Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Michal Botka

Řešení soustav lineárních rovnic

Katedra algebry

Vedoucí bakalářské práce: RNDr. David Stanovský, Ph.D.
Studijní program: matematické metody informační bezpečnosti

2006

Děkuji svému vedoucímu práce RNDr. Davidu Stanovskému, Ph.D. za jeho cenné rady a pomoc při tvorbě této bakalářské práce.

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

V Praze dne 26.7. 2006 Michal Botka

Obsah

Název práce: Řešení soustav lineárních rovnic
Autor: Michal Botka
Katedra (ústav): Katedra algebry
Vedoucí bakalářské práce: RNDr. David Stanovský, Ph.D.
e-mail vedoucího: stanovsk@karlin.mff.cuni.cz

Abstrakt: V předložené práci studujeme metody pro řešení soustav lineárních rovnic. Výchozím bodem práce je klasická Gaussova eliminační metoda, z které odvodíme bezzlomkovou Gaussovu eliminační metodu a modulární metodu. Porovnáváme časovou složitost metod a testujeme dobu výpočtu na různých typech dat. Všechny popsané metody jsou implementovány v jazyce C. Vstupem programu je matice nad oborem celých čísel nebo oborem polynomů jedné proměnné s celočíselnými koeficienty. Ukážeme, že ačkoliv je možné provádět výpočty v podílových tělesech, je efektivnější provést většinu výpočtů v původních oborech.

Klíčová slova: soustava lineárních rovnic, Gaussova eliminace

Title: Solving systems of linear equations
Author: Michal Botka
Department: Department of Algebra
Supervisor: RNDr. David Stanovský, Ph.D.
Supervisor's e-mail address: stanovsk@karlin.mff.cuni.cz

Abstract: In the present work we study methods of solving systems of linear equations. The initial point is the ordinary Gaussian elimination which we use to derive methods like fraction-free Gaussian elimination and modular method. We compare the time complexity of methods and we test the computation speed on different types of inputs. All discribed methods are implemented in the C programming language. The input of the program is a matrix over the integral domain or the integral polynomial domain. We show that the computation in integral domains is more effective than computing in their quotient fields.

Keywords: system of linear equations, Gaussian elimination

Kapitola 1

Úvod

Algoritmy pro řešení soustav lineárních rovnic lze v zásadě rozdělit do dvou skupin podle toho, zda poskytují pouze přibližné řešení nebo přesné řešení.

Algoritmy pro nalezení přibližného řešení jsou předmětem numerické matematiky a v této práci se jimi nebudeme zabývat. Oproti algoritmům poskytující přesné řešení, jež jsou předmětem této práce, jsou obvykle daleko rychlejší, avšak právě z důvodu chyby jsou pro některé typy úloh nepoužitelné.

Budeme pracovat se soustavami, jejichž koeficienty jsou z oboru celých čísel nebo z oboru polynomů s celočíselnými koeficienty. Ačkoliv je možné provést výpočty v podílových tělesech těchto oborů, lze postupovat efektivněji. Cílem bude odvodit metodu bezzlomkové Gaussovy eliminace, která se po svém autorovi nazývá někdy v literatuře jako Bareissův algoritmus, a modulární algoritmus, jež využívá zrychlení výpočtů s prvky v modulární reprezentaci.

1.1 Základní pojmy

Nyní v krátkosti zopakujme základní pojmy z lineární algebry, které pro nás budou klíčové.

Definice 1.1 *Soustavou m lineárních rovnic o n neznámých nad tělesem T rozumíme soustavu tvaru*

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & b_1, \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & b_2, \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \ldots & + & a_{mn}x_n & = & b_m, \end{array}$$

kde a_{ij}, b_i, $i = 1, \ldots, m$, $j = 1, \ldots, n$ jsou nějaké prvky tělesa T. Vektor $(x_1, x_2, \ldots, x_n)$ takový, že jeho dosazením do soustavy je splněno všech m rovností, se nazývá řešení soustavy. Matice $A = (a_{ij})$ typu (m, n) se nazývá matice soustavy. Vektor $b = (b_1, b_2, \ldots, b_m)^T$ je sloupec pravých stran. Matice

$$\begin{pmatrix} a_{11}, & a_{12}, & \ldots, & a_{1n} & b_1 \\ a_{21}, & a_{22}, & \ldots, & a_{2n} & b_2 \\ a_{m1}, & a_{m2}, & \ldots, & a_{mn} & b_m \end{pmatrix}$$

typu $(m, n+1)$ se nazývá rozšířená matice soustavy.

Pozorování 1.2 *Použijeme-li značení z předchozí definice: $A = (a_{ij})$, $x = (x_1, x_2, \ldots, x_n)^T$, $b = (b_1, b_2, \ldots, b_n)^T$, lze soustavu přepsat do maticového tvaru*

$$Ax = b$$

Definice 1.3 *Nenulová matice A typu (m, n) nad tělesem T se nazývá Gaussova, jestliže existují indexy $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ tak, že $a_{1i_1} \neq 0$, $a_{2i_2} \neq 0$, $\ldots$, $a_{ki_k} \neq 0$ a $a_{ji} = 0$, jestliže buď $k < j \leq m$, nebo $j \leq k$ a zároveň $i < i_j$.*

Věta 1.4 *Každou nenulovou matici lze konečným počtem elementárních transformací na řádky matice převést na Gaussovu matici.*

Řešení soustavy lineárních rovnic získáme tak, že rozšířenou matici soustavy transformujeme na Gaussovu matici, což díky předchozí větě lze provést vždy.

