatku každej iterácie. Vychádza z predpokladu, že smer k minimu leží na spojnici vrcholu najhoršieho odhadu ($\vec{P}_h$) a ťažiska simplexu ostatných vrcholov ($\vec{C}$). Pre nový odhad $\vec{P}^*$ získaný reflexiou a vrcholy pôvodného simplexu $\vec{P}_i$ platí

$$\vec{P}^* = \vec{C} + \alpha\left(\vec{C} - \vec{P}_h\right) \qquad \alpha > 0$$

$$\vec{C} = \sum_{i \neq h}^{m} \frac{\vec{P}_i}{m}$$

α je reflexný koeficient. Pre $\alpha = 1$ je nový odhad priemetom $\vec{P}_h$ cez $\vec{C}$.

Označme ohodnocovaciu funkciu ako y a nech $y_i = y\left(\vec{P}_i\right)$. Ak je nový odhad najlepším odhadom, teda $\forall i : y\left(\vec{P}^*\right) < y_i$, tak pokračujeme v hľadaní odhadu vo zvolenom smere expanziou.

$$\vec{P}^{**} = \vec{C} + \gamma\left(\vec{P}^* - \vec{C}\right) \qquad \gamma > 1$$

γ je expanzný koeficient. Vyjadruje veľkosť posunu – násobok posunu predtým získaného pri reflexii. Ak je $y\left(\vec{P}^{**}\right) < y\left(\vec{P}^*\right)$, potom $\vec{P}_h \leftarrow \vec{P}^{**}$. Ak nie, tak $\vec{P}_h \leftarrow \vec{P}^*$. Následne pokračujeme ďalšou iteráciou.

Ak $\vec{P}^*$ najlepším odhadom nie je, ale existuje aspoň nejaký horší odhad (okrem $\vec{P}_h$), čiže $\exists\,(i \neq h) : y\left(\vec{P}^*\right) < y_i$, potom tiež položíme $\vec{P}_h \leftarrow \vec{P}^*$ a ukončíme prebiehajúcu iteráciu.

Poslednou možnosťou je, že nový odhad je najhorším odhadom. Ak je ale aspoň zlepšením $\vec{P}_h$, potom ešte pred uskutočnením kontrakcie priradíme $\vec{P}_h \leftarrow \vec{P}^*$.

$$\vec{P}^{**} = \vec{C} + \beta\left(\vec{P}_h - \vec{C}\right) \qquad 0 < \beta < 1$$

β je kontrakčný koeficient. Nový odhad teda hľadáme v opačnom smere - približovaním k $\vec{C}$. Vzdialenosť $\vec{P}^{**}$ od $\vec{C}$ bude β-násobkom vzdialenosti $\vec{P}_h$ od $\vec{C}$. Ak sme kontrakciou najhorší odhad zlepšili, tak $\vec{P}_h \leftarrow \vec{P}^{**}$ a pokračujeme ďalšou iteráciou. V prípade neúspechu celý simplex pred ďalšou interáciou zmenšíme a to tak, že vzdialenosť každého vrcholu od vrcholu najlepšieho odhadu ($\vec{P}_l$) zmenšíme na λ-násobok

$$\vec{P}_i \leftarrow \vec{P}_l + \lambda\left(\vec{P}_i - \vec{P}_l\right) \qquad 0 < \lambda < 1$$

Meloun a Militký [3] odporúčajú voliť koeficienty nasledovne: $\alpha = 1$, $\beta = 0{,}55$, $\gamma = 2{,}9$ a $\lambda = 0{,}5$. Rovnaké koeficienty sú použité aj vo vyhotovenom programe.

Ukončenie hľadania

Jedným z používaných testov ukončenia algoritmu je $|y_h - y_l| < \epsilon$. Možné je ale aj porovnanie [4]

$$\sqrt{\sum_i \frac{\left(y_i - y\left(\vec{C}\right)\right)^2}{m+1}} < \epsilon$$

Odporúča sa voliť $\epsilon = 10^{-4}$ v prvom, resp. $\epsilon = 10^{-8}$ v druhom prípade [3].

