TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra technické kybernetiky

JIŘÍ LASÍK

Algoritmus syntézy řízení pro vícerozměrný ARMAX model

Diplomová práce

1996

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra technické kybernetiky Školní rok: 1995/96

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro Jiřího LASÍKA

obor 23-40-8 Automatizované systémy řízení ve strojírenství

Vedoucí katedry Vám ve smyslu zákona č. 172/1990 Sb. o vysokých školách určuje tuto diplomovou práci:

Název tématu:

ALGORITMUS SYNTÉZY ŘÍZENÍ PRO VÍCEROZMĚRNÝ **ARMAX** MODEL

Zásady pro vypracování:

1. Determinant maticového polynomu.
2. Faktorizace skalárního polynomu.
3. Řešení maticové diofantické rovnice.
4. Největší společný dělitel maticových polynomů.

OBSAH

A. ÚVOD

B. TEORIE K ZADANÉMU PROBLÉMU 5
1 Jednorozměrný ARMAX model 5
2 Mnohorozměrný ARMAX model 6
3 Syntéza řízení mnohorozměrného ARMAX modelu 7

C. ZPRACOVÁNÍ ZADANÉHO PROBLÉMU 9
1 Stanovení problému 9
2 Kroky při řešení 10
2.1 Úvod a postup 10
2.2 Vytvoření základního algoritmu 11
2.3 Úplný algoritmus pro hlavní program 11
3 Determinant maticového polynomu 12
3.1 Úvod 12
3.2 Postup výpočtu 12
3.3 Vytvoření základního algoritmu 13
3.4 Úplný algoritmus na výpočet determinantu 14
4 Faktorizace skalárního polynomu 17
4.1 Úvod 17
4.2 Postup při výpočtu faktorizace 17
4.3 Základní algoritmus pro faktorizaci polynomu 17
4.4 Reflexe polynomu 18
4.4.1 Úvod do reflexe polynomu 18
4.4.2 Postup výpočtu reflexe 19
4.4.3 Základní algoritmus výpočtu 22
4.5 Největší společný dělitel 2 polynomů 23
4.5.1 Úvod a postup výpočtu 23
4.5.2 Základní algoritmus výpočtu 24

4.6 Úplný algoritmus faktorizace skalárního polynomu 25
5 Největší společný dělitel maticových polynomů 30
5.1 Úvod 30
5.2 Společný pravý dělitel 2 maticových polynomů 30
5.2.1 Postup při výpočtu 30
5.2.2 Základní algoritmus výpočtu 31
5.3 Společný levý dělitel 2 maticových polynomů 32
5.3.1 Úvod a postup 32
5.3.2 Základní algoritmus výpočtu 33
5.4 Úplný algoritmus pro výpočet společného dělitele 34
6 Syntéza řízení mnohorozměrného ARMAX modelu 39
6.1 Úvod 39
6.2 Postup při řešení 39
6.3 Základní algoritmus výpočtu 41
6.4 Základní algoritmus pro inverzní matici 42
6.5 Postup řešení při dělení maticových polynomů 44
6.6 Úplný algoritmus na syntézu řízení ARMAX modelu 47

D. ZÁVĚR 50

LITERATURA

PŘÍLOHA

PROGRAM NA VÝPOČET DETERMINANTU

PROGRAM S POUŽITÝMI FUNKCEMI A PROCEDURAMI

PŘILOŽENÉ JEDNODUCHÉ PŘÍKLADY

A. ÚVOD

V dnešní době je široce rozšířen trend zavádění osobních počítačů, nebo počítačových sítí na různá pracoviště. Účelem je odstranění nutné stereotypní činnosti, ulehčení práce, řízení výrobních procesů, nebo řízení a kontrola činnosti celých podniků, nemocnic, úřadů , škol a pod. Proto se také na všech školách (středních i vysokých) zintenzivnila výuka obsluhy PC i počítačových sítí. Výuka práce s uživatelskými programy i výuka samostatného programování, a to na všech možných úrovních. V rámci tohoto trendu se na Katedře technické kybernetiky Technické univerzity v Liberci již tradičně zadávají diplomové práce, které mají za úkol programově zpracovat různé vědecké zprávy, nebo navrhovat řešení teoretických matematických problémů.

Má diplomová práce vychází a programově zpracovává zprávu, zabývající se popisem mnohorozměrného ARMAX modelu a teorií algebraické syntézy jeho řízení. Je v ní popsána syntéza řízení při kvadratickém kritériu, při dané vstupní poruše.

Úkolem diplomové práce je z popsané teorie vytvořit vhodné algoritmy řešení. Algoritmy pak přepsat do programů, aby při zadání rovnice mnohorozměrného ARMAX modelu na vstupu byly na výstupu rovnice popisující regulátor, kterým můžeme řídit i se zadanou vstupní poruchou.

Při vytváření algoritmů jsem použil i jiné matematické metody. Pro výpočet determinantu metodu řádkových redukcí. Pro výpočet faktorizací upravený test stability, Newtonovu metodu iteračních kroků, nalezení společného dělitele 2 polynomů. V ostatních algoritmech jsou tyto metody také použity, ale v pozměněné formě.

Předpokládám, že výsledná diplomová práce najde využití jako pomůcka k ulehčení výpočtů, nebo jako základ při zpracovávaní jiného matematického problému.

B. TEORIE K ZADANÉMU PROBLÉMU

1 JEDNOROZMĚRNÝ ARMAX MODEL

Jednorozměrný ARMAX model můžeme popsat diferenční rovnicí (pro názornost i stručnost jsme zde zvolili konkrétní příklad)

1" $y(n)=-a_1*y(n-1)-a_2*y(n-2)-a_3*y(n-3)+b_0*u(n-j)+b_1*u(n-j-1)+c_0*d(n)+c_1*d(n-1).$

V této rovnici je y(n) výstupní veličina, u(n) je ovladatelný vstup a d(n) je porucha. Rovnici můžeme také zapsat v obrazovém zápisu

2" $A*y=q^j*B*u+C*d,$

kde y, u a d jsou obrazy posloupností y(n), u(n) a d(n), tj. např. $y=\acute{I}y(n)*q^n$. Obrazy A, B, C mají pak tyto prvky

3" $A=1+a_1*q+a_2*q^2+a_3*q^3$

$B=b_0+b_1*q$

$C=c_0+c_1*q$.

Existují také speciální případy ARMAX modelu :

4" $A*y=d$ se nazývá AR model a

5" $y=C*d$ se nazývá MA model.

2 MNOHOROZMĚRNÝ ARMAX MODEL

Základem pro mnohorozměrný ARMAX model je model popsaný podobnou rovnicí jako v předchozí kapitole. Všechny veličiny v ní jsou ale čtvercové matice o rozměru $n_x n$ a tedy je píšeme malými tučnými písmeny u koeficientů, nebo podtrženými malými písmeny u proměnných. Tedy

6" $y(n)=-\mathbf{a}_1*y(n-1)-\mathbf{a}_2*y(n-2)-\mathbf{a}_3*y(n-3)+\mathbf{b}_0*u(n)+\mathbf{b}_1*u(n-1)+\mathbf{c}_0*d(n)+\mathbf{c}_1*d(n-1)$

Podobně jako v jednorozměrném případě můžeme i tady rovnici přepsat do obrazového zápisu.

Takže definujme

7" $\mathbf{A}=E+\mathbf{a}_1*q+\mathbf{a}_2*q^2+\mathbf{a}_3*q^3$

$\mathbf{B}=\mathbf{b}_1+\mathbf{b}_1*q$

$\mathbf{C}=\mathbf{c}_0+\mathbf{c}_1*q$.

Stupně maticových polynomů **A**, **B**, **C** jsou samozřejmě voleny konkrétně, a to kvůli názornosti i stručnosti. V obrazovém zápisu pak rovnice mnohorozměrného ARMAX modelu vypadá takto

8" $\mathbf{A}*y=q^j*\mathbf{B}*u+\mathbf{C}*d.$

Veličiny y, u, d jsou obrazy posloupností y(n), u(n), d(n). Jsou tvořeny podobným vztahem jako v jednorozměrném případě.

3 SYNTÉZA ŘÍZENÍ MNOHOROZMĚRNÉHO ARMAX MODELU

Uvažujeme tedy mnohorozměrný ARMAX model popsaný rovnicí

9“ $\mathbf{A}*y=q^{j}*\mathbf{B}*u+\mathbf{C}*d.$

Pak můžeme odvodit pro y

10“ $y=q^{j}*\mathbf{A}^{-1}*\mathbf{B}*u+\mathbf{A}^{-1}*\mathbf{C}*d.$

Úkolem je najít takové u stabilní, aby bylo i y stabilní. Hledáme tedy přenos r, pro který platí u=r*y, nebo-li $r=u*y^{-1}$. Jestliže najdeme stabilní u i y, bude přenos r fyzikálně realizovatelný, tzn. že u(n) nezávisí na budoucích hodnotách y(n+1),... Pro řešení použijeme kritérium minimalizace funkcionálu <y,y>.

Vyjdeme z předpokladu, že **B** je obecně nestabilní a $B_2=B_-$ je minimální nestabilní faktor maticového polynomu **B**. Pak součin $\mathbf{B}_{11}^{-1}*\mathbf{B}_1$ je minimální stabilní faktor maticového polynomu **B**, nebo-li

11“ $\mathbf{B}=B_2*\mathbf{B}_{11}^{-1}*\mathbf{B}_1.$

Rovnici 7“ pak můžeme zapsat ve tvaru

12“ $\mathbf{A}*y=q^{j}*B_2*\mathbf{B}_{11}^{-1}*\mathbf{B}_1*u+\mathbf{C}*d.$

Pro y pak tuto rovnici přepíšeme do tvaru

13“ $y=\mathbf{A}*(q^{j}*B_2*\mathbf{B}_{11}^{-1}*\mathbf{B}_1*u+\mathbf{C}*d).$

Jestliže si pro u definujeme

14“ $u=\mathbf{B}_1^{-1}*\mathbf{B}_{11}*\mathbf{g}*\mathbf{C},$

pak můžeme napsat rovnici pro y takto

15“ $y=\mathbf{A}^{-1}*(\mathbf{E}*d+q^{j}*B_2*\mathbf{g})*\mathbf{C}.$

Tyto obě rovnice dosadíme do vztahu $r=u*y^{-1}$ a získáme

16“ $r=\mathbf{B}_1^{-1}*\mathbf{B}_{11}*[\mathbf{g}/(\mathbf{E}*d+q^{j}*B_2*\mathbf{g})]*\mathbf{A}.$

Z tohoto vztahu plyne, že za předpokladu stabilních **A**, **C** a stabilního **g** jsou stabilní i u a y. Úlohu tedy můžeme definovat takto,

najdi pro $g\in M_+$ min $\langle y,y\rangle = \langle \mathbf{A}^{-1}*(\mathbf{E}*d+q^{j}*B_2*\mathbf{g})*\mathbf{C}, \mathbf{A}^{-1}*(\mathbf{E}*d+q^{j}*B_2*\mathbf{g})*\mathbf{C}\rangle$.