V této práci budeme předpokládat, že prvky vstupní matice jsou z R, kde R je nějaký obor integrity, např. $\mathbb{Z}$ nebo $\mathbb{Z}[x]$. V principu lze pracovat v podílovém tělese Q, ale je to neefektivní. Naším cílem bude vyvinout metody, kde se většina výpočtů odehrává v R.

Kapitola 2

Gaussova eliminační metoda

Uvažme soustavu lineárních rovnic nad oborem integrity R zapsanou ve tvaru

$$Ax = b, \tag{2.1}$$

kde A je matice typu (m, n), $x = (x_1, x_2, \ldots, x_n)^T$ a $b = (b_1, b_2, \ldots, b_m)^T$. Rozšířenou matici soustavy označme $A^{(0)}$. Na matici $A^{(0)}$ aplikujeme sérii transformací

$$A^{(0)} \to A^{(1)} \to \ldots$$

tak, aby výsledná matice soustavy byla Gaussova matice.

2.1 Klasická Gaussova metoda

Algoritmus popíšeme v úplné obecnosti bez dodatečných předpokladů jako je čtvercovost či regulárnost matice soustavy.

Algoritmus 2.1 (GO)

Mějme tedy rozšířenou matici soustavy $A = (a_{ij})$ typu $(m, n + 1)$.

0. V inicializační fázi nastavíme pomocné proměnné $u = 1$, $v = 1$, prvek a_{uv} je vedoucím prvkem.

1. Je-li $u \leq m$ a $v \leq n + 1$, pokračujeme dále, v opačném případě jsme hotovi a ukončíme algoritmus.

2. Je-li $a_{uv} = 0$, hledáme takové $i > u$, aby $a_{iv} \neq 0$. Jestliže takové i existuje prohodíme řádky u a i.

3. Je-li nyní $a_{uv} \neq 0$, pak od i-tých řádků, $i = u+1, \ldots, m$ odečteme a_{iv}/a_{uv} násobek u-tého řádku a nakonec zvýšíme u o 1.

4. Zvýšíme v o 1 a přejdeme na krok 1.

Pozorování 2.2 *Výše popsaný algoritmus převede v konečném počtu kroků pomocí elementárních řádkových transformací libovolnou nenulovou matici na Gaussovu matici.*

Princip algoritmu je jasný, stačí se podívat, co se děje s v-tým sloupcem v kroku 3.

$$a_{iv} - \frac{a_{iv}}{a_{uv}}\, a_{uv} = a_{iv} - a_{iv} = 0,\ i = u+1, \ldots, m$$

Postupně dojde v každém sloupci k vynulování všech prvků ležících pod vedoucím prvkem a tím se matice převede na Gaussův tvar dle definice 1.3.

Příklad 2.3 Ukažme si, jak vypadají jednotlivé kroky algoritmu na jednoduchém příkladě soustavy 4 rovnic o 4 neznámých.

$$A^{(0)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ -3 & 7 & 4 & -1 & 2 \\ 1 & 9 & -2 & 2 & 4 \\ 4 & 6 & 4 & -3 & 2 \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41/5 & 14/5 & -2/5 & 4/5 \\ 0 & 43/5 & -8/5 & 9/5 & 22/5 \\ 0 & 22/5 & 28/5 & -19/5 & 18/5 \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41/5 & 14/5 & -2/5 & 4/5 \\ 0 & 0 & -186/41 & 91/41 & 146/41 \\ 0 & 0 & 168/41 & -147/41 & 130/41 \end{pmatrix}$$

$$A^{(3)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41/5 & 14/5 & -2/5 & 4/5 \\ 0 & 0 & -186/41 & 91/41 & 146/41 \\ 0 & 0 & 0 & -49/31 & 198/31 \end{pmatrix}$$

Jestliže obor R není těleso, je nevyhnutelné pracovat s jeho podílovým tělesem. V našem případě soustav s celočíselnými koeficienty vede tento postup na počítání se zlomky, což se negativně projeví na době výpočtu. Pokročilejší metody umožňují provádět většinu výpočtů v oboru celých čísel a tím snížit časovou náročnost.

2.2 Gaussova metoda bez dělení

Jednoduchou úpravou předchozího algoritmu můžeme docílit toho, že se zbavíme nepříjemného dělení v třetím kroku.

Algoritmus 2.4 (GDF)

0. V inicializační fázi nastavíme pomocné proměnné $u = 1$, $v = 1$, prvek a_{uv} je vedoucím prvkem.

1. Je-li $u \leq m$ a $v \leq n + 1$, pokračujeme dále, v opačném případě jsme hotovi a ukončíme algoritmus.

2. Je-li $a_{uv} = 0$, hledáme takové $i > u$, aby $a_{iv} \neq 0$. Jestliže takové i existuje prohodíme řádky u a i.

3. Je-li nyní $a_{uv} \neq 0$, pak každý řádek $i = u + 1, \ldots, m$ vynásobíme a_{uv} a odečteme od něho a_{iv} násobek řádku u. Nakonec zvýšíme u o 1.