Derivačné metódy

Derivačné metódy pre kritérium najmenších štvorcov patria medzi najpoužívanejšie [3]. Tieto metódy sú použiteľné pre všetky aspoň dvakrát diferencovateľné modelové funkcie. Ich nevýhodou však je lokálna konvergencia, ktorá závisí na voľbe počiatočného odhadu.

Jedná sa o iteračné algoritmy, kedy sa v i-tej iterácii vychádza z odhadu parametrov $\vec{\beta}^{(i)}$, ku ktorému sa priratúva vhodný prírastok $\vec{\Delta}_i$

$$\vec{\beta}^{(i+1)} = \vec{\beta}^{(i)} + \vec{\Delta}_i$$

Postup sa delí na štyri fázy [3]

1. stanovenie prvých odhadov $\vec{\beta}^{(0)}$
2. nájdenie vhodného smerového vektora $\vec{V}_i$
3. určenie α_i tak, aby bol prírastok $\vec{\Delta}_i = \alpha_i \vec{V}_i$ prijateľný
4. test dosiahnutia minima

Stanovenie prvotných odhadov

Stanovenie odhadov je rozhodujúce pre celý proces. Z veľmi zlých odhadov nenájde minimum žiadna z metód tejto skupiny, príp. sa jedná len o lokálne minimum. Naopak z dobrých odhadov väčšinou konvergujú aj menej náročné algoritmy.

Ak sú regresné modely linearizovateľné, je možné počiatočné parametre určiť ich linearizáciou a následnou aplikáciu lineárnej metódy najmenších štvorcov.

Pri interaktívnej práci s počítačom je zas možné ľahko porovnať priebeh modelovej funkcie $f\left(x, \vec{\beta}^{(0)}\right)$ so zadanými dátami a overovať tak kvalitu prvého odhadu [3], na čo sa spolieha aj metóda implementovaná v programe.

Nájdenie smerového vektora

Majme množinu dát (x_i, y_i) a ohodnocovaciu funkciu

$$U\left(\vec{\beta}\right) = \left\|\vec{y} - \vec{f}\right\|^2$$

$\vec{y} = (y_1, \ldots, y_n)^T$, $\vec{f} = \left(f\left(x_1, \vec{\beta}\right), \ldots, f\left(x_n, \vec{\beta}\right)\right)$ a $\|\vec{v}\| = \sqrt{\vec{v}^T \vec{v}}$ je euklidovská norma.

Určíme deriváciu U v mieste $\vec{\Delta} = \vec{\beta} + \alpha\vec{V}$ podľa α.

$$\frac{\partial U\left(\vec{\beta}\right)}{\partial \alpha} = \left(\frac{\partial U\left(\vec{\beta}\right)}{\partial \vec{\beta}}\right)^T \frac{\partial \vec{\beta}}{\partial \alpha} = \vec{g}^{\,T}\left(\vec{\beta}\right)\vec{V}$$

pričom gradient $\vec{g}$ je uvažovaný v mieste $\vec{\Delta}$. Pre $\alpha \to 0$ získame smerovú deriváciu S_D

$$S_D = \left.\frac{\partial U\left(\vec{\beta}\right)}{\partial \alpha}\right|_{\alpha\to 0} = \vec{g}^{\,T}\vec{V}$$

$$\vec{g} = -2\,\mathbf{J}^T\vec{e} \qquad J_{ij} = \frac{\partial f\left(x_i, \vec{\beta}\right)}{\partial \beta_j} \qquad e_i = y_i - f\left(x_i, \vec{\beta}\right)$$

$\mathbf{J}$ je Jakobiho matica, $\vec{e}$ je vektor diferencií.