Pak po několika odvozeních a dosazeních (pro **A** nesoudělné s B_2), při definovaných předpokladech a při splnění kritéria minimalizace funkcionálu $\langle y \rangle^2$ je optimální regulátor popsán soustavou

17“ $$\mathbf{V}^*z=y$$

$$\mathbf{B}_1^*u=-\mathbf{B}_{11}^*\mathbf{T}^*z,$$

kde maticové polynomy **T**, **V** získáme z lineární diofantické rovnice

18“ $$\mathbf{C}^*\sim B_2=\mathbf{T}^*q^j*B_2+\mathbf{A}^*\mathbf{V}.$$

Naším úkolem bude najít takový algoritmus pro vytvoření programu, aby ten po zadání prvků rovnice 9“ vypočítal a vypsal rovnice 17“.

C. ZPRACOVÁNÍ ZADANÉHO PROBLÉMU

1 STANOVENÍ PROBLÉMU

Z teorie o řízení mnohorozměrného ARMAX modelu vyplývá, že naším úkolem je najít vhodný algoritmus pro výpočet maticových polynomů **T,V** . Pomocí nich potom vytvoříme rovnice, které určují, jaké bude u, aby y bylo stabilní. Zkrácený tvar těchto rovnic vypadá takto

1.1“ $$\mathbf{V}*z=y\ ,$$

$$\mathbf{B}_1*u=-\mathbf{B}_{11}*\mathbf{T}*z.$$

Z celého postupu výpočtu pro vytvoření těchto rovnic využijeme na tvorbu algoritmu jen pro nás potřebné části. Zároveň budeme muset vytvořit další pomocné algoritmy (= pomocné programy). Mezi tyto patří program na výpočet determinantu maticového polynomu, program na faktorizaci skalárního polynomu, program na nalezení největšího společného dělitele maticových polynomů. Pomocné programy budou vytvořeny tak, aby mohly být používány i samostatně. To znamená, že uživatel si potom může vybrat z nabídky, kterou budou tvořit tyto body: determinant maticového polynomu (polynomiální matice), faktorizace skalárního polynomu, nalezení největšího společného dělitele 2 maticových polynomů, nalezení matic T,V a vytvoření rovnic 1.1“ pro řízení mnohorozměrného ARMAX modelu.

2 KROKY PŘI ŘEŠENÍ

2.1 ÚVOD A POSTUP

Jak už jsem uvedl v předchozí kapitole, chceme, aby měl uživatel našeho programu při jeho spuštění možnost výběru ze základní nabídky:

1) Determinant maticového polynomu,
2) Faktorizace skalárního polynomu,
3) Největší společný dělitel matic. polynomů,
4) Nalezení rovnic pro řízení daného mnohorozměrného ARMAX modelu.

Pokud si uživatel vybere první možnost, program vypočítá determinant ze zadaného maticového polynomu (případně polynom. matice) a znovu nabídne výběr ze 4 základních bodů. Uživatel si může samozřejmě vybrat bod číslo 2. Po této volbě program provede faktorizaci zadaného skalárního polynomu, tzn. rozdělí polynom na součin jeho stabilní a nestabilní části. A opět si bude moci vybrat z hlavního menu. Při výběru možnosti číslo 3 program nalezne největšího společného dělitele 2 zadaných maticových polynomů. Pokud si uživatel vybere z nabídky 4. možnost, program vypočte maticové polynomy T,V a vytvoří rovnice pro řízení mnohorozměrného ARMAX modelu, který bude zadán maticovými polynomy A,B,C a dalším popisem. V této části (rozuměj 4.) jsou zahrnuty spolu s dalšími pomocnými algoritmy i tři předchozí. Je samozřejmé, že v každé části programu bude možnost předčasného ukončení a návratu do hlavního menu, případně úplné ukončení programu.

2.2 Vytvoření základního algoritmu

Základní algoritmus na hlavní program bude krátký. Jde vněm pouze o poskytnutí 4 možností výběru , výběr jedné z nich a možnost návratu opět na začátek.

Zápis algoritmu :

1) Vyber si ze 4 možností : a) determinant , b) faktorizace, c) největší společný dělitel, d) syntéza řízení

2) Je_li vybráno a) pak zavolej program na výpočet determinantu maticového polynomu a po jeho ukončení jdi na 1)

3) Je_li vybráno b) , zavolej program na výpočet faktorizace skalárního polynomu a po jeho skončení jdi na 1)

4) Je-li vybráno c) , zavolej program na výpočet největšího společného dělitele maticových polynomů a po jeho ukončení jdi na 1)

5) Je-li vybráno d), pak zavolej program na syntézu řízení a po jeho ukončení jdi na 1)

2.3 Úplný algoritmus na hlavní program

Úplný algoritmus se od základního liší minimálně. Podprogramy, které reprezentují jednotlivé možnosti výběru jsou zapsány jako procedury v Unitech a budou tak i volány. Po ukončení práce v těchto procedurách se bude hlavní program vracet na místo, kde je základní nabídka.

Zápis algoritmu:

1) Vypiš hlavičku hlavního programu a vypiš nabídku

...

Body 2) až 6) jsou shodné z body 1) až 5) v kapitole 2.2, až na malou změnu -> po skončení práce v procedurách se vrať na 2)

7) Je-li vybráno x), pak konec

8) Jdi na 2)

3 DETERMINANT MATICOVÉHO POLYNOMU

3.1 Úvod

Determinant matice číselné nebo polynomiální (případně maticového polynomu) se nejčastěji používá při výpočtu inverzní matice. Dá se však využít i k samostatnému účelu, např. určení stability maticového polynomu. My ho v našem případě použijeme v algoritmu na výpočet a určení rovnic pro řízení ARMAX modelu. Determinant lze počítat pouze z čtvercových matic, nebo čtvercových maticových polynomů. S tímto předpokladem jsme hledali vhodný postup při výpočtu. K vytvoření algoritmu jsme našli a použili metodu jednostranných řádkových redukcí polynomiální matice.

3.2 Postup výpočtu

Tato metoda jednostranných řádkových redukcí využívá následující postup. Nejprve určíme nenulový polynom nejmenšího stupně v prvním sloupci zadané matice (jako příklad ji označme A). Pokud není hledaný polynom na prvním řádku, zaměníme první řádek s řádkem, ve kterém se tento polynom nalézá. Tím tento polynom přemístíme na hlavní diagonálu matice A.

Pak odčítáme takové násobky tohoto polynomu od zbývajících polynomů v prvním sloupci, abychom snížili jejich stupně. Pokud je po tomto kroku v prvním sloupci polynom nižšího stupně, než polynom v prvním řádku a prvním sloupci, opět provedeme výměnu řádků tak, aby byl na hlavní diagonále (v prvním sloupci) polynom nejnižšího stupně. Opakováním tohoto postupu nakonec anulujeme všechny polynomy pod hlavní diagonálou v prvním sloupci matice. Dále nalezneme nenulový polynom nejnižšího stupně v druhém sloupci a řádcích počínaje druhým. Opět vyměníme řádky tak, abychom tento polynom přemístili na hlavní diagonálu matice A. Potom všechny zbývající polynomy druhého sloupce pod hlavní diagonálou redukujeme už výše uvedeným způsobem na nulu.

Takto postupujeme ve všech sloupcích matice (kromě posledního) a nakonec obdržíme původní matici A ve tvaru horní trojúhelníkové matice. Jelikož je každá z uvedených řádkových transformací ekvivalentní násobení matice zleva

nějakou jednotkou, determinant původní matice se redukcemi nemění. Vzhledem k tomu, že původní matice A byla upravena na trojúhelníkovou, její determinant je dán součinem jejích prvků na hlavní diagonále násobených s-mocninou konstanty (-1), kde s má stejnou hodnotu, jako je počet výměn řádků při redukci matice.

3.3 Vytvoření základního algoritmu

Před vytvořením úplného algoritmu pro snadnější přepis do programu přepíšeme postup z podkapitoly C.3.2 do základního algoritmu, který ukáže jednotlivé kroky názorněji.

Zápis algoritmu:

(1) k=0,s=0

(2) k=k+1. Jestliže k=m, jděte na (9)

(3) Určete polohu nenulového polynomu a_{ik} matice A a položte s=s+1

(4) Jestliže se i nerovná k, zaměňte řádky i a k matice A a položte s=s+1

(5) Je-li pouze jeden nenulový polynom ve sloupci k a řádcích k,k+1,...,m matice A, jděte na (2)

(6) Určete čísla $\lambda_{k+1},...,\lambda_m$ jako podíly koeficientů u nejvyšších mocnin $q=z^{-1}$ a čísla $d_{k+1},...,d_m$ jako rozdíly stupňů polynomů ve sloupci k a řádcích k+1,...,m a polynomu ve sloupci k a řádku k matice A

(7) Odečtěte řádek k násobený $\lambda_{k+1}*q^{d(k+1)}=\lambda_{k+1}*z^{-d(k+1)},...,\lambda_m*q^m$ od řádků k+1,...,m

(8) Jděte na (9)

(9) Vypočtěte součin $a=(-1)^S*a_{11}*a_{22}*...*a_{mm}$

(10) Je-li k=1, konec.

Na konci tohoto algoritmu je výsledek takový: a=detA. Tento algoritmus dále upravíme a rozšíříme tak, aby bylo možno podle něj napsat program v TP.

3.4. Úplný algoritmus na výpočet determinantu

V algoritmu jsou svými jmény volány určité důležité procedury, které v něm ale nejsou popsány. Proto jsou zapsány za algoritmem. Řádky, v nichž volám tyto procedury jsou označeny *.

Zápis algoritmu:

1) Bude na vstupu zadáván **M**aticový polynom nebo **P**olynomiální matice ?

2) M : Zadejte rozměr matic v maticovém polynomu -> rm , nebo

P : Zadejte rozměr polynomiální matice -> rm.

3) M : Zadejte maximální stupeň maticového polynomu -> sp , nebo

P : Zadejte maximální stupeň polynomu v polynomiální matici -> sp

4)*M : Načtěte prvky matic maticového polynomu, nebo

P : Načtěte prvky polynomiální matice

5) Vypište zadaný maticový polynom převedený na polynomiální matici, nebo vypište zadanou polynomiální matici

6) k=1, s=0 (počet výměn řádků)

7) Jestliže k=rm, jděte na 17)

8)* Vyhledejte nenulový polynom a_{ik} nejnižšího stupně ve sloupci k a řádcích i=k,k+1...rm a číslo řádku uložte do i1

9) Jestliže i1=0 je determinant = 0, Konec.

10)* Jestliže se $i1 \neq k$, zaměňte řádek i1 a k matice a položte s=s+1

11) Vypište polynomiální matici po výměně řádků

12)* Zjistěte počet nenulových polynomů ve sloupci k polynomiální matice a zapište ho do i1

13) Jestliže i1=1, ukončete cyklus redukce ve sloupci k a položte k=k+1 a jděte na 7)

14)* Redukujte polynomiální matici

15) Vypište polynomiální matici po redukci

16) Jděte na 7)

17) Vypište výslednou polynomiální matici

18)* Vypočtěte součin det = $a_{11}*a_{22}*...*a_{rm,rm}$

19) Proveďte det = $(-1)^s$ * det

20) Vypište determinant polynomiální matice = det, Konec.

Volané procedury:

*Načítání prvků

M -> Pro q=0..sp, i=1..rm, j=1..rm indexujeme prvky při načítání [q, i, j]. Potom máme v matici q na pozici i,j koeficent pro mocninu q.