4. Zvýšíme v o 1 a přejdeme na krok 1.

Příklad 2.5 Ukažme si jak vypadají jednotlivé kroky algoritmu na příkladě.

$$A^{(0)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ -3 & 7 & 4 & -1 & 2 \\ 1 & 9 & -2 & 2 & 4 \\ 4 & 6 & 4 & -3 & 2 \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 43 & -8 & 9 & 22 \\ 0 & 22 & 28 & -19 & 18 \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 0 & -930 & 455 & 730 \\ 0 & 0 & 840 & -735 & 650 \end{pmatrix}$$

$$A^{(3)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 0 & -930 & 455 & 730 \\ 0 & 0 & 0 & 301350 & -1217700 \end{pmatrix}$$

Již z příkladu je patrné, že nevýhodou této metody je velký růst délky koeficientů. Ačkoliv je tato metoda pro větší soustavy nepoužitelná, lze z ní vycházet a vylepšit ji.

2.3 Bezzlomková Gaussova metoda

Vezměme matici $A^{(0)}$ a aplikujme na ni několik kroků algoritmu GDF. Pro zjednodušení označme řádky matice $\bar{a} = (a_1, \ldots, a_m)$, $\bar{b} = (b_1, \ldots, b_m)$, ... Předpokládejme, že $a_1 \neq 0$ a $(a_1 b_2 - a_2 b_1) \neq 0$.

$$A^{(0)} = \begin{pmatrix} \bar{a} \\ \bar{b} \\ \bar{c} \\ \vdots \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} \bar{a} \\ a_1\bar{b} - b_1\bar{a} \\ a_1\bar{c} - c_1\bar{a} \\ \vdots \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} \bar{a} \\ a_1\bar{b} - b_1\bar{a} \\ (a_1b_2 - b_1a_2)(a_1\bar{c} - c_1\bar{a}) - (a_1c_2 - c_1a_2)(a_1\bar{b} - b_1\bar{a}) \\ \vdots \end{pmatrix}$$

Třetí řádek matice $A^{(2)}$ dále upravíme.

$$(a_1b_2 - b_1a_2)(a_1\bar{c} - c_1\bar{a}) - (a_1c_2 - c_1a_2)(a_1\bar{b} - b_1\bar{a}) =$$

$$a_1^2b_2\bar{c} - a_1b_2c_1\bar{a} - a_1a_2b_1\bar{c} + a_2b_1c_1\bar{a} - a_1^2c_2\bar{b} + a_1b_1c_2\bar{a} + a_1a_2c_1\bar{b} - a_2b_1c_1\bar{a} =$$

$$a_1((b_1c_2 - b_2c_1)\bar{a} + (a_2c_1 - a_1c_2)\bar{b} + (a_1b_2 - a_2b_1)\bar{c})$$

Jak je vidět, třetí řádek matice $A^{(2)}$ je dělitelný prvkem a_1. Stejně tak jsou a dělitelné i všechny následující řádky. Analogický vztah platí pro všechny matice $A^{(i)}$. Tím dostáváme algoritmus bezzlomkové Gaussovy eliminace.

Algoritmus 2.6 (GFF)

0. V inicializační fázi nastavíme pomocné proměnné $u = 1$, $v = 1$, prvek a_{uv} je vedoucím prvkem. Dále potřebujeme pomocné proměnné $r = 0$, $s = 0$, prvkem a_{rs} budeme dělit. Pro tyto účely dodefinujme $a_{00} = 1$.

1. Je-li $u \leq m$ a $v \leq n + 1$, pokračujeme dále, v opačném případě jsme hotovi a ukončíme algoritmus.

2. Je-li $a_{uv} = 0$, hledáme takové $i > u$, aby $a_{iv} \neq 0$. Jestliže takové i existuje, prohodíme řádky u a i.

3. Je-li nyní $a_{uv} \neq 0$, pak každý řádek $i = u + 1, \ldots, m$ vynásobíme a_{uv}, odečteme od něho a_{iv} násobek řádku u a vydělíme prvkem a_{rs}. Nakonec položíme $r = u$, $s = v$ a zvýšíme u o 1.

4. Zvýšíme v o 1 a přejdeme na krok 1.

Příklad 2.7 Algoritmus aplikujeme na příkladě.

$$A^{(0)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ -3 & 7 & 4 & -1 & 2 \\ 1 & 9 & -2 & 2 & 4 \\ 4 & 6 & 4 & -3 & 2 \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 43 & -8 & 9 & 22 \\ 0 & 22 & 28 & -19 & 18 \end{pmatrix}$$

$$A^{(2)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 0 & -186 & 91 & 146 \\ 0 & 0 & 168 & -147 & 130 \end{pmatrix}$$

$$A^{(3)} = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ 0 & 41 & 14 & -2 & 4 \\ 0 & 0 & -186 & 91 & 146 \\ 0 & 0 & 0 & 294 & -1188 \end{pmatrix}$$

Tato metoda splňuje naše požadavky, výpočty se provádí celočíselně a růst délky koeficientů je přijatelný.

Další způsob zrychlení algoritmu spočívá ve spojení dvou kroků GFF v jeden. Pro účely této práce označme vylepšený GFF algoritmus jako GF2.

Algoritmus 2.8 (GF2)

0. V inicializační fázi nastavíme pomocné proměnné $u = 1$, $v = 1$, prvek a_{uv} je vedoucím prvkem. Dále potřebujeme pomocné proměnné $r = 0$, $s = 0$, prvkem a_{rs} budeme dělit. Pro tyto účely dodefinujme $a_{00} = 1$.