Ohodnocovacia funkcia klesá najpríkrejšie v smere $-\vec{g}$. Ak je smerový vektor $\vec{V}$ prijateľný, tak musí existovať nejaká pozitívne definitná matica $\mathbf{R}$ taká, že

$$\vec{V} = -\mathbf{R}\vec{g}$$

Určenie veľkosti prírastku vektora

Optimálna veľkosť prírastku zvoleného smeru sa určí z Taylorovho rozvoja druhého stupňa funkcie $U\left(\vec{\beta}\right)$

$$U\left(\vec{\beta} + \alpha\vec{V}\right) = U\left(\vec{\beta}\right) + \alpha\vec{g}^{\,T}\vec{V} + \frac{\alpha^2}{2}\vec{V}^T\mathbf{H}\vec{V}$$

$$\mathbf{H} = 2\left[\mathbf{J}^T\mathbf{J} + \mathbf{B}\right] \qquad B_{j,k} = \sum_{i=1}^{n} e_i \frac{\partial^2 f\left(x_i, \vec{\beta}\right)}{\partial \beta_j\, \partial \beta_k}$$

$\mathbf{H}$ je Hessova matica. Tento Taylorov rozvoj je vzhľadom k α približne kvadratický, čiže optimálne α sa dá stanoviť položením derivácie $U\left(\vec{\beta} + \alpha\vec{V}\right)$ podľa α rovnej 0. Dostaneme

$$\alpha^* = \frac{-\frac{\partial U\left(\vec{\beta}\right)}{\partial \alpha}}{\frac{\partial^2 U\left(\vec{\beta}\right)}{\partial \alpha^2}} = -\vec{g}^{\,T}\vec{V}\left(\vec{V}^T\mathbf{H}\vec{V}\right)^{-1}$$

$$\alpha^* = \vec{g}^{\,T}\mathbf{R}\vec{g}\left(\vec{g}^{\,T}\mathbf{R}^T\mathbf{H}\mathbf{R}\vec{g}\right)^{-1}$$

α^* je tzv. *Raleighov koeficient* a jeho použiteľnosť je obmedzená na oblasť aproximácie uvedeným Taylorovým rozvojom 2. stupňa.

Vektor $\vec{\Delta}_i$, spočítaný na konci každej iterácie, sa obyčajne považuje za prijateľný, ak

$$U\left(\vec{\beta}^{(i)} + \vec{\Delta}_i\right) < U\left(\vec{\beta}^{(i)}\right)$$

Test dosiahnutia optima

Prirodzeným kritériom pre optimum je nulový gradient $\vec{g}$. Rozšírenou terminačnou podmienkou teda je

$$\epsilon > \|\vec{g}\|^2 = \sum_{i=1}^{m} {g_i}^2$$

Ďalšou možnosťou je algoritmus ukončiť ak dochádza k príliš malým zmenám veľkosti odhadu.

Marquardtova metóda

Marquardtova metóda patrí medzi tzv. *hybridné metódy*. Kombinuje totiž postup *gradientných* a *newtonových* metód.

Gradientné metódy volia smerový vektor rovný najväčšiemu spádu $-\vec{g}$, teda maticu $\mathbf{R}$ volia ako jednotkovú. Pre α^* dostaneme

$$\alpha^* = \vec{g}^T\vec{g}\left[\vec{g}^T\mathbf{H}\vec{g}\right]^{-1} = \vec{g}^T\vec{g}\left[\vec{g}^T\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\vec{g}\right]^{-1}$$

Aj ďaleko od optima sú tieto metódy schopné nájsť smer k minimu. Bohužiaľ ale v blízkosti optima konvergujú pomaly.

Newtonove metódy volia už uvádzaný Taylorov rozvoj pre $\alpha = 1$. Optimálny smerový vektor i-tej iterácie dostaneme pri $\partial U\left(\vec{\beta} + \vec{V}\right) / \partial \vec{V} = 0$ ako

$$\vec{V_i} = -\mathbf{H}^{-1}\vec{g} = \left(\mathbf{J}^T\mathbf{J} + \mathbf{B}\right)^{-1}\mathbf{J}^T\vec{e}$$

Ak bude U kvadratickou funkciou – eliptický paraboloid, metóda nájde optimum v jedinom kroku. Pre iné funkcie alebo pre vzdialené odhady však konverguje pomaly a navyše vyžaduje znalosť matice druhých derivácií (určenie matice $\mathbf{B}$).