P -> Pro i=1..rm, j=1..rm, q=0..sp indexujeme prvky při načítání [i, j, q]. Potom máme na pozici i,j celý polynom se stupněm a=0..sp.

*Vyhledej polynom nejnižího stupně

řádek=0, stupeň=11

Pro i=k...rm přiřaďtedo a stupeň polynomu ve sloupci k a řádcích i. Jestliže je a=>0 a menší než stupeň, pak přiřaďte stupeň=a, řádek=i. Na konci této procedury je v proměnné řádek číslo řádku s nejmenším nenulovým polynomem ve sloupci k. To pak položíme = i1.

*Počet nenulových polynomů

počet=0. Pro i=k...rm přiřaďte do a stupeň polynomu ve sloupci k a řádku i. Jestliže a=>0 pak počet=počet+1. Na konci této procedury je v proměnné počet množství nenulových polynomů ve sloupci k.

*Vyměň řádky

Pro j=1...rm a q=0...sp b=koeficient u mocniny q v prvku[i1, j], koeficient u mocniny q v prvku [i1, j]=koeficient u mocniny q v prvku [i, j] , koeficient u mocniny q v prvku [i, j]=b.

*Redukce matice

Pro i=k+1...rm určete čísla l_i jako podíly koeficientů u nejvyšších mocnin q a čísla d_i jako rozdíly stupňů polynomů ve sloupci k a řádcích i a polynomu ve sloupci k a řádku k. Pro i=k+1...rm, j=k...rm vynásobte polynom na pozici [k,j] číslem l_i*q^{di} a odečtěte ho od polynomu na pozici [i, j].

*Součin na diagonále (součin polynomů)

Pro i=0...sp, j=0...sp c[i+j]=c[i+j]+a[i]*b[j]

4 FAKTORIZACE SKALÁRNÍHO POLYNOMU

4.1 ÚVOD

Faktorizací nějakého skalárního polynomu a rozumíme jeho rozklad na součin stabilní a nestabilní části, tj. $a=a_+*a_-$. Stabilita skalárních polynomů se dá bezpečně určit z jejich kořenů. Ale matematický výpočet kořenů u polynomů s řádem vyšším než 3 je velmi časově náročný, složitý a nejistý. Proto jsme zvolili jiný postup.

4.2 POSTUP PŘI VÝPOČTU FAKTORIZACE

Postup vychází z upraveného testu stability polynomu, ve kterém redukční koeficienty λ určují stabilitu polynomu. Jsou-li všechny $|\lambda_i| <= 1$, je celý polynom stabilní a můžeme napsat, že $a=a_+$. Jsou-li naopak všechny λ_i v absolutní hodnotě větší než 1, můžeme napsat, že $a=a_-$. Pokud jsou λ_i v absolutní hodnotě různé od jedné (tzn. jednou větší, jednou menší,...), je stabilní reflexe polynomu a, kterou označíme a^*. Potom je a_+ největším společným dělitelem polynomů a, a^* (tzn. $a_+=[a,a^*]$). Nestabilní část a_- pak vypočteme jako podíl a, a_+ (tzn. $a_-= a\backslash a_+$).

4.3 ZÁKLADNÍ ALGORITMUS PRO FAKTORIZACI POLYNOMU

Postup při výpočtu faktorizace přepíšeme do jednoduchého algoritmu, který názorně ukáže jednotlivé kroky. Řádky v nichž je zanesen upravený test stability označím navíc hvězdičkou.

Zápis algoritmu:

*(1) Zadej $a=a_l*q^l+a_{l+1}*q^{l+1}+...+a_n*q^n$, a_l se nerovná 0. Potom určete polynom $p=a_l+a_{l+1}*q+...+a_n*q^{n-l}$

*(2) i=0

*(3) s=0

*(4) Je-li i=n-l, jděte na (16)

*(5) Položte $r=\sim p*q^{(n-l-1-\partial p)}$, kde ∂p je stupeň polynomu p a ~p je polynom p upravený(viz př.: $b=1+2*q-2*q^2$, pak $\sim b=q^2+2*q-2$)

Takto jsme získali prvky matice [P*T], která je horní trojúhelníkovou maticí. Pak můžeme rovnici P*T*X=Q zapsat takto:

4.4.8“

$$\begin{vmatrix} 2*\sigma_{p,0} & 0 & \dots & 0 & 0 \\ \sigma_{p,0} & \dots & \sigma_{2,p-2} & \sigma_{1,p-1} & \\ \sigma_{p-1,0} & \dots & \sigma_{2,p-3} & \sigma_{1,p-2} & \\ \dots & \dots & \dots & \dots & \\ & & \sigma_{2,0} & \sigma_{1,1} & \\ & & & \sigma_{1,0} & \end{vmatrix} * \begin{vmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \dots \\ \dots \\ \xi_p \end{vmatrix} = \begin{vmatrix} 2*\gamma_0 \\ 2*\gamma_1 \\ 2*\gamma_2 \\ \dots \\ \dots \\ 2*\gamma_p \end{vmatrix}$$

Z této rovnice snadno vypočítáme X a to dosazováním vždy vypočítan prvku matice X do vztahu o řádek výše. Pak už jen zbývá vypočítat řešení X pomocí zpětné transformace. Toto řešení získáme podobným rekurentním sc matem (B) jako bylo schéma A.

Schéma B: $\xi_0 \quad \xi_1 \quad \xi_2 \quad \dots \quad \xi_p$

$\xi_1 \quad \xi_0 \quad 0 \quad \dots \quad 0$ | $*\lambda_{p-1}$

$\xi_{1,0} \quad \xi_{1,1} \quad \xi_{1,2} \quad \dots \quad \xi_{1,p}$

$\xi_{1,2} \quad \xi_{1,1} \quad \xi_{1,0} \quad \dots \quad 0$ | $*\lambda_{p-2}$

$\dots \quad \dots \quad \dots \quad \dots \quad \dots$

$\xi_{p-1,0} \quad \xi_{p-1,1} \quad \xi_{p-1,2} \quad \dots \quad \xi_{p-1,p}$

$\xi_{p-1,p} \quad \xi_{p-1,p-1} \quad \xi_{p-1,p-2} \quad \dots \quad \xi_{p-1,0}$ | $*\lambda_0$

$\xi_{p,0} \quad \xi_{p,1} \quad \xi_{p,2} \quad \dots \quad \xi_{p,p}$

Poslední řádek schématu B obsahuje řešení rovnice P*X=Q. To znamená, že lynom x^i má tvar

4.4.9“ $x^i=\xi_{p,0}+\xi_{p,1}*q+\dots+\xi_{p,p}*q^p$.

ních kroků, kterými se přiblížíme k a^* s požadovanou přesností (př.: jestliže se $\varepsilon=10^{-9}$, pak se počítá tak dlouho, dokud se jednotlivé iterace po sobě jdoucí neliší až v desátém místě za desetinnou čárkou).

4.4.2 Postup výpočtu reflexe

Označme si i-tou iteraci reflexe polynomu $\underline{a}$ jako s^i. Potom můžeme vytvořit výraz $f=s^i*s^i-\underline{a}*\underline{a}$, kde polynomy $\underline{a}$, s^i jsou upravené polynomy $\underline{a}$, s^i (viz př.: $b=q+3*q^2-2*q^3$ a $b=q^{-1}+3*q^{-2}-2*q^{-3}$). Princip Newtonovy metody vychází z rovnice df+f=0. Tato rovnice po dosazení substituce

4.4.1" $$s^{i+1}=0,5*(s^i+x^i)$$

a po provedení úprav nabývá tvaru

4.4.2" $$s^i*x^i+x^i*s^i=2*\underline{a}*\underline{a} \quad i=1,2,\ldots .$$

Potom platí limita $\lim_i s^i=a^*$, kde i jde do nekonečna. V každém iteračním kroku musíme proto řešit speciální rovnici 4.4.2" pro polynomy x^i,x^i s vedlejší podmínkou, že stupeň polynomu x^i je stejný jako stupeň s^i.

Na začátku si musíme zvolit nějaký určitě stabilní polynom, z kterého budeme pomocí iteračních kroků počítat a^*. Nejlépe je vycházet z tohoto

4.4.3" $$s^i=s^0=(1/\sqrt{\gamma_0})*(\gamma_0+\gamma_1*q+\ldots+\gamma_p*q^p),$$

kde koeficienty γ_i (i=1,2,...) jsou koeficienty z rovnice

4.4.4" $$\underline{a}=\gamma_p*q^p+\ldots+\gamma_1*q+\gamma_0+\gamma_1*q^{-1}+\ldots+\gamma_p*q^{-p},$$

kde se γ_p nerovná nule. Po vydělení koeficientů γ_i zlomkem $1/\sqrt{\gamma_0}$ ve vztahu 3" můžeme s^0 a x^0 přepsat do tvaru:

4.4.5" $$s^i=s^0=\sigma_0+\sigma_1*q+\ldots+\sigma_p*q^p ,$$

4.4.6" $$x^i=x^0=\xi_0+\xi_1*q+\xi_2*q^2+\ldots+\xi_p*q^p.$$

Takto jsme získali prvky matice [P*T], která je horní trojúhelníkovou maticí.

Pak můžeme rovnici P*T*X=Q zapsat takto:

4.4.8“

$$\begin{vmatrix} 2*\sigma_{p,0} & 0 & \dots & 0 & 0 \\ & \sigma_{p,0} & \dots & \sigma_{2,p-2} & \sigma_{1,p-1} \\ & & \sigma_{p-1,0} & \dots \sigma_{2,p-3} & \sigma_{1,p-2} \\ & & \dots & \dots \dots & \dots \\ & & & \sigma_{2,0} & \sigma_{1,1} \\ & & & & \sigma_{1,0} \end{vmatrix} * \begin{vmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \dots \\ \dots \\ \xi_p \end{vmatrix} = \begin{vmatrix} 2*\gamma_0 \\ 2*\gamma_1 \\ 2*\gamma_2 \\ \dots \\ \dots \\ 2*\gamma_p \end{vmatrix}$$

Z této rovnice snadno vypočítáme X a to dosazováním vždy vypočítaného prvku matice X do vztahu o řádek výše. Pak už jen zbývá vypočítat řešení X z X pomocí zpětné transformace. Toto řešení získáme podobným rekurentním schématem (B) jako bylo schéma A.

Schéma B: $\xi_0 \quad \xi_1 \quad \xi_2 \quad \dots \quad \xi_p$

$\_\xi_1 \quad \xi_0 \quad 0 \quad \dots \quad 0\_ \quad | *\lambda_{p-1}$

$\xi_{1,0} \quad \xi_{1,1} \quad \xi_{1,2} \quad \dots \quad \xi_{1,p}$

$\_\xi_{1,2} \quad \xi_{1,1} \quad \xi_{1,0} \quad \dots \quad 0\_ \quad | *\lambda_{p-2}$

$\dots \quad \dots \quad \dots \quad \dots \quad \dots$

$\xi_{p-1,0} \quad \xi_{p-1,1} \quad \xi_{p-1,2} \quad \dots \quad \xi_{p-1,p}$

$\_ \xi_{p-1,p} \quad \xi_{p-1,p-1} \quad \xi_{p-1,p-2} \quad \dots \quad \xi_{p-1,0}\_ \quad | *\lambda_0$

$\xi_{p,0} \quad \xi_{p,1} \quad \xi_{p,2} \quad \dots \quad \xi_{p,p}$

Poslední řádek schématu B obsahuje řešení rovnice P*X=Q. To znamená, že polynom x^i má tvar

4.4.9“ $\qquad x^i=\xi_{p,0}+\xi_{p,1}*q+\dots+\xi_{p,p}*q^p.$

Nová iterace s^{i+1} potom vyplyne ze vztahu 4.4.1" . Z ní pak dosazujeme koeficienty do schématu A. A takto počítáme, dokud se koeficienty s^i a s^{i+1} mění pouze až za mezí přesnosti (př.: na 10-tém desetinném místě).