1. Je-li $u \leq m$ a $v \leq n + 1$, pokračujeme dále, v opačném případě jsme hotovi a ukončíme algoritmus.

2. Je-li $a_{uv} = 0$, hledáme takové $i > u$, aby $a_{iv} \neq 0$. Jestliže takové i existuje, prohodíme řádky u a i.

3. Je-li nyní $a_{uv} \neq 0$, pak hledáme řádek $i > u$ takový, aby $a_{i,v+1}a_{uv} - a_{iv}a_{u,v+1} \neq 0$. Neexistuje-li, pak každý řádek $i = u + 1, \ldots, m$ vynásobíme a_{uv}, odečteme od něho a_{iv} násobek řádku u, vydělíme prvkem a_{rs} a položíme $r = u$, $s = v$ a zvýšíme u o 1, stejně jako v případě GFF.

   Pokud takové i nalezneme, prohodíme i-tý a $u + 1$ řádek a spočítáme:

$$z_1 = (a_{uv}a_{u+1,v+1} - a_{u+1,v}a_{u,v+1})/a_{rs}$$

   Následně pro každé $k = u + 2, \ldots, m$ spočítáme:

$$\begin{aligned} z_2 &= (a_{u+1,v}a_{k,v+1} - a_{u+1,v+1}a_{k,v})/a_{rs} \\ z_3 &= (a_{u,v+1}a_{k,v} - a_{u,v}a_{k,v+1})/a_{rs} \\ z_4 &= (z_1a_{kj} + z_2a_{u,j} + z_3a_{u+1,j})/a_{rs} \end{aligned}$$

Nakonec do a_{kj} uložíme hodnotu z_4, vypočítáme $u+1$ řádek podle GFF schematu, nastavíme $r = u+1$, $s = v+1$, u a v zvýšíme o 2 a přejdeme na krok 1

4. Zvýšíme v o 1 a přejdeme na krok 1.

2.4 Modulární algoritmus

Bezzlomková Gaussova eliminace je metodou, která vylepšuje klasickou Gaussovu eliminaci tím, že větší část výpočtů probíhá v oboru R bez nutnosti přecházet k podílovému tělesu Q. Konkrétně v případě oboru integrity celých čísel to znamená, že se vyhneme počítání se zlomky. Modulární algoritmus přistupuje k problému odlišně. Prvky matice nahradí jejich modulární reprezentací a v této reprezentaci se provádí všechny výpočty. Výsledek je poté převeden z modulární reprezentace zpět. Pro obor celých čísel je vhodnou modulární reprezentací konečné těleso $\mathbb{Z}_p$, pro některé jiné obory nemusí ani žádná vhodná reprezentace existovat, což zužuje možnost použití modulární metody oproti univerzálnosti Gaussovy eliminační metody.

Máme-li soustavu s celočíselnými koeficienty, lze očekávat, že koeficienty řešení budou ležet v $\mathbb{Q}$, což nám znemožňuje použít modulární reprezentaci. Má-li soustava právě jedno řešení, lze jej vyjádřit jako podíl determinantů. Z toho důvodu funguje modulární algoritmus pouze pro čtvercové soustavy s regulární maticí. Než přistoupíme k popisu modulárního algoritmu pro řešení soustav lineárních rovnic, potřebujeme několik vět.

Věta 2.9 (Cramerovo pravidlo) *Mějme soustavu lineárních rovnic tvaru $Ax = b$, kde $A = (a_{ij})$ je regulární čtvercová matice stupně n a vektor $b = (b_1, b_2, \ldots, b_n)^T$ je sloupec pravých stran. Označme A_j matici, která vznikne z matice A nahrazením j-tého sloupce vektorem b, a dále označme $x_j = \frac{detA_j}{detA}$, $j = 1, 2, \ldots, n$. Potom $(x_1, x_2, \ldots, x_n)$ je řešením soustavy.*

Věta 2.10 (Čínská věta o zbytcích) *Nechť $m_1, m_2, \ldots, m_n$ jsou po dvou nesoudělná čísla, $N = m_1 \ldots . m_n$. Pak pro každé $u_1, u_2, \ldots u_n$ existuje právě jedno $0 \leq x < N$, že pro všechna $i = 1, 2, \ldots n$ platí $x \equiv u_i \pmod{m_i}$.*

Důkaz. Stačí nalézt $a_1, a_2, \ldots a_n$ splňující kongruence $a_i \equiv 1 \pmod{m_i}$ a $a_i \equiv 0 \pmod{m_j}$ pro $j \neq i$. Položíme-li $x = \sum_{i=1}^{n} u_i a_i \pmod{N}$, získáme hledané řešení. Označíme-li $M_i = \prod_{j \neq i} m_j$, pak z předpokladu věty plyne, že m_i a

M_i jsou nesoudělná. Pomocí rozšířeného Euklidova algoritmu najdeme s_i a t_i taková, že $s_i m_i + t_i M_i = 1$. Snadno nahlédneme, že $a_i \equiv 1 - s_i m_i \pmod{N}$ má požadovanou vlastnost.