Marquardtova metóda volí smerový vektor ako

$$\vec{V_i}(\lambda) = \left(\mathbf{J}^T\mathbf{J} + \lambda{\mathbf{D}_i}^T\mathbf{D}_i\right)^{-1}\mathbf{J}^T\vec{e}$$

$\mathbf{D}_i$ je diagonálna matica eliminujúca rôzne veľkosti zložiek $\mathbf{J}$ – obyčajne s prvkami na diagonále rovnými diagonálnym prvkom $\mathbf{J}^T\mathbf{J}$. λ je parameter metódy. Jeho vhodná voľba zaisťuje

- pozitívnu definitnosť $\mathbf{R} = \mathbf{J}^T\mathbf{J} + \lambda\mathbf{D}^T\mathbf{D}$, ktorá zaručí, že existuje $\mathbf{R}^{-1}$
- skracovanie vektora $\vec{V_i}$ pri vzďaľovaní sa od optima
- možnosť výberu medzi aproximáciou kvadratúrou a gradientným smerom (aj keď obmedzená, keďže s väčšou vzdialenosťou – a teda aj smerom približujúcim sa $-\vec{g}$, sa dĺžka $\vec{V_i}$ znižuje)

- obmedzenie dĺžky vektora $\vec{V}_i$ na istú oblasť v okolí $\vec{\beta}^{(i)}$, čo umožňuje zanedbať maticu druhých derivácií $\mathbf{B}$ Newtonovej metódy, čo je ekvivalentné linearizácii regresného modelu

Nevýhodami sú nutnosť pri každej iterácii počítať $\mathbf{R}^{-1}$, pričom pri veľkých λ sú $\vec{V}_i$ príliš malé.

Pôvodný algoritmus pred každou ďalšou iteráciou pri zmenšení U upraví parameter $\lambda \leftarrow \lambda/10$, pričom sa na lepší – teda s menším súčtom štvorcov chýb, upraví aj odhad $\vec{\beta}^{(i)}$. Naopak pri zhoršení U, resp. nevýraznom zlepšení, sa volí $\lambda \leftarrow 10\lambda$, pričom staré parametre ostávajú v platnosti.

Press a kol. [6] odporúčajú začať s $\lambda = 10^{-3}$ a zvýšiť čítač zlých krokov pri každom zhoršení odhadu, resp. nedostatočnom zlepšení. Konkrétne ak

$$\left|U\left(\vec{\beta}^{(i)} + \vec{\Delta}_i\right) - U\left(\vec{\beta}^{(i)}\right)\right| < \epsilon \qquad \epsilon = 0{,}1$$

V prípade zlepšenia sa čítač vynuluje. Ak nasledujú 4 málo úspešné kroky po sebe, tak algoritmus končí [5].

Kapitola 4

Implementácia

4.1 Použité nástroje

Kvôli ľahšej prenositeľnosti a aj vďaka predchádzajúcim skúsenostiam, som sa rozhodol napísať požadovaný program v jazyku `C#`.

Vývojové prostredie som zvolil aktuálne a obľúbené v čase zadania práce – Microsoft Visual Studio 2005 pre platformu `Microsoft .NET Framework 2.0`.

4.2 Užívateľské grafické rozhranie

Požiadavkou bola grafická aplikácia. Základným konceptom sa nakoniec stal program poskytujúci dve základné funkcie

- jednoduchý tabuľkový procesor
- zobrazovanie zvolených údajov v grafe

Tieto funkcie sú implementované ako dve záložky hlavného okna.