4.4.3 Základní algoritmus výpočtu

Opět napíšeme jednoduchý algoritmus pro výpočet reflexe polynomu a, abychom názorněji ukázali jednotlivé kroky při řešení.

Zápis algoritmu:

(1) Vypočítejte součin $\alpha*a= \gamma_p*q^p+...+\gamma_0+...+\gamma_p*q^{-p}$ a položte

$c_0=2*\gamma_0+2*\gamma_1*q+...+2*\gamma_p*q^p$

(2) $s=[1/(2*\sqrt{\gamma_0})]*c_0$

(3) i=0

(4) i=i+1, $c=c_0$

(5) Položte x=0,y=s

(6) j=0

(7) Je-li j=p, jděte na (16)

(8) Položte $r=\sim s*q^{(p-j-\partial s)}$

(9) Určete číslo λ_j jako podíl koeficientů u nejvyšších mocnin q polynomů s a r

(10) Odedečtěte λ_j-násobek polynomu r od polynomu s

(11) Určete ξ jako podíl koeficientů u (p-j)-té mocniny q polynomu c a nulté mocniny q polynomu s

(12) Přičtěte polynom $\xi*q^{(p-j)}$ k polynomu x

(13) Jestliže j< (p-1), položte $c=c-[\xi*q*(\sim s)]$

(14) j=j+1

(15) Jděte na (7)

(16) Určete číslo ξ jako polovinu podílu koeficientů u nultých mocnin q polynomů c a s

(17) Přičtěte ξ k polynomu x

(18) Je-li j=0, jděte na (23)

(19) j=j-1

(20) Nechť $x = \xi_0+\xi_1*q+..+\xi_p*q^p$, pak položte $u=\xi_0+\xi_1*q+...+\xi_p*q^{(p-j)}$

(21) $x=x-\lambda_j*(\sim u)$

(22) Jděte na (18)

(23) s=0,5*(x+y)

(24) Nechť je s ve tvaru $s=\sigma_0+\sigma_1*q+...+\sigma_p*q^p$. Jestliže $(\sum\sigma_k^2 - \gamma_0) \geq \varepsilon$ a i<N (pro k=0..p), jděte na (4), v opačném případě konec.

Po skončení algoritmu je reflexe a^* na místě polynomu s. Konstantou ε určíme už na začátku algoritmu požadovanou přesnost při iteračních krocích a zadáním konstanty N omezujeme počet iteračních kroků.

4.5. Největší společný dělitel 2 polynomů

4.5.1 Úvod a postup při výpočtu

Postup, který chceme použít na vytvoření algoritmu, umožňuje najít ke dvěma zadaným polynomům jejich největšího společného dělitele. Pro výpočet společného dělitele vycházíme ze vztahů

4.5.1“ $a*p+b*q = d$

$a*r+b*s = 0,$

kde a,b jsou zadané polynomy a d je hledaný největší společný dělitel. Polynomy p,q a r,s jsou polynomy, které splňují vztahy 4.5“ a vyplynou z výpočtu. Když uspořádáme rovnice do matic, dává jejich zápis následující tvar

4.5.2“

$$\begin{vmatrix} p & q \\ r & s \end{vmatrix} * \begin{vmatrix} a \\ b \end{vmatrix} = \begin{vmatrix} d \\ 0 \end{vmatrix}$$

Matice obsahující polynomy p,q a r,s označíme P a matici s polynomy a,b označíme M. Potom v matici M postupně snižujeme stupeň polynomu v druhém řádku jako při algoritmu na výpočet determinantu matice (pomocí redukčních čísel λ) tak, až ho úplně anulujeme. Všechny redukční operace prováděné v matici M uskutečňujeme i v matici P. Na konci výpočtu je na prvním řádku matice M polynom d a na druhém řádku je 0. V matici P jsou pak polynomy p,q a r,s na svých místech.

4.5.2 Základní algoritmus výpočtu

Postup z podkapitoly C.4.5.1 jsme zase přepsali do jednoduchého algoritmu.

Zápis algoritmu:

(1) Vytvořte matici M doplněním jejich prvků (1. řádek a, 2. řádek b)

(2) Prvky v matici P nadefinujeme takto: p=1,q=0,r=0,s=1

(3) Určete nenulový polynom menšího stupně v matici M. Jsou-li oba polynomy nulové, pak konec.

(4) Je-li polynom s menším stupněm v druhém řádku, zaměňte řádky v matici M i v matici P

(5) Je-li polynom v druhém řádku matice M nulový, konec.

(6) Určete číslo λ jako podíl koeficientů u nejvyšších mocnin q a číslo f jako rozdíl stupňů polynomů ve druhém a prvním řádku matice M

(7) Odečtěte první řádek násobený $\lambda * q^f$ od druhého řádku v maticích M i P

(8) Jděte na (3)

Po ukončení výpočtu jsou polynomy p,q a r,s na svých místech v matici P a polynom d (=největší společný dělitel a,b) je na místě polynomu a v matici M.

4.6 Úplný algoritmus faktorizace skalárního polynomu

Spolu s programem vytvoříme i 3 podprogramy, které budou zapsány jako procedury v příslušných Unitech a budou také jako procedury volány. Proto i podrobný algoritmus bude mít 3 vedlejší algoritmy, které bude volat jejich jménem.

Hlavní algoritmus:

1) Zadejte maximální stupeň polynomu (1 - 5) -> n

2) Zadejte koeficienty polynomu při mocninách i, pro i = 0,1...n -> a[i]

3) Vypište zadaný polynom

4) Zadejte počet iteračních kroků v případné reflexi -> EN

5) Volejte Stabilitu, která pro zadaný polynom určí, zda-li obsahuje neznámý nestabilní faktor (tzn. stabilita = ne, pak polynom d[i] = 0), nebo obsahuje známý, minimální nestabilní faktor (tzn. stabilita = ano, pak polynom d[i] vrací jako nenulový).

6) Jestliže Stabilita = ano , jděte na 9)

7) Vypočtěte Reflexi ra[i] polynomu a[i] , pro i = 0,1...n a EN

8) Vypočtěte Největšího společného dělitele d[i] polynomu a[i], ra[i] pro i = 0,1...n

9) Položte a+[i] = d[i] pro i = 0,1...n

10) Položte a-[i] = a[i] / d[i] pro i = 0,1...n

11) Vypište polynomy a, a+, a- . Konec.

Algoritmus Stabilita :

1) Jestliže zadané a[i] = $a_l*q^l+a_{l+1}*q^{l+1}+...+a_n*q^n$ pro $a_n \neq 0$, pak vytvořte polynom p0 = $a_l+ a_{l+1}*q+...+a_n*q^{n-l}$

2) p = p0, vypište polynom p

3) i=0, s=0

4) Je-li i = n-l, jděte na 22). Vypište i

5) Vytvořte polynom ~p a z něj pak polynom r = ~p * $q^{(n-l-i-\partial p)}$

6) Vypište polynom r

7) Určete číslo λ jako podíl koeficientů u nejvyšších mocnin q polynomů p a r

8) Vypište λ

9) Je-li |λ| => 1 jděte na 15)

10) Vynásobte polynom r číslem λa uložte výsledek do pomocného polynomu pom1

11) Odečtěte polynom pom1 od polynomu p a výsledek uložte do pom2

12) Položte p = pom2

13) s = s+1, i = i+1

14) Jděte na 4)

15) Vynásobte polynom r číslem 1/λ a uložte výsledek do pom1

16) Odečtěte polynom pom1 od polynomu p a výsledek uložte do pom2

17) Položte p = pom2

18) Uvažujme, že polynom p má teď tvar $p = p_m*q^m+p_{m+1}*q^{m+1}+...+p_{n-i}*q^{n-i}$ pro $p_m \neq 0$. Pak položme $p = p_m+p_{m+1}*q+...+p_{n-i}*q^{(n-i-m)}$

19) Vypište polynom p, vypište m

20) i= i+m

21) Jděte na 4)

22) Je_li s = n-l, položte d[i] = p0[i] a Stabilita = ano. Konec.

23) d[i] = 0 a Stabilita = ne. Konec.

Algoritmus Reflexe :

1) Ze zadaného polynomu a vytvořte polynom ~a. Vypište polynom ~a.

2) Vytvořte p jako rozdíl ∂a a nejnižší mocniny u nenulového koeficientu polynomu a.

3) Vypočtěte součin ~a *a tak, že ~aa[i+j] = ~aa[i+j] + a[i] * ~a[j] pro i,j =0,1...n

4) Vytvořte polynom c0 ze součinu ~aa tak, že c0[p-i] = 2* ~aa[i] pro i = 0,1...p

5) Vypište polynom c0

6) k =0, ε = 0.0000001

7) Položte s[i] = (1/ sqrt(c0[0]/2))*0,5*c0[i] pro i = 0,1...p

8) Vypište polynom s

9) k =k+1

10) Polynom x[i] = 0, polynom y[i] = s[i] pro i = 0,1...p

11) Polynom c[i] = c0[i] pro i = 0,1...p

12) j = 0, vytvořte polynom ~s z polynomu s

13) Je-li j = p, jděte na 27)

14) Vytvořte polynom r = ~s * $q^{(p-j-\partial s)}$

15) Vypište polynom r

16) Určete číslo λ[j] jako podíl koeficientů u nejvyšších mocnin q polynomů s a r. Vypište λ[j]

17) Vynásobte polynom r číslem λ[j] a výsledek uložte do polynomu pom

18) Odečtěte polynom pom od polynomu s a výsledek uložte do s

19) Určete číslo ξ jako podíl koeficientů u (p-j)-té mocniny q polynomu c a nulté mocniny q polynomu s. Vypište ξ

20) Přičtěte $\xi * q^{(p-j)}$ k polynomu x

21) Vypočtěte polynom ~s[i] pro i = 0,1...p-j

22) Je-li j = p-1, jděte na 25)

23) Vynásobte polynom ~s polynomem ξ* q a uložte výsledek do polynomu pom

24) Odečtěte polynom pom od polynomu c a výsledek zapište do c

25) j = j+1

26) Jděte na 13)

27) Určete číslo ξ jako polovinu podílu koeficientů u nultých mocnin q polynomů c a s

28) Přičtěte $\xi * q^0$ k polynomu x[i] pro i = 0,1...p

29) Vypište polynom x[i] pro i = 0,1...p

30) Je-li j = 0, jděte na 37)

31) j = j-1

32) Vytvořte polynom u[i] z polynomu x[i] tak, že u[i] = x[i] pro i = 0,1..p-j

33) Vytvořte polynom ~u

34) Vynásobte polynom ~u číslem λ[j] a výsledek zapište do polynomu pom

35) Odečtěte polynom pom od polynomu x a výsledek zapište do x

36) Jděte na 30)

37) Vypište polynom x

38) Položte s[i] = (x[i]+y[i]) pro i = 0,1...p. Vynásobte polynom s[i] číslem 0.5 a zapište do s[i]

39) Vypište polynom s

40) Položte suma = suma + sqr(s[i]) pro i = 0,1...p

41) Jestliže (suma - c0[0]/2) < ε a k =EN , ra[i] =s[i] a konec.