Nechť čísla x_1 a x_2 splňují soustavu kongruencí $x_j \equiv u_i \pmod{m_i}$, pro $j = 1, 2$ a $i = 1, \ldots n$. Bez újmy na obecnosti lze předpokládat, že $0 \leq x_1 \leq x_2 < N$. Odečtením rovnic dostáváme $x_2 - x_1 \equiv 0 \pmod{m_i}$ pro $i = 1, \ldots, n$, odkud díky nesoudělnosti čísel m_i plyne $x_2 - x_1 \equiv 0 \pmod{N}$. Protože je $0 \leq x_2 - x_1 < N$, musí být $x_2 - x_1 = 0$ a tedy $x_1 = x_2$. Tím je jednoznačnost dokázána.

Tyto věty nyní využijeme ke konstrukci modulárního algoritmu. Mějme soustavu n lineárních rovnic o n neznámých ve tvaru

$$Ax = b.$$

Předpokládejme navíc, že A je regulární matice. Podle věty 2.9 platí, že soustava má právě jedno řešení $(x_1, x_2, \ldots, x_n)$, kde x_i je rovno podílu determinantů příslušných matic A_i, A.

Označme $A^* = (a_{ij})$ rozšířenou matici soustavy a pro zvolené prvočíslo p definujme matici $M^* = (m_{ij})$ typu $(n, n+1)$ vztahem $m_{ij} = a_{ij} \pmod{p}$. Nad tělesem $\mathbb{Z}_p$ vyřešme Gaussovou eliminací soustavu příslušnou matici M^*. Nechť M je matice soustavy příslušná rozšířené matici soustavy M^* a M_i je matice, která vznikne z M nahrazením i-tého sloupce sloupcem pravých stran. Jestliže p nedělí $\det A$, pak $\det M \neq 0$ a soustava má právě jedno řešení $(y_1, y_2, \ldots, y_n)$, kde $y_i = \frac{detM_i}{detM}$, odkud dostáváme $\det M_i = y_i \det M$. Vrátíme-li se zpět k původní soustavě, obdržíme

$$\det A \equiv \det M \pmod{p} \quad \det A_i = \det M_i \pmod{p}.$$

Jestliže nyní postup zopakujeme pro dostatečně mnoho prvočísel $p_1, p_2, \ldots$, jsme schopni pomocí věty 2.10 získat $\det A, \det A_1, \ldots, \det A_n$ a tím i řešení soustavy příslušné matici A^* podle věty 2.9.

Protože předem nevíme nic o hodnotách $\det A, \det A_1, \ldots, \det A_n$, nelze předem určit, kolik dílčích výpočtů s prvočísly p_i je třeba provést. Navíc ne všechna prvočísla jsou vhodná, neboť musí platit, že p_i nedělí $\det A$. Je-li $\det A = 0$, pak Cramerovo pravidlo nelze použít. Tuto situaci ale nelze detekovat okamžitě, protože jediné, co během výpočtů získáme, jsou kongruence $\det A \equiv 0 \pmod{p_i}$, pro nějaká $i = 1, \ldots, k$. Z čehož plyne pouze, že $\prod_{i=1}^{k} p_i$

dělí $\det A$, nikoliv rovnost $\det A = 0$. Následující věta nám poskytne horní odhad determinantu.

Věta 2.11 (Hadamardova nerovnost) *Nechť $a_1, \ldots, a_n \in \mathbb{R}^n$ jsou sloupce čtvercové matice $A = (a_{ij})$ takové, že $|a_{ij}| \leq 1$ pro všechna $1 \leq i, j \leq n$. Pak platí nerovnost*

$$|\det A| \leq \prod_{i=1}^{n} |a_i|,$$

kde $|a_i|$ značí Euklidovu normu vektoru a_i v $\mathbb{R}^n$.

Důkaz. Předpokládejme, že A je regulární, pro singulární případ nerovnosti platí triviálně. Na množinu $\{a_1, \ldots, a_n\}$ aplikujme Grammův-Schmidtův ortogonalizační proces a získáme ortonormální bázi $\{b_1, \ldots, b_n\}$. Pro každé $m = 1, \ldots, n$ platí

$$a_m = \sum_{i=1}^{m} \langle a_m, b_i \rangle b_i.$$

Označme matici $B = (b_1, \ldots, b_n)$ a dále definujme matici $C = (c_{kl})$ typu n takto

$$\begin{aligned} c_{kl} &= \langle a_l, b_k \rangle \text{ je-li } 1 \leq k \leq l, \\ c_{kl} &= 0 \text{ jinak.} \end{aligned}$$

To nám umožňuje vyjádřit matici A jako

$$A = BC.$$

Dále platí

$$(\det A)^2 = \det A^T A = \det C^T B^T B C = \det C^T C = (\det C)^2$$

$$= \prod_{i=1}^{n} |\langle a_i, b_i \rangle|^2 \leq \prod_{i=1}^{n} \sum_{m=1}^{i} |\langle a_m, b_i \rangle|^2 = \prod_{i=1}^{n} |a_i|^2$$

a tím je důkaz hotov.

Lemma 2.12 (Odhad determinantu) *Nechť $A = (a_{ij})$ je čtvercová matice nad $\mathbb{R}$ řádu n. Označme $M = \max_{i,j} |a_{ij}|$. Pak platí*

$$|\det A| \leq M^n \prod_{i=1}^{n} |a_i| \leq M^n n^{\frac{n}{2}}$$

Důkaz. Vydělíme každý matice řádek číslem M a aplikujeme Hadamardovu nerovnost.