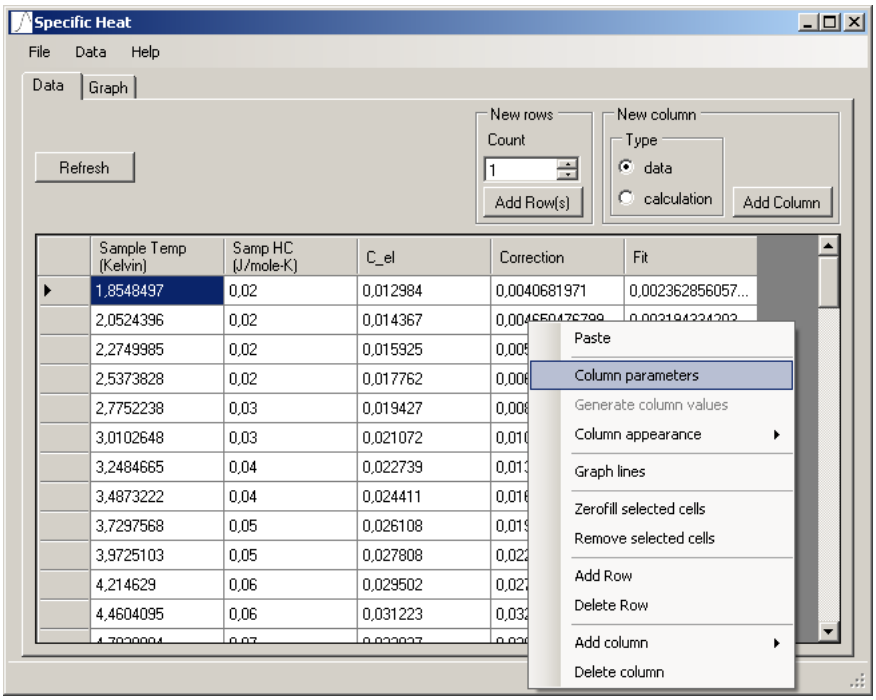
Tabuľkový procesor

V tejto časti užívateľ zadáva dáta. Do tabuľky môžu byť pridávané dva druhy stĺpcov: dátové (*data*) alebo výpočtové (*calculation*).

Dátové stĺpce sú editovateľné a program nemanipuluje s údajmi, ktoré obsahujú.

Údaje vo výpočtových stĺpcoch naopak nie sú priamo editovateľné. Užívateľ len špecifikuje druh výpočtu, jeho parametre, ako aj stĺpec nad ktorým sa má výpočet uskutočniť. Možné výpočty sú

- *Cel* – výpočet merného tepla vodivostných elektrónov na základe parametra γ
- *Cph* – príspevok kmitov kryštálovej mriežky (fonónový príspevok) – súčet Debyeho a Einsteinovho modelu
- *Schottky* – Schottkyho príspevok



Obr. 4.1: Tabuľkový procesor

- *Own* – vlastná operácia, ktorú je možné zostaviť z jednoduchých operácií sčítanie, odčítanie, násobenie a delenie. Ich parametrom môže byť číslo alebo stĺpec, príp. ďalšia operácia (syntax je popísaná priamo v zadávanom okne).

Pri prvých troch typoch výpočtov je nutné špecifikovať aj stĺpec s údajmi o teplotách.

Einsteinov model a Shottkyho príspevok majú variabilný počet parametrov. Tieto parametre sú zapisované do tabuľky. Posledný riadok tabuľky umožňuje pridať nový riadok – je označený hviezdičkou. Je nutné vyplniť aspoň jeden parameter, potom sa nový riadok do tabuľky pridá. Riadok je možné zmazať klávesom `Del` po kliknutí na hlavičku riadka.

Riadky, resp. stĺpce je možné pridávať kliknutím na príslušné talčidlá (`Add Row(s)`, resp. `Add Column`), príp. obdobnými položkami kontextového menu (vyvolané kliknutím na tabuľku pravým tlačidlom myši).

Tlačidlom `Refresh` sa všetky výpočtové stĺpce nanovo prepočítajú.