42) Jděte na 9)

Algoritmus Společného dělitele :

1) Vypište převzaté polynomy a[i] = a[i] , b[i] = ra[i] pro i = 0,1...n

2) Vytvořte matici M o rozměrech [j,i] tak, že M[1,i] = a [i] a M[2, i] = b[i] pro i=0,1...p

3) Vypište matici M

4) Položte st_p =5, rad=0, pak pro j =1,2 najděte stupeň polynomu M[j,i] a jeho hodnotu zapište do h. Je-li h < st_p, pak st_p = h a rad = j

5) Je-li rad= 0, pak matice M je nulová, Konec

6) Je-li rad =1, jděte na 9)

7) Vyměňte řádky v matici M pro i = 0,1...n tak, že pom = M[2,i] a M[2,i] = M[1,i] a M[1,i] = pom

8) Vypište matici M po výměně

9) Položte počet = 0, pak pro j = 1,2 stupeň polynomu M[j,i] zapište do h, je_li h=>0 počet = počet+1

10) Je-li počet = 0, konec

11) Je-li počet = 1 , druhý řádek je nulový a jděte na 17)

12) Určete číslo λ jako podíl koeficientů u nejvyšších mocnin q a číslo f jako rozdíl stupňů polynomů ve druhém a prvním řádku matice M. Vypište λ, f

13) Vynásobte první řádek vztahem $\lambda * q^f$ a výsledek zapište do polynomu ZM

14) Odečtěte polynom ZM od druhého řádku matice M, výsledek zapište do M[2,i] pro i = 0,1...n

15) Vypište matici M po redukci

16) Jděte na 4)

17) Položte dělitel = M[1,i], konec.

V celém algoritmu jsou použity modifikované procedury, popsané již v kapitole 3.4. Ostatní důležité procedury jsou rozepsány v algoritmu.

5 NEJVĚTŠÍ SPOLEČNÝ DĚLITEL MATICOVÝCH POLYNOMŮ

5.1 Úvod

Postup, který chceme použít, umožňuje ke dvěma zadaným maticovým polynomům nalézt jejich největšího společného dělitele. Protože se jedná o operace s maticemi, může být samozřejmě tento dělitel pravý nebo levý.

Budoucí algoritmus výpočtu vychází ze vztahů pro zadané maticové polynomy A,B

5.1“ $$P_1*A+Q_1*B=D_1$$

$$R_1*A+S_1*B=0 \ ,$$

kde A,B jsou známy, D_1 je hledaný největší společný pravý dělitel a P_1,Q_1 a R_1,S_1 jsou matice splňující vztah 5.1“.

Případně může algoritmus vycházet ze vztahů

5.2“ $$A*P_2+B*Q_2=D_2$$

$$A*R_2+B*S_2=0 \ ,$$

kde matice D_2 je největším společným levým dělitelem polynomiálních matic A,B a matice P_2,Q_2 a R_2,S_2 jsou opět matice splňující vztah 5.2“.

5.2 Společný pravý dělitel 2 maticových polynomů

5.2.1 Postup při výpočtu

Pro získání matice D_1 je lepší si soustavu rovnic 5.1“ napsat jako maticovou rovnici, když si předtím vytvoříme matice P a M. Obě jsou podobné maticím z algoritmu na výpočet společného dělitele skalár. polynomů.

Takže pak vypadají takto

5.2.1“

$$P = \begin{vmatrix} P_1 & Q_1 \\ R_1 & S_1 \end{vmatrix} \qquad M = \begin{vmatrix} A \\ B \end{vmatrix}$$

Potom můžeme použít zobecněný postup pro výpočet společného dělitele dvou polynomů, který jsme uvedli při faktorizaci polynomu. Matici M se budeme snažit převést na horní (zobecněnou) trojúhelníkovou matici pomocí řádkových redukcí, podobně jako při výpočtu determinantu. Takto upravená matice M je pak největším společným pravým dělitelem D_1 a počet jejich řádků je roven n=rank M. Po skončení úprav bude soustava rovnic 5.1" odpovídat následujícímu maticovému zápisu

5.2.2"

$$\begin{vmatrix} P_1 & Q_1 \\ R_1 & S_1 \end{vmatrix} * \begin{vmatrix} A \\ B \end{vmatrix} = \begin{vmatrix} D_1 \\ 0 \end{vmatrix}$$

V matici P jsou po výpočtu matice P_1,Q_1,R_1,S_1 a jejich rozměry jsou dány rozměry matice D_1 a matic A,B.

5.2.2 Základní algoritmus výpočtu

Před samotným výpisem algoritmu si musíme definovat podmínku týkající se rozměrů matic : $A_{l,m}$, $B_{p,m}$ -> $M_{l+p,m}$

Zápis algoritmu:

(1) Položte $P=I_{l+p}$

(2) k=0

(3) k=k+1

(4) Je-li k=m+1 nebo k=l+p, konec.

(5) Určete polohu nenulového polynomu a_{ik} nejmenšího stupně ve sloupci k a řádcích k,k+1,...,l+p matice M. Jsou-li všechny polynomy nulové, jděte na (3)

(6) Je-li i≠k, zaměňte řádky i a k matic M a P

(7) Je-li pouze jeden nenulový polynom ve sloupci k a řádcích k,k+1,..,l+p matice M, jděte na (3)

(8) Určete čísla $\lambda_{k+1},...,\lambda_{l+p}$ jako podíly koeficientů u nejvyšších mocnin q a čísla $f_{k+1},...,f_{l+p}$ jako rozdíly stupňů polynomů ve sloupci k a řádcích k+1,...,l+p a polynomu ve sloupci k a řádku k matice M

(9) Odečtěte řádek k násobený $\lambda_{k+1}*q^{f(k+1)},...,\lambda_{l+p}*q^{f(l+p)}$ od řádků k+1,...,l+p matice M i matice P

(10) Jděte na (5)

Po skončení našeho algoritmu jsou matice P_1,Q_1,R_1,S_1 na svých místech v polynomiální matici P a matice D_1 je tvořena nenulovými řádky konečné matice M.

5.3 Společný levý dělitel 2 maticových polynomů

5.3.1 Úvod a postup

Pro získání matice D_2 ze soustavy rovnic 5.2" přepíšeme tuto do maticové rovnice. Musíme nejdříve vytvořit polynomiální matice

5.3.1"

$$N = \left| \begin{array}{cc} A & B \end{array} \right| \qquad Q = \left| \begin{array}{cc} P_2 & R_2 \\ Q_2 & S_2 \end{array} \right|$$

Pro získání společného levého dělitele D_2 použijeme podobný postup jako při výpočtu matice D_1. Jen místo řádkových použijeme jednostranné sloupcové redukce. Těmi převedeme matici N na dolní zobecněnou trojúhelníkovou matici. Celou rovnici (sestavenou z polynomiálních matic), která odpovídá soustavě 5.2" po provedení úprav, můžeme zapsat takto

5.3.2"

$$\left| \begin{array}{cc} A & B \end{array} \right| * \left| \begin{array}{cc} P_2 & R_2 \\ Q_2 & S_2 \end{array} \right| = \left| \begin{array}{cc} D_2 & 0 \end{array} \right|$$

Takto upravená matice N je potom největším společným dělitelem D_2 a počet jeho sloupců je pak n=rank N. Rozměry polynomiálních matic P_2, R_2, Q_2, S_2 obsažených po skončení výpočtu v matici Q jsou dány rozměry matic A, B a rozměrem výsledného společného levého dělitele D_2.

5.3.2 Základní algoritmus výpočtu

Opět si musíme před samotným výpisem algoritmu definovat rozměry zadaných matic : $A_{l,m}$, $B_{l,r}$ -> $N_{l,m+r}$

Zápis algoritmu:

(1) Položte $Q=I_{m+r}$

(2) k=0

(3) k=k+1

(4) Je-li k=l+1 nebo k=m+r, konec.

(5) Určete polohu nenulového polynomu a_{ik} nejmenšího stupně v řádku k a sloupcích k,k+1,...,m+r matice N. Jsou-li všechny polynomy nulové, jděte na (3)

(6) Je-li j≠k, zaměňte sloupce j a k matice N i matice Q

(7) Je-li pouze jeden nenulový polynom v řádku k a sloupcích k,k+1,...,m+r matice N, jděte na (3)

(8) Určete čísla $\mu_{k+1},\ldots, \mu_{m+r}$ jako podíly koeficientů u nejvyšších mocnin q a čísla $g_{k+1},\ldots,g_{m+r}$ jako rozdíly stupňů polynomů v řádku k a sloupcích k+1,...,m+r a polynomu v řádku k a sloupci k matice N

(9) Odečtěte sloupec k násobený $\mu_{k+1}*q^{g(k+1)},\ldots,\mu_{m+r}*q^{g(m+r)}$ od sloupců k+1,...,m+r matice N i matice Q

(10) Jděte na (5)

Po ukončení algoritmu jsou matice P_2,R_2 a Q_2,S_2 na svých místech v polynomiální matici Q a matice D_2 je tvořena nenulovými sloupci matice N.

5.4 Úplný algoritmus pro výpočet společného dělitele

V algoritmu jsou svými jmény volány určité důležité procedury, které v něm ale nejsou popsány. Jsou popsány v předchozích algoritmem. Řádky, v nichž volám tyto procedury jsou označeny *.

0) maxstup = 11

1) Budou na vstupu **M**aticové polynomy nebo **P**olynomiální matice ?

2) M : Chcete vypočítat **Pr**avého společného dělitele maticových polynomů nebo **Le**vého společného dělitele maticových polynomů ?

P : Chcete vypočítat **Pr**avého společného dělitele polynomiálních matic nebo **Le**vého společného dělitele polynomiálních matic ?