Algoritmus 2.13 (HOM)

0. V inicializační fázi spočteme $H = 2n^{\frac{n}{2}}(\max_{i,j}|a_{ij}|)^n$ a nastavíme pomocné proměnné $k = 0$, $p_0 = 1$.

1. Je-li $\prod_{i=1}^{k} p_i > H$, algoritmus neúspěšně skončí.

2. Zvýšíme k o 1, nalezneme prvočíslo $p_k > p_{k-1}$ a spočteme matici $M_k = (m_{ij}^k)$ podle vzorce $m_{ij} \equiv a_{ij} \pmod{p}$.

3. Na matici M_k aplikujeme algoritmus GO v tělese $\mathbb{Z}_{p_k}$, determinant matice M_k označme d_k.

4. Je-li $d_k \neq 0$, určíme jediné řešení soustavy příslušné matici M_k $x_k = (x_{k1}, \ldots, x_{kn})$ a vektor $y_k = (y_{k1}, \ldots, y_{kn}) = d_k x_k$.

5. Určíme množinu indexů $K = \{1 \leq k,\ d_k \neq 0\}$ a pomocí čínské věty o zbytcích hledáme s takové, aby

$$s \equiv d_k \pmod{p_k},\ k \in K,$$

a takové s_i, $i = 1, \ldots, n$, aby

$$s_i \equiv y_{ki} \pmod{p_k},\ k \in K,$$

přičemž volíme hodnoty ze symetrické reprezentace okruhu $\mathbb{Z}_p$, $p = \prod_{k \in K} p_k$. Ověříme, zda vektor $\frac{1}{s}(s_1, \ldots, s_n)$ je řešením původní soustavy, pokud ne, přejdeme na krok 1, jinak algoritmus úspěšně skončí.

Příklad 2.14 Algoritmus aplikujeme na příkladě, pro názornost použijeme prvočísla $p_1 = 11, p_2 = 13, p_3 = 17$.

$$A^* = \begin{pmatrix} 5 & 2 & -2 & 1 & -2 \\ -3 & 7 & 4 & -1 & 2 \\ 1 & 9 & -2 & 2 & 4 \\ 4 & 6 & 4 & -3 & 2 \end{pmatrix}$$

$$M_1^* = \begin{pmatrix} 5 & 2 & 9 & 1 & 9 \\ 8 & 7 & 4 & 10 & 2 \\ 1 & 9 & 9 & 2 & 4 \\ 4 & 6 & 4 & 8 & 2 \end{pmatrix} \sim \begin{pmatrix} 5 & 2 & 9 & 1 & 9 \\ 0 & 6 & 5 & 4 & 3 \\ 0 & 0 & 7 & 10 & 10 \\ 0 & 0 & 0 & 8 & 0 \end{pmatrix}$$

$$M_2^* = \begin{pmatrix} 5 & 2 & 11 & 1 & 11 \\ 10 & 7 & 4 & 12 & 2 \\ 1 & 9 & 11 & 2 & 4 \\ 4 & 6 & 4 & 10 & 2 \end{pmatrix} \sim \begin{pmatrix} 5 & 2 & 11 & 1 & 11 \\ 0 & 3 & 8 & 10 & 6 \\ 0 & 0 & 11 & 0 & 8 \\ 0 & 0 & 0 & 11 & 11 \end{pmatrix}$$

$$M_3^* = \begin{pmatrix} 5 & 2 & 15 & 1 & 15 \\ 14 & 7 & 4 & 16 & 2 \\ 1 & 9 & 15 & 2 & 4 \\ 4 & 6 & 4 & 14 & 2 \end{pmatrix} \sim \begin{pmatrix} 5 & 2 & 9 & 1 & 9 \\ 0 & 15 & 13 & 3 & 11 \\ 0 & 0 & 5 & 13 & 16 \\ 0 & 0 & 0 & 5 & 2 \end{pmatrix}$$

Vyřešením soustav a aplikací čínské věty o zbytcích získáme z matic M_1^*, M_2^*, M_3^* hodnoty o determinantech.

	mod 11	mod 13	mod 17	mod 2431
$\det A$	-3	-5	5	294
$\det A_1$	4	-5	2	-304
$\det A_2$	-5	1	-7	248
$\det A_3$	2	-6	4	-812
$\det A_4$	0	-5	2	-1188

Dosádíme-li $x_i = \det A_i / \det A$ dostaneme

$$x_1 = -152/147,\ x_2 = 124/147,\ x_3 = -58/21,\ x_4 = -198/49.$$

Nakonec ověříme, že se jedná skutečně o řešení původní soustavy.

Kapitola 3

Složitost algoritmů

V předchozí kapitole jsme odvodili a popsali různé metody řešení soustav lineárních rovnic, aniž bychom se zabývali časovou nebo prostorovou složitostí. Odhad složitosti popsaných metod bude předmětem této kapitoly.

3.1 Časová složitost

Mějme soustavu n lineárních rovnic o m neznámých zadanou pomocí matice $A^{(0)}$ typu $(n, m+1)$. Metody GO, GDF a GFF popsané v kapitole 2 se liší aritmetickými operacemi, jež se provádějí s prvky matice, vnější cyklus je pro všechny algoritmy totožný. Předpokládejme, že výpočet jednoho prvku matice uvnitř cyklu trvá konstatní čas t. Definujeme-li

$$N = \max\{n, m+1\},$$

snadno odhadneme shora čas $T(N)$ potřebný pro transformaci matice pomocí výše zmíněných metod.