V kontextovom menu sú ešte položky pre vkladanie dát zo schránky do dátových stĺpcov (`Paste`), úpravu parametrov výpočtových stĺpcov (`Column parameters`), generovanie postupnosti hodnôt s voliteľným začiatkom, koncom a veľkosťou kroku pre dátové stĺpce (`Generate column values`).

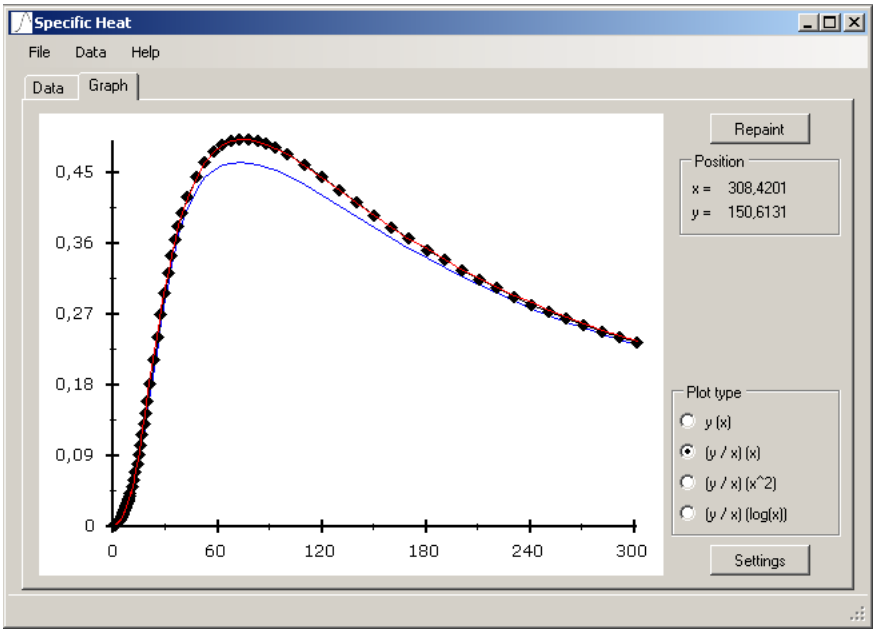
Ďalej zmena názvu stĺpca (rovnakú akciu vyvolá aj dvojklik na hlavičku stĺpca) a spôsobu zobrazovania číselných hodnôt v stĺpci (`Column appearance` a podzložky `Name`, resp. `Format`), vyvolanie okna so zoznamom liniek zobrazovaných v grafe (`Graph lines`), nulovanie, resp. odstránenie označených buniek (`Zerofill`, resp. `Remove selected cells`) a konečne mazanie aktuálneho riadka, resp. stĺpca (`Delete`

`row`, resp. `Delete column`).

Graf

Ďalšia záložka zobrazuje vybrané dáta v grafe.

V sekcii `Plot type` je možné prepínať medzi spôsobmi zobrazenia (lineárne, logaritmické a pod.), tlačidlo `Repaint` prekreslí graf, `Settings` otvorí okno s nastaveniami grafu.



Obr. 4.2: Graf

V nastaveniach je možné v záložke `Axises` pre kadžú os zvoliť rozsah, príp. ho nechať zvoliť automaticky. Rovnako je možné nastaviť značky na osi (`Marks`). Nastavenie `Interval` v sekcii `Small` značí rozostup medzi dvoma značkami – pri spôsobe zobrazenia `y(x)` je to absolútna vzdialenosť, pri `(y/x)(log(x))` je to rozostup medzi dvoma logaritmami a pod. Nastavenie `Frequency` v sekcii `With values` určuje s akou frekvenciou sú značky na osi s popísanou hodnotou (napr. pri nastavení 2 je s hodnotou každá druhá značka). Aj tieto nastavenia je možné nechať určiť automaticky.

Ďalej je možné nastaviť mriežku (`Grid`), farby (`Colour`), a hrúbky (`Width`) či veľkosti (`Size`) osí, resp. značiek na nich. Rovnako font číselných popisov osi (`Font`), či spôsob ich zobrazovania (`Format`).