PR1) M : Zadejte rozměry matic v maticových polynomech A, B (2* řádky, 2* sloupce : max = 5) -> ra, sa, rb, sb

P : Zadejte rozměry polynomiálních matic A, B (2* řádky, 2* sloupce : max= 5) -> ra, sa, rb, sb

„Pozn. : A a B musí mít stejný počet sloupců. „

PR2) M : Zadejte maximální stupeň maticového polynomu A (max=5) -> stpa

P : Zadejte maximální stupeň polynomu v polynomní matici A (max=5) -> -> stpa

PR3) M : Zadejte maximální stupeň maticového polynomu B (max=5) -> stpb

P : Zadejte maximální stupeň polynomu v polynomní matici B (max=5) -> ->stpb

PR4) rm = ra + rb, stpol se rovná většímu z stpa, stpb

PR5)* M : Načtěte prvky matic maticového polynomu A

* P : Načtěte prvky polynomiální matice A

PR6)* M : Načtěte prvky matic maticového polynomu B

* P : Načtěte prvky polynomiální matice B

PR7) Matice P = $E_{rm,rm}$

PR8) Vypište maticový polynom A převedený na polynomiální matici, nebo vypište polynomiální matici A

PR9) Vypište maticový polynom B převedený na polynomiální matici, nebo vypište polynomiální matici B

PR10) Vypište polynomiální matici P

PR11) Vytvořte polynomiální matici M tak, že radm = rm, slm = sa a předpokládaný stupeň je stpol

PR12) Dosaďte do M takto,

pro i=1..ra, j=1..slm, k=0..stpol M[i,j,k] = A[i,j,k]

pro i=1..rb, j=1..slm, k=0..stpol M[i+ra,j,k] = B[i,j,k]

PR13) Vypište matici M

PR14) krok = 1

PR15) Je-li krok = slm+1 nebo krok = radm, jděte na PR28)

PR16)* Vyhledejte nenulový polynom nejnižšího stupně ve sloupci krok a v řádcích i = krok, krok+1...radm a číslo řádku v němž leží uložte do i1

PR17) Je-li i1 = 0, sloupec krok je nulový a krok = krok+1, jděte na PR15)

PR18)* Jestliže i1 ≠ krok, zaměňte řádek i1 a krok v matici M a zaměňte řádky i1 a krok v matici P

PR19) Vypište matici M po výměně

PR20) Vypište matici P po výměně

PR21)* Zjistěte počet nenulových polynomů ve sloupci krok a řádcích krok, krok+1...radm matice M a zapište ho do i1

PR22) Jestliže i1 = 1, ukončete cyklus redukce ve sloupci krok, krok = krok+1 a jděte na PR15)

PR23)* Redukujte polynomní matici M

PR24)* Redukujte stejnými redukčními koeficienty i polynomní matici P

PR25) Vypište matici M po redukci

PR26) Vypište matici P po redukci

PR27) Jděte na PR16)

PR28) Vypište konečnou polynomiální matici M

PR29) Vypište konečnou polynomiální matici P

PR30) Určete počet nenulových řádků v polynomní matici M a zapište ho do i2, i3=radm - i2

PR31) Pro i = 1..i2, j = 1..slm, k= 0..stpol polynomní matice D1[i,j,k] = M[i,j,k]

PR32) Pro i = 1..i2, j = 1..ra, k = 0..stpol polynomní matice P1[i,j,k] = P[i,j,k]

PR33) Pro i = 1..i3, j = 1..ra, k = 0..stpol polynomní matice R1[i,j,k] = P[i+i2,j,k]

PR34) Pro i = 1..i2, j = 1..rb, k = 0..stpol polynomní matice Q1[i,j,k] = P[i,j+ra,k]

PR35) Pro i = 1..i3, j = 1..rb, k = 0..stpol polynomní matice S1[i,j,k] = P[i+i2,j+ra,k]

PR36) Vypište polynomiální matice D1, P1, Q1, R1, S1

PR37) Konec, jděte na 1)

LV1) M : Zadejte rozměry matic v maticových polynomech A, B (2* řádky, 2* sloupce : max = 5) -> ra, sa, rb, sb

P : Zadejte rozměry polynomiálních matic A, B (2* řádky, 2* sloupce : max= 5) -> ra, sa, rb, sb

„Pozn. : A a B musí mít stejný počet řádků. „

LV2) M : Zadejte maximální stupeň maticového polynomu A (max=5) -> stpa

P : Zadejte maximální stupeň polynomu v polynomní matici A (max=5) -> -> stpa

LV3) M : Zadejte maximální stupeň maticového polynomu B (max=5) -> stpb

P : Zadejte maximální stupeň polynomu v polynomní matici B (max=5) -> ->stpb

LV4) rm = sa + sb, stpol se rovná většímu z stpa, stpb

LV5)* M : Načtěte prvky matic maticového polynomu A

* P : Načtěte prvky polynomiální matice A

LV6)* M : Načtěte prvky matic maticového polynomu B

* P : Načtěte prvky polynomiální matice B

LV7) Matice Q = $E_{rm,rm}$

LV8) Vypište maticový polynom A převedený na polynomiální matici, nebo vypište polynomiální matici A

LV9) Vypište maticový polynom B převedený na polynomiální matici, nebo vypište polynomiální matici B

LV10) Vypište polynomiální matici Q

LV11) Vytvořte polynomiální matici N tak, že radn = ra, sln = rm a předpokládaný stupeň je stpol

LV12) Dosaďte do N takto,

pro i=1..radn, j=1..sa, k=0..stpol N[i,j,k] = A[i,j,k]

pro i=1..radn, j=1..sb, k=0..stpol N[i,j+sa,k] = B[i,j,k]

LV13) Vypište matici N

LV14) krok = 1

LV15) Je-li krok = radn+1 nebo krok = sln, jděte na LV28)

LV16)* Vyhledejte nenulový polynom nejnižšího stupně v řádku krok a v sloupcích j = krok, krok+1...sln matice N a číslo sloupce v němž leží uložte do j1

LV17) Je-li j1 = 0, řádek krok je nulový a krok = krok+1, jděte na LV15)

LV18)* Jestliže j1 ≠ krok, zaměňte sloupce j1 a krok v matici N a zaměňte sloupce j1 a krok v matici Q

LV19) Vypište matici N po výměně

LV20) Vypište matici Q po výměně

LV21)* Zjistěte počet nenulových polynomů ve řádku krok a sloupcích krok, krok+1...sln matice N a zapište ho do j1

LV22) Jestliže j1 = 1, ukončete cyklus redukce v řádku krok, krok = krok+1 a jděte na LV15)

LV23)* Redukujte v řádcích polynomní matici N

LV24)* Redukujte stejnými redukčními koeficienty i polynomní matici Q

LV25) Vypište matici N po redukci

LV26) Vypište matici Q po redukci

LV27) Jděte na LV16)

LV28) Vypište konečnou polynomiální matici N

LV29) Vypište konečnou polynomiální matici Q

LV30) Určete počet nenulových sloupců v polynomní matici N a zapište ho do j2, pro j = 1..sln

LV31) Pro i = 1..radn, j = 1..j2, k= 0..stpol polynomní matice D2[i,j,k] = N[i,j,k]

LV32) Vypište polynomiální matici D2

LV33) Konec, jděte na 1)

Algoritmus lze ukončit zadáním x při zadávání vstupních hodnot, nebo vždy odpovězením „a“ na otázku : „Chcete skončit?“.

6 SYNTÉZA ŘÍZENÍ MNOHOROZMĚRNÉHO ARMAX MODELU

6.1 Úvod

Jak víme, mnohorozměrný ARMAX model je popsán rovnicí

6.1" $\mathbf{A}*y= q^{j}*\mathbf{B}*u+\mathbf{C}$,

kde maticové polynomy jsou opět označny tučně a matice vypisuji do svislých závorek. Na začátku programu na výpočet rovnic pro řízení ARMAX modelu budeme tedy zadávat maticové polynomy **A**, **B**, **C** a j-tou mocninu q. Dále také musíme zadat počet iteračních kroků N a požadovanou přesnost ε, které jsou potřebné pro faktorizaci determinantu maticového polynomu **B**. Jak vidíme, budou zde použity algoritmy z předchozích kapitol.

6.2 Postup při řešení

Po zadání maticových polynomů **A**, **B**, **C** i ostatních údajů převedeme maticový polynom **B** na polynomiální matici. Potom vypočítáme její determinant. Z determinantu můžeme totiž odvodit stabilitu maticového polynomu nebo jeho inverze. Jestliže je stabilní determinant, což je skalární polynom, je stabilní i maticový polynom **B** i $\mathbf{B}^{-1}$. Stabilitu determinantu zjistíme pomocí algoritmu na výpočet faktorizace skalárního polynomu. Pokud je v detB nestabilní prvek, algoritmus pak rozloží detB=b na součin stabilní a nestabilní části, tzn. $b=b_+*b_-$. Stabilní část si pro další použití označíme b_1 ($b_1=b_+$) a nestabilní si označme b_2 ($b_2=b_-$).

Do rovnic 1.1" dále potřebujeme maticové polynomy označené $\mathbf{B}_{1,0}$ a $\mathbf{B}_{1,1}$. Ty získáme následujícím postupem. Vyjdeme z předpokladu, že jestliže je v detB=b nestabilní část, je nestabilní i $\mathbf{B}^{-1}$. Pak je ale stabilní vztah $b_2*\mathbf{B}^{-1}$. Pro tento vztah platí rovnice

6.2" $\mathbf{B}^{-1}*b_2=b_2*\mathbf{B}^{-1}$.

Vzhledem k podobě vztahu 6.2" se vztahem polynomiálních matic **A**, **B** při výpočtu jejich společného dělitele (musí pro ně totiž platit podmínka $\mathbf{B}*\mathbf{A}^{-1}=\mathbf{A}^{-1}*\mathbf{B}$, tzn. jsou nesoudělné) můžeme určit matice $\mathbf{P}_1$, $\mathbf{Q}_1$, $\mathbf{R}_1$, $\mathbf{S}_1$ pro matice **B** a $[b_2*E]$.

Použijeme tedy algoritmus na výpočet společného největšího pravého dělitele z matic **B** a $[b_2*E]$. Na konci tohoto algoritmu získáme matice $\mathbf{R}_1$, $\mathbf{S}_1$ a jetliže upravím druhou rovnici ze vztahu 5.1" na tvar

6.3" $$-\mathbf{S}_1^{-1}*\mathbf{R}_1=\mathbf{B}^{-1}*[b_2*E]$$

a zároveň vím, že maticový polynom B jsme rozložili na tvar

6.4" $$\mathbf{B}=B_2*\mathbf{B}_{11}^{-1}*\mathbf{B}_1 \qquad (B_2=[b_2*E]),$$

mohu potom napsat $-\mathbf{R}_1=\mathbf{B}_{11}$ a $\mathbf{S}_1=\mathbf{B}_1$. Tím jsme získali maticové polynomy $\mathbf{B}_1$, $\mathbf{B}_{11}$.