Pro výpočet matice $A^{(i)}$ z matice $A^{(i-1)}$ je třeba projít nejvýše N řádků, přičemž na prvních i souřadnicích bude jistě 0. To znamená, že je nutno provést nejvýše $N(N-i)$ aritmetických operací pro výpočet matice $A^{(i)}$. Tím dostáváme odhad pro $T(N)$

$$T(N) \leq \sum_{i=1}^{N} N(N-i)t = \frac{1}{2}N^2(N-1)t \leq N^3 t$$

Vidíme tedy, že počet aritmetických operací potřebných k transformaci matice závisí na její velikosti kubicky.

Algoritmus HOM spočívá v opakovaném provedení algoritmu GO nad tělesem $\mathbb{Z}_p$ pro různá p spolu s aplikací čínské věty o zbytcích a ověření správnosti dosazením.

3.2 Prostorová složitost

U všech popsaných algoritmů s výjimkou algoritmu HOM lze provádět výpočty přímo se vstupními daty, které však během procesu rostou. Tento růst je teoreticky velmi těžké odhadnout, neboť hrubý horní odhad silně přesahuje skutečné využití paměti. Nejvyšší prostorovou složitost vykazuje algoritmus GDF, kde dochází v každém kroku k výraznému zvětšování koeficientů, což ho činí pro větší vstupy zcela nepoužitelným.

V případě algoritmu HOM je třeba v každém kroku uchovávat všechna řešení v $\mathbb{Z}_p$ pro různá p, avšak samotný výpočet je nenáročný na prostor, neboť je každý koeficient omezen p.

Kapitola 4

Implementace a testy

Při implementaci metod pro řešení soustav lineárních rovnic byl kladen důraz zejména na rychlost. Proto byl použit programovací jazyk C, který má kromě rychlosti další podstatnou výhodu, a tou je přenositelnost. Ačkoliv byl program původně vyvíjen v prostředí Windows, je snadno použitelný v prostředí Linux.

Pro práci s dlouhými čísly byla použita knihovna GNU Multiple Precision Arithmetic Library (GMP), jež je proslulá svou rychlostí a navíc je díky licenci GNU snadno dostupná. Kromě knihovny GMP byl navíc použit The Netwide Assembler (NASM) pro implementaci modulární aritmetiky, zbytek programu je napsaný v čistém C, použité jsou pouze standardní knihovny tohoto jazyka.

4.1 Porovnání algoritmů GO, GFF a GF2

Pro porovnání časové složitosti algoritmů GO, GFF a GF2 bylo vygenerováno 50 matic typu $(N, N+1)$. Prvky matic byly vygenerovány pomocí pseudonáhodného generátoru s nula-jedničkovým rozdělením s pravděpodobností $p = 0,5$. Každá z těchto matic se stala vstupem pro testované algoritmy, z naměřených časových údajů byl spočítán aritmetický průměr. Výsledky shrnuje tab.1, naměřené hodnoty jsou v sekundách. Obr.1 obsahuje grafické znázornění, vodorovná osa představuje velikost vstupu N, svislá dobu výpočtu. Test proběhl na počítači s procesorem Intel Pentium II 350MHz, 128MB RAM.

N	GO	GFF	GF2	N	GO	GFF	GF2
20	0,023	0,002	0,002	60	0,661	0,059	0,043
25	0,041	0,004	0,003	65	0,869	0,079	0,055
30	0,071	0,005	0,005	70	1,124	0,105	0,072
35	0,117	0,009	0,006	75	1,436	0,133	0,094
40	0,177	0,015	0,011	80	1,843	0,171	0,119
45	0,255	0,022	0,016	85	2,310	0,213	0,149
50	0,358	0,031	0,023	90	2,864	0,262	0,183
55	0,489	0,043	0,032	100	4,400	0,417	0,283

tab.1

obr.1

S rostoucí velikostí vstupní matice roste během výpočtů velikost prvků matice a tím i časová náročnost prováděných aritmetických operací. Z obr.1 plyne, že operace prováděné v algoritmu GO jsou výrazně pomalejší. Tab. 1 dále ukazuje, že bez ohledu na velikost N je rychlost algoritmu GF2 oproti GFF zhruba 1,5 krát vyšší.

Tab.2 ukazuje dobu výpočtu, změní-li se velikost prvků vstupní matice. Místo nula-jedničkového rozdělení byl zvolen výběr z množiny $\{x \in Z; |a| \leq 100\}$ s rovnoměrným rozdělením.

N	GO	GFF	GF2	N	GO	GFF	GF2
20	0,044	0,005	0,002	60	3,600	0,295	0,191
25	0,110	0,011	0,009	65	4,987	0,404	0,263
30	0,216	0,022	0,012	70	6,743	0,541	0,352
35	0,407	0,038	0,023	75	9,263	0,743	0,482
40	0,712	0,063	0,040	80	-	1,108	0,697
45	1,135	0,098	0,063	85	-	1,210	0,793
50	1,715	0,145	0,095	90	-	1,595	1,004
55	2,534	0,209	0,136	100	-	2,445	1,617

tab.2

Vstupem pro následující test byla opět matice, jejíž prvky mají nula-jedničkové rozdělení. Tab.3 ukazuje výsledek pro řídké matice, kde $p = \frac{2N}{N^2}$. Tab.4 obsahuje výsledky pro husté matice, kde $p = 1 - \frac{2N}{N^2}$.