V záložke `Data` je zoznam zobrazovaných liniek. Tlačidlá `Add` a `Delete` slúžia na pridanie novej, resp. odobranie označenej linky. Pridaním novej, resp. úpravou existujúcej linky (dvojité kliknutie na zvolenú linku) sa otvorí okno `Data selection`.

Nastavenie zobrazovanej linky

V okne `Data selection` sa nastavujú parametre zobrazovanej linky, resp. skupiny dát.

Predovšetkým je potrebné zaškrtávacími políčkami v zozname stĺpcov určiť stĺpce pre x-ovú a y-ovú os zobrazovaných dát, príp. stĺpec chýb (`Error`).

Nastaviť je možné symbol pre zobrazované body (`Type` v sekcii `Symbol`), jeho veľkosť (`Size`), farbu (`Colour`), určiť, či majú byť body spájané úsečkami (`line points`, príp. šírku úsečiek – `line width`) a či sa majú v grafe zobrazovať chyby (`show errors`).

V sekcii `Fitting` sa určuje, či sa k danej skupine dát má fitovať zvolený model. `Colour` v rámci tejto sekcie určuje farbu, akou sa nafitovaný odhad vykreslí, `line width` šírku jeho čiary. Tlačidlom `Fitting` sa otvára okno s nastavením fitovacích parametrov.

Okno `Fitting` ponúka výber troch modelov. Pri každom parametri je zaškrtávacie políčko umožňujúce daný parameter počas fitovania nemeniť (`Fix`). Je možné zvoliť medzi simplexovou a Marquardtovou fitovacou metódou (`Algorithm`). Fit sa uskutoční tlačidlom `Use` (`OK` okrem toho ešte zavrie okno). Po každom fite sa zobrazí hodnota χ^2 (`Chi^2`) modelu s nafitovanými parametrami.

Program ešte umožňuje spočítať Hamiltonián, uložiť a načítať úlohu, ako aj načítať dáta z externého textového súboru (`Import`). Tieto možnosti sú prístupné z hlavného menu.

Pre načítaný textový súbor je k dispozícii náhľad (`Preview`). Zadať je potrebné oddeľovač stĺpcov – vytlačiteľný znak, príp. znak konca riadka (`(CR/)LF`) alebo tabulátor (`Tab`). Následne je možnosť vybrať požadované stĺpce z rozpoznaných.

4.3 Triedy

Program je rozdelený na tieto základné časti.

- `Calc` - statická trieda implementujúca všetky potrebné výpočty
- `Graph` - trieda starajúca sa o vykresľovanie grafu
- `ReadDataFile` - načítanie textových súborov
- triedy formulárov GUI - spracúvajú požiadavky užívateľa

Calc

V tejto statickej triede sú implementované všetky algoritmy popísané v tretej kapitole.

Pôvodne mala byť implementovaná len Marquardtova fitovacia metóda, avšak v praxi sa zistilo, že je pre praktické použitie nedostatočná – príliš často odhady končili v lokálnom minime v tesnej blízkosti prvotných odhadov a zlepšili sa len minimálne, či dokonca vôbec.

Preto je okrem metódy `NonLinearFit`, ktorá implementuje Marquardtov algoritmus, implementovaná aj metóda `SimplexFit`, ktorá sa o zlepšenie odhadov pokúša prostredníctvom simplexovej metódy.

Tieto metódy ale nie sú volané priamo. Verejnými metódami sú `CphFit` a `SchottkyFit`, ktoré po kontrole vstupných parametrov zavolajú na základe *bool* parametra `simplex` buď simplexovú alebo Marquardtovu metódu s danou modelovou funkciou. Vo výstupnom parametri vracajú aj χ^2 výsledného odhadu parametrov.

V niektorých algoritmoch sú volené prísnejšie konštanty (napr. vyšší počet zlých iterácií požadovaný pre ukončenie algoritmu v Marquardtovej metóde), než aké odporúča literatúra, keďže významnejšie neovplyvnili rýchlosť algoritmov. Nezdá sa však, že by mali významný pozitívny vplyv na výsledky.