Dále musíme pro soustavu rovnic 1.1" vypočítat maticové polynomy **T**, **V**. Pro jejich získání budeme řešit lineární diofantickou rovnici

6.5" $$\mathbf{C}*\sim b_2=\mathbf{T}*q^j*b_2+\mathbf{A}*\mathbf{V}.$$

V této diofantické rovnici už známe ze zadání maticové polynomy **A**, **B**, **C** i j-tou mocninu q. Máme již také vypočten polynom b_2, z něhož snadno získáme $\sim b_2$. Pro maticové polynomy **T**, **V** platí podmínky :

6.6" $$\partial T=\partial A-1$$

$$\partial V=\partial(q^j*b_2)-1.$$

Po stanovení těchto podmínek můžeme skalární polynom (b_2*q^j) vynásobit jednotkovou maticí a tím získáme maticový polynom (skalární) B_2. Potom použijeme algoritmus pro výpočet společného levého dělitele maticových polynomů, který aplikujeme na polynomní matice **A** a B_2. Po provedeném výpočtu získáme mimo společného dělitele $\mathbf{D}_2$ i polynomní matice $\mathbf{P}_2$ a $\mathbf{Q}_2$, pro které platí vztah

6.7" $$\mathbf{A}*\mathbf{P}_2+B_2*\mathbf{Q}_2=\mathbf{D}_2 \ .$$

Z této rovnice vyplývá, že musíme vypočítat inverzní matici k polynomní matici $\mathbf{D}_2$ a tou pak vynásobit celou rovnici zprava. Potom si můžeme definovat polynomní matice **TP** a **VP**, pro které platí vztahy

6.8" $$\mathbf{TP}=\mathbf{Q}_2*\mathbf{D}_2^{-1}$$

$$\mathbf{VP}=\mathbf{P}_2*\mathbf{D}_2^{-1} \ .$$

Zápis algoritmu:

1)* Z maticového polynomu **B** udělej polynomní matici **BD**

2)* Vypočítejte determinant polynomní matice **BD** -> detb

3)* Proveďte faktorizaci polynomu detb -> polynomy b1, b2

4) Proveďte polynomní matice B2 =E * b2

5)* Vypočtěte největšího společného pravého dělitele z polynomních matic **BD** a B2 -> polynomní matice **B**11 = -**R**1, **B**1 = **S**1

6) Vypočtěte reciproký polynom ~b2 z b2 -> ob2

7)* Vytvořte maticový polynom **G** = **C** * ob2

8) b2 = b2 * q^j (= posuv koeficientů polynomu b2 o j-míst do prava) => B_2=E*b2

9)* Vypočtěte největšího společného levého dělitele z polynomních matic **A** a B_2 -> polynomní matice $\mathbf{D}_2$, $\mathbf{P}_2$, $\mathbf{Q}_2$

10)* Vypočtěte inverzní matici k matici $\mathbf{D}_2$

11)* Proveďte součiny $\mathbf{P}_2*\mathbf{D}_2^{-1}=\mathbf{VP}$ a $\mathbf{Q}_2*\mathbf{D}_2^{-1}=\mathbf{TP}$

12)* Proveďte součiny **TP***G=$\mathbf{T}_2$ a **VP***G=$\mathbf{V}_2$

13)* Proveďte dělení polynomních matic pro matici **A** a $\mathbf{V}_2$ -> **V**

13)* Proveďte dělení polynomních matic pro matici $\mathbf{B}_2$ a $\mathbf{T}_2$ -> **T**

14) Konec.

6.4 Základní algoritmus pro inverzní matici

Tento algoritmus vychází z algoritmu na výpočet determinantu matice. V podstatě se k tomuto algoritmu přidají řádky, které zajistí výpočet adjukované matice. Její prvky potom pouze vydělíme determinantem a máme inverzní matici k matici zadané.

Zápis algoritmu: zadaná je matice $\mathbf{A}_{m,m}$

(1) Položte matici **P**=I_m

(2) k=0, s=0

(3) k=k+1. Jestliže k=m, jdi na (10)

(4) Určete polohu nenulového polynomu a_{ik} nejmenšího stupně ve sloupci k a řádcích k,k+1,...,m matice **A**

(5) Je-li i≠k, zaměňte řádky i a k matice **A** i **P** a položte s=s+1

(6) Je-li pouze jeden nenulový polynom ve sloupci k a řádcích k,k+1,...,m matice **A**, jděte na(3)

(7) Určete čísla $\lambda_{k+1},...,\lambda_m$ jako podíly koeficientů u nejvyšších mocnin q a čísla $d_{k+1},...,d_m$ jako rozdíly stupňů polynomů ve sloupci k a řádcích k+1,...,m a polynomu ve sloupci k a řádku k matice **A**

(8) Odečtěte řádek k násobený $\lambda_{k+1}*q^{d(k+1)},...,\lambda_m*q^{d(m)}$ od řádků k+1,...,m matice **A** i matice **P**

(9) Jděte na (4)

(10) Vypočtěte součin $a=(-1)^{s}*a_{11}*a_{22}...*a_{mm}$

(11) Je-li k=1, konec.

(12) Vypočtěte pomocí algoritmu na společného dělitele 2 skal. polynomů polynomy r_i, s_i splňující vztah $a_{kk}*r_i+a_{ik}*s_i=0$ pro i=1,2...,k-1

(13) Přičtěte řádek k násobený polynomem r_i k řádku i násobenému polynomem s_i v matici **A** i v matici **P** pro i=1,2,...,k-1

(14) k=k-1

(15) Je-li k≠1, jděte na (12)

(16) Vypočtěte polynomy u_i a v_i splňující $a*u_i+a_{ii}*v_i=0$ pro i=1,2,...,m

(17) Násobte řádek i matice **A** i matice **P** podílem $-(v_i/u_i)$ pro i=1,2,...,m. Konec.

Jak vidíme, v algoritmu se do řádku (11) jedná o modifikovaný algoritmus na výpočet determinantu a další řádky pak umožňují konečný výpočet adjukované matice. Na konci je tedy a rovno determinantu matice **A** (a=detA) a adjA je na místě matice **P**. Protože tento algoritmus použijeme při výpočtu inverzní matice z matice číselné, můžeme prvky matice **P** bez obtíží vydělit determinantem a. Výsledná matice **P** je pak inverzní k zadané matici **A**.

6.5 Postup řešení při dělení maticových polynomů

Nechť **A**(q) je např. stupně 5 a **B**(q) je stupně 2 a řešme dělení (levé) **A** =**P** * **B** + **R**, kde **P** je stupně 3, **R** je stupně 1. Jestliže jsou např. **A**, **B** čtvercové matice 3x3, můžeme sestavit matice

6.5.1“

$$\left|\begin{array}{ccc} -1 & & \\ & -1 & \\ & & -1 \\ \hline & 0 & \end{array}\right| = \mathbf{Q} \qquad \left|\begin{array}{ccc} (2) & \dots & (2) \\ (2) & \dots & (2) \\ (2) & \dots & (2) \\ \hline (5) & \dots & (5) \\ (5) & \dots & (5) \\ (5) & \dots & (5) \end{array}\right| = \mathbf{S}$$

Matici **S** jsme utvořili z matice **B** a **A** čísla v závorce symbolizují maximální stupeň příslušného polynomu. Nechť tedy např. prvek (1,1) v matici **S** je polynom 2. stupně. Není-li , vyměníme ho s jiným řádkem horní poloviny, který 2.stupně je. Potom máme v pozici (1,1) matice **S** polynom 2. stupně a můžeme snížit stupeň prvků (2,1) a (3,1) o 1 v případě, že i jejich stupně = 2. Dostaneme tak v prvním sloupci horní poloviny matice **S** prvky pod diagonálou se stupněm menším než 2. Podobně eliminujeme i druhý sloupec horní poloviny matice **S**, ale začínáme od řádku 2. Všechny prováděné operace v matici **S** uskutečňujeme samozřejmě i v matici **Q**. Tím získáme v horní polovině všechny prvky pod diagonálou se stupněm menším než 2.

6.5.2“

$$\left|\begin{array}{c} \mathbf{G}_2 \\ \hline 0 \end{array}\right| = \mathbf{Q} \qquad \left|\begin{array}{ccc} (2) & (2) & (2) \\ (1) & (2) & (2) \\ (1) & (1) & (2) \\ \hline (5) & (5) & (5) \\ (5) & (5) & (5) \\ (5) & (5) & (5) \end{array}\right| = \mathbf{S}$$

Dále pomocí přičítání vhodných násobků 1. řádku horní poloviny snížíme stupeň v 1. sloupci dolní poloviny matice **S** a stejnými násobky vynásobíme i první řádek matice **Q** a přičteme ho ke odpovídajícím prvkům 1. sloupce dolní poloviny matice **Q**.

6.5.3“

$\mathbf{G}_2$		(2) (2) (2) (1) (2) (2) (1) (1) (2)	
$\mathbf{H}_1$	$= \mathbf{Q}$	(4) (5) (5) (4) (5) (5) (4) (5) (5)	$= \mathbf{S}$

Dále přičítáním 2. řádku snížíme o 1 stupeň druhý sloupec dolní poloviny matice **S**, přičemž díky provedené úpravě (viz. 6.5.2) se stupně v 1. sloupci nezvýší. Pomocí třetího řádku potom upravíme 3. sloupec. Všechny operace uskutečňované v matici **S** provádíme samozřejmě i v matici **Q**. Tak dostaneme

6.5.4“

$\mathbf{G}_2$		(2) (2) (2) (1) (2) (2) (1) (1) (2)	
$\mathbf{H}_2$	$= \mathbf{Q}$	(4) (4) (4) (4) (4) (4) (4) (4) (4)	$= \mathbf{S}$

Tím jsme snížili stupeň dolní poloviny **S** o 1. Proceduru snižování opakujeme až do konečného tvaru

6.5.5“

G_2	(2)	(2)	(2)	$= B_1$
	(1)	(2)	(2)	
	(1)	(1)	(2)	
P	(1)	(1)	(1)	$= R$
	(1)	(1)	(1)	
	(1)	(1)	(1)	

Naznačené úpravy spočívaly v tom, že se ke spodní polovině přičítaly kombinace řádků z horní poloviny neboli od spodní matice se odečetl nějaký **W**-násobek matice horní a tedy **R** = **A** - **W** * **P**, **P** = 0 - **W** * (-E) = **W** neboli **A** = **P** * **B** + **R**. Jelikož **R** je stupně 1 , našli jsme řešení.

My při volání tohoto výpočtu budeme zadávat polynomní matice $\mathbf{T}_2$ a $\mathbf{B}_2$ (**A** a $\mathbf{V}_2$) a výstup pro nás bude polynomní matice **R** (=**T** nebo **V**).

Samotný algoritmus je velmi podobný algoritmu na společného pravého dělitele maticových polynomů, jak je vidět z postupu výpočtu. Změny se v podstatě týkají pouze krokování při úpravách jednotlivých řádků, tj. nejdříve redukuji horní polovinu (k = 1..3 v našem příkladě), a pak redukuji dolní polovinu (k = 4..6 v našem příkladě). Nenuluji ovšem až na nulu, ale hlídám jestli je redukovaný prvek vyššího nebo stejného stupně jako redukční prvek. Jestliže ano, redukuji. Jestliže ne, neredukuji. Tyto úpavy jsou tedy minimální, a proto zde algoritmus nevypísuji . Uživatel si bude moci tyto nepatrné změny prohlédnout a porovnat ve zdrojovém programu.

6.6 Úplný algoritmus na syntézu řízení ARMAX modelu

Všechny procedury v tomto algoritmu volané jsou popsány v předchozích algoritmech (zde jsou některé nepatrně modifikovány) nebo to jsou přímo přejmenované a mírně upravené algoritmy. Důležité procedury budu při jejich volání označovat *.