N	GO	GFF	GF2	N	GO	GFF	GF2
40	0,110	0,010	0,007	70	0,584	0,038	0,025
45	0,144	0,015	0,009	75	0,723	0,044	0,029
50	0,237	0,020	0,012	80	0,861	0,058	0,052
55	0,283	0,026	0,014	85	1,001	0,062	0,042
60	0,326	0,028	0,018	90	1,213	0,076	0,054
65	0,492	0,034	0,022	100	1,512	0,084	0,060

tab.3

N	GO	GFF	GF2	N	GO	GFF	GF2
20	0,030	0,002	0,002	60	0,795	0,062	0,042
25	0,048	0,004	0,003	65	1,024	0,091	0,064
30	0,084	0,005	0,005	70	1,335	0,124	0,082
35	0,138	0,010	0,007	75	1,710	0,157	0,109
40	0,210	0,015	0,012	80	2,189	0,203	0,135
45	0,302	0,023	0,017	85	2,744	0,254	0,169
50	0,425	0,034	0,025	90	3,406	0,311	0,212
55	0,580	0,047	0,034	100	5,227	0,495	0,334

tab.4

4.2 Test algoritmu HOM

Vstupem pro tento test byla matice, jejíž prvky mají nula-jedničkové rozdělení s pravděpodobností $p = 0,5$. Kromě velikosti vstupu a časového údaje obsahuje tab.5 v sloupci k počet prvočísel použitých při výpočtu. Pro porovnání byl na stejné vstupy aplikován také algoritmus GF2.

N	HOM	k	GF2	N	HOM	k	GF2
20	0,000	1	0,000	105	1,091	6	0,340
25	0,010	1	0,000	110	1,221	6	0,420
30	0,020	1	0,010	115	1,352	6	0,490
35	0,030	1	0,010	120	1,710	7	0,560
40	0,050	2	0,010	125	1,892	7	0,711
45	0,070	2	0,010	130	2,083	7	0,821
50	0,080	2	0,030	135	2,543	8	0,951
55	0,110	2	0,030	140	2,814	8	1,081
60	0,160	3	0,050	145	3,084	8	1,231
65	0,210	3	0,060	150	3,725	9	1,482
70	0,240	3	0,070	155	4,045	9	1,602
75	0,340	4	0,100	160	4,857	10	1,872
80	0,400	4	0,120	165	5,227	10	2,103
85	0,460	4	0,160	170	6,129	11	2,373
90	0,530	5	0,180	180	7,170	11	2,944
95	0,731	5	0,230	190	8,953	12	3,755
100	0,841	5	0,270	200	10,895	13	4,116

tab.5

Pro $n = 200$ je podle lemmatu 2.12 hodnota determinantu nula-jedničkové matice nejvýše rovna $d = 200^{100} \doteq 1,28.10^{230}$. Znamená to, že potřebujeme zhruba 24 prvočísel o délce 32 bitů, aby platilo $2d < \prod p_k$. V testovaném případě nám ale stačilo pouze 13 prvočísel.

Kapitola 5

Závěr

Práce obsahuje popis několika algoritmů pro řešení soustav lineárních rovnic s celočíselnými koeficienty. Prvním je algoritmus klasické Gaussovy eliminace, který nijak nevyužívá celočíselnosti a pracuje v tělese $\mathbb{Q}$. Každá operace prováděná v $\mathbb{Q}$ vyžaduje výsledné zlomky zkracovat na základní tvar, aby jejich velikost nerostla neúnosně. S tím je spojen výpočet největšího společného dělitele čitatele a jmenovatele zlomku, což se nepříznivě projeví na době výpočtu.

Dalším algoritmem je Gaussova eliminace bez dělení, kde již veškeré výpočty probíhají v $\mathbb{Z}$, ale prudce se zvyšující délka koeficientů činí tento algoritmus pro tento obor nepoužitelným. Uplatnění by však mohl najít například v tělese $\mathbb{Z}_p$.

Bezzlomková Gaussova eliminace nebo též Barreisův algoritmus se ukázal být nejrychlejším algoritmem na všech testovaných datech. Vychází z předchozího algoritmu, ale dělením koeficientů během eliminace zamezuje jejich růstu. Algoritmus byl dále vylepšen spojením dvou kroků eliminace v jeden, čímž byla doba výpočtu zkrácena zhruba o 40%.

Modulární algoritmus byl v testech zhruba třikrát pomalejší než bezzlomková Gaussova eliminace, navíc dokáže řešit pouze soustavy, jejichž matice jsou regulární, což značně omezuje jeho aplikovatelnost. Pro nalezení řešení využívá Crammerovo pravidlo a determinanty původních matic rekonstruuje pomocí čínské věty o zbytcích. Pro odhad velikosti determinantu je použita Hadamardova nerovnost.

Literatura

[1] Geddes, K. O., Czapor, S. R., Labahn, G.: *Algorithms for Computer Algebra*, Kluwer Academic Publishers, Boston, 1992.

[2] F. Winkler: *Polynomial Algorithms in Computer Algebra*, Springer, New York, 1996.

[3] Bican, L.: *Lineární algebra a geometrie*, Academia, Praha, 2002.

[4] R. A. Horn, C. R. Johnson: *Matrix Analysis* , Cambridge University Press, 1985.