Graph

Počas existencie programu existuje vždy jeden objekt triedy `Graph`, ten obhospodaruje kreslenie do grafu a všetky potrebné parametre získava z objektu triedy `Graph.Settings`.

Tento uchováva informácie o nastaveniach grafu, kokrétne oboch osí (`AxisSettings`), zoznam nastavení jednotlivých liniek `Graph.GraphLineSettings` a typ zobrazenia grafu (lineárne, logaritmus, ...).

ReadDataFile

Načíta dáta z textového súboru. Poradí si so súborom s danou štruktúrou výstupu experimentálnej aparatúry, ale postačí aj akýkoľvek textový sbor, v ktorom sú riadky dát zároveň riadkami textového súboru.

Triedy formulárov

Najdôležitejším formulárom je trieda `mainWindow`. Hlavné okno obsahuje objekt triedy `DataGridView`, čo je vlastne hlavná tabuľka.

Každý stĺpec odkazuje na objekt so svojimi nastaveniami `Settings`. Dátové stĺpce tam majú uložený len formát, výpočtové typ a parametre výpočtu – v prípade vlastného výpočtu sa ukladá strom objektov triedy `CommandNode`.

Každý z týchto objektov obsahuje typ operácie a odkaz na `CommandNode`-y svojich operandov, v prípade uzla reprezentujúceho číslo aj číselnú hodnotu, či odkaz na stĺpec (ak uzol reprezentuje stĺpec).

Kapitola 5

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť použiteľný program schopný spracovať experimentálne dáta meraní tepelnej kapacity pokusných vzoriek.

Program je schopný spracovať výstupné súbory z experimentálneho zariadenia, ako aj akýkoľvek rozumne štruktúrovaný textový súbor, príp. dáta načítať zo schránky. Získané dáta je možné následne úpraviť základnými matematickými operáciami a vyniesť do grafu.

Program rovnako dokáže spočítať požadované fyzikálne príspevky ako aj ďalšiu požadovanú operáciu – výpočet Hamiltoniánu. Implementované je aj uloženie a opätovné načítanie stavu programu do štruktúrovaného XML súboru.

Možné je aj fitovanie parametrov vybraného modelu na zvolené dáta. Použité sú dva osvedčené algoritmy rôznych typov – Marquardtova a simplexová metóda. Užívateľ tak má možnosť operatívne meniť nielen parametre fitovaného modelu, ale aj samotné postupy hľadania lepších odhadov parametrov. Úspešnosť odhadu si môže overiť jednak hodnotou χ^2, ako aj vizuálnou kontrolou dát vynesených do grafu.

Otázkou je, či program obstojí pri pravidelnom používaní. Základné požiadavky kladené naň však boli splnené. Postupným zapracovávaním nových pripomienok by sa z neho ale mohol stať prakticky používaný nástroj.

Literatúra

[1] Hutchings M. T.: *Point-charge calculations of energy levels of magnetic ions in crystalline electric fields*, Solid State Physics **16** (1964) 227–273.

[2] Kittel, Ch.: *Úvod do fyziky pevných látek*, Academia, Praha, 1985.

[3] Meloun M., Militký J.: *Statistická analýza experimentálních dat*, Academia, Praha, 2004.

[4] Nelder J. A., Mead R.: *A simplex method for function minimization*, Computer Journal **7** (1965) 308–313.

[5] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes Example Book (C++)*, Cambridge University Press, Cambridge, 2002.

[6] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes in C++*, Cambridge University Press, Cambridge, 2002.

Dodatok A

Adresárová štruktúra CD

`/` – bakalárska práca vo formáte PDF
`/bin` – spustiteľný program pre platformu win32
`/doc` – užívateľská dokumentácia
`/inst` – inštalačné súbory pre platformu win32
`/src` – zdrojové súbory programu
`/test` – testovacie súbory