Zápis algoritmu :

1) Zadejte rozměr maticových polynomů **A, B, C** (pro všechny stejný) -> rm

2) Zadejte maximální stupeň maticového polynomu **A** -> stupola

3) Zadejte maximální stupeň maticového polynomu **B** -> stupolb

4) Zadejte maximální stupeň maticového polynomu **C** -> stupolc

5) Zadejte mocninu j -> je

6)* Načtěte prvky matic maticového polynomu **A**

7)* Načtěte prvky matic maticového polynomu **B**

8)* Načtěte prvky matic maticového polynomu **C**

9) Vypište maticový polynom **A**

10) Vypište maticový polynom **B**

11) Vypište maticový polynom **C**

12)* Transformujte maticový polynom **B** na polynomiální matici **BD**

13) Vypište polynomiální matici **BD**

14)* Vypočtěte determinant polynomní matice **BD** (pro rm a stupolb) : procedura vrací polynom -> detb

15) Vypište polynom detb

16)* Proveďte faktorizaci polynomu detb (pro stupeň detb) : procedura vrací 2 polynomy a jejich stupně -> b1, b2, db1, db2

17) Vypište polynom b1

18) Vypište polynom b2

19)* Proveďte součin E * b2 = B2 (pro rm a db2)

21)* Vypočtěte největšího společného pravého dělitele polynomních matic **BD**, **B2** (pro rm, stupolb, db2) : procedura vrací 2 polynomní matice -> **B11**, **B1**

22) Vypište polynomní matici **B11**

23) Vypište polynomní matic **B1**

24)* Vytvořte reciproký polynom ~b2 z polynomu b2 (pro db2) -> ob2

25) Vypište polynom ob2

26) Proveďte součin b2 * q^{je} a výsledek zapište do b2 (pro db2)

27) Vypište nový polynom b2 i nové db2

28) Pro i = 1..rm, j = 1..rm, k = 0..stupolc, l = 0..db2 proveďte maticový polynom **G**[k+l, i, j] = **G**[k+l, i, j] + **C**[k, i, j] * ob2[l]

29) Vypište maticový polynom **G** (pro rm a stupolg)

30) Stupeň maticového polynomu **T** = dt = stupola -1

31) Stupeň maticového polynomu **V** = dv = db2 - 1

32) Vypište dt, dv

33)* Vypočtěte největšího společného levého dělitele z polynomních matic **A** a **B2** polynomní matice **D2**, **P2**, **Q2**

34)* Vypočtěte inverzní matici k matici **D2**

35) Vypište matici $\mathbf{D2}^{-1}$

36)* Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte součin **P2** * $\mathbf{D2}^{-1}$ = **VP**

37)* Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte součin **Q2** * $\mathbf{D2}^{-1}$ = **TP**

38) Vypište matici **TP**

39) Vypište matici **VP**

40)* Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte součin **V2** = **VP** * **G**

41)* Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte součin **T2** = **TP** * **G**

42) Vypište matici **T2**

43) Vypište matici **V2**

44)* Proveďte dělení polynomních matic pro matice **A** a **V2** -> **R**

45) Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte porovnání **V** = **R**

46)* Proveďte dělení polynomních matic pro matice **B2** a **T2** -> **R**

47) Pro i = 1..rm, j = 1..rm, k = 1..rm proveďte porovnání **T** = **R**

48) Vypište matice **B1**, **B11**, **T**, **V**

49) Konec.

D. ZÁVĚR

V této kapitole bych chtěl shrnout poznatky získané při vytváření algoritmů i při jejich přepisování do programů na výpočet syntézy řízení mnohorozměrného ARMAX modelu. Chtěl bych se věnovat nejen výčtu metod, které jsem zvolil pro vytváření algoritmů, ale i problémům při jejich přepisu na algoritmy a programy.

Je několik metod nebo způsobů, jak počítat determinant čtvercové matice. Zvolil jsem algoritmus jednostranných řádkových redukcí matice (publikovaný v literatuře [6]), protože vnáší do výpočtu determinantu pevný řád, který umožňuje úspěšně řešit kteroukoliv zadanou polynomní matici. Při tvorbě programu jsem zvolil možnost výběru zadávání buď maticového polynomu nebo polynomiální matice. Při syntéze řízení sice budeme zadávat pouze maticové polynomy, ale úlohu jsem zpracovával tak, aby se dala využívat i samostatně. Proto má dvojí způsob zadávání. Mezi největší problémy při tvorbě tohoto programu patřilo určení správného způsobu indexování jednotlivých prvků polynomů při jejich zadávání i při úpravách a výpisu. Dále pak také určení správné přesnosti výpočtu tak, aby ovlivnila správnost výsledků opravdu minimálně. Zvětšováním a zmenšováním měřítka přesnosti spolu s využitím více příkladů jsem nakonec tuto hranici stanovil. Takže při všech příkladech, které jsem použil, je výsledek hodně přesný v porovnání s výsledky počítanými ručně.

Pro výpočet faktorizace, která se použila v syntéze řízení pro určení nestabilní části determinantu maticového polynomu B, jsem využil upravený test stability. V literatuře se totiž často faktorizace nazývá součinovým rozkladem na stabilní a nestabilní část. Tato definice je však nepřesná, protože a^- nemusí být vždy nutně nestabilní (např. $a^- = 1$, pokud $a^+ = a$). A jelikož zatím neexistuje přímá metoda faktorizace, zvolil jsem právě upravený test stability. Když tento test objeví neznámý nestabilní faktor, vypočtu stabilní faktor pomocí společného dělitele zadaného polynomu a jeho reflexe. Mezi největší problémy při sestavování programů z těchto tří výše jmenovaných metod patřila správná úprava Newtonovy metody iteračních kroků v reflexi polynomu. Počítá se v ní totiž se zápornými mocninami. Což pro nás znamenalo indexování zápornými čísly, a to

je v Turbopascalu nemožné. Nakonec jsem mohl, po nalezení a využití podobnosti mezi sdruženým polynomem a reciprokým polynomem, vytvářet program pro reflexi tak, aby výsledek byl vždy stabilní. K řešení problémů s přesností a indexováním jsem zde řešil podobným způsobem, jako v předešlém programu. Takže při aplikaci několika příkladů byly výsledky všech 3 dílčích podprogramů i celkové faktorizace velmi přesné v předem stanovené míře přesnosti.

Při tvorbě programu na společného dělitele maticových polynomů jsem vycházel z již úspěšně vyřešené první části (determinant). Rozdíl mezi nimi je v podstatě ten, že se na začátku programu na společného dělitele zadají dva maticové polynomy nebo dvě polynomiální matice(publikováno v literatuře [6]). Ty se potom přetransformují do jedné polynomiální matice, která se potom zpracovává stejným způsobem jako při výpočtu determinantu. Spolu s ní se však zpracovává polynomiální matice P nebo Q, na začátku načítané jako jednotkové. Z výsledné matice (původně vytvořené ze 2 polynomních matic) pak určíme společného dělitele. Při tvorbě tohoto programu byl největším problémem dvojí způsob indexování při výpočtu buď pravého nebo levého společného dělitele. Při operacích z maticemi se totiž mohou vyskytnout dva největší společní dělitelé, a to pravý a levý. Určit správný způsob indexování a rozměrů při redukcích (při zvolení pravého nebo levého dělitele) bylo asi nejdůležitějším problémem. Dalším problémem bylo ve výsledné matici P (Q) správně určit rozměry matic v ní obsažených. Jejich rozměry a umístění totiž plynou z rozměrů zadaných maticových polynomů, výsledného společného dělitele a samozřejmě z výběru Levý - Pravý dělitel maticových polynomů. Problémy s přesností výpočtu jsem řešil stejným způsobem jako v první části. Pro několik použitých příkladů pak byly výsledky během výpočtu i na konci programu v mezích přesnosti vzhledem ke kontrole s výsledky ručních výpočtů.

Pro samotný program na syntézu řízení jsem využil všechny 3 předchozí programy. Ty byly předtím zpracovány jako samostatné celky a pro program syntézy řízení byly přepsány a upraveny na závislé podprogramy, řízené hlavním programem. Při vytváření programu jsem se setkal s problémem komunikace mezi podprogramy vytvořenými s dříve hlavních programů. Byla nutná kont-

rola typizace proměnných, jejich správné volání a dokonce i jejich přepis. Dále byla nutná i kontrola správného volání procedur nebo testování, zda-li nejsou obsaženy shodné procedury a typizace proměnných ve dvou nebo více podprogramech. Po odstranění chyb a správné kontrole výše uvedených problémů probíhala komunikace a předávání proměnných bez problémů. Menším problémem byla správná transformace algoritmu na výpočet pravého společného dělitele na algoritmus pro výpočet dělení polynomiálních matic. Jelikož jsem před touto transformací nejdříve dlouho zkoušel vytvořit program na řešení diofantické rovnice podle jiné metody, nezbýval mi na velmi kvalitní odladění tohoto algoritmu čas. Podle ručně vypočteného příkladu (kontrola správnosti) tento program probíhá v pořádku. Ale na kontrolu jinými příklady už mi bohužel nezbyl čas. Podle mě je třeba tomuto programu (rozuměj program na dělení matic) dát ještě nějaký čas pro dobré vyladění . Lituji že jsem to nestihl já ve své práci.

Ošetření přesnosti výpočtu je stejné jako při předchozích operacích. Při dosazení několika příkladů, předem počítaných ručně pro kontrolu, byly výsledky v mezích předem definované přesnosti.

Jinak diplomová práce splnila očekávání, která byla do ní vložena. Úkolem bylo vytvoření algoritmu z původní teorie syntézy řízení a ten byl splněn. Program vypočte ze zadaných maticových polynomů popisujících vícerozměrný ARMAX model maticové polynomy, určující rovnice mnohorozměrného regulátoru. Diplomová práce neměla za úkol vytvoření simulace mnohorozměrných soustav. Řešení tohoto problému však může vyjít z této diplomové práce. Při tvorbě programů na simulaci lze využít všechny postupy a programy z této diplomové práce.

LITERATURA

[1] ČERNÝ I. : Úvod do teorie funkcí komplexní proměnné. SNP, Praha 1960.

[2] HALOUSKOVÁ A. : Syntéza mnohaparametrových lineárních diskrétních regulačních obvodů podle kvadratických kritérií. Zpráva č.174, ÚTIA ČSAV, Praha 1966.

[3] HANUŠ B., KRACÍK V., TŮMA L. : Algebraická metoda nvrhování číslicových řídících algoritmů. ASŘ Sešity INORGA, Inorga Praha 1988.

[4] HANUŠ B., KRACÍK V. : Algebrická syntéza řízení v mnohorozměrném ARMAX modelu I. Výzkumná zpráva KTK-0166, VŠST Liberec 1987.

[5] KRACÍK V., TŮMA L. : Algebraická syntéza řízení diskrétního systému. Výzkumná zpráva KTK-0113, VŠST Liberec 1983.

[6] KUČERA V. : Algebraická teorie diskrétního lineárního řízení. Academia Praha 1978.

[7] STREJC V. : Teorie lineární regulace. Učební text FEL ČVUT, Praha 1972.

[8] VOSTRÝ Z. : Syntéza lineárního diskrétního řízení podle kvadratického kritéria pro soustavy s nestejným počtem vstupů a výstupů. Kand. disert. práce, ÚTIA ČSAV, Praha 1973.

PROGRAMY NA VÝPOČET DETERMINANTU

```
Program determinant_maticoveho_polynomu;

uses crt, polynom, polmat;

(*
  zjednodusujici predpoklady vypoctu
   - maximalni rozmer matice je 5
   - maximalni zadavany stupen polynomu je 5, deklarace pro roznasobeni je 11
   - pri zadavani vstupu jako maticovy polynom je max. pocet matic 6 (stupen
polynomu = 5)
  poznamky
   - radky oznacene {*} jsou pouze pro sledovani prubeznych vysledku vypoctu
     a je treba je odstranit
   - lepsi je si pozadovanou cast odkrokovat od nastaveneho bodu preruseni a
     sledovani promennych v okne watch
   - procedura Nacti_zadanou_matici zavede hodnoty z prikladu
   - po odzkouseni je treba ji odstranit a jeji volani v hlavnim programu
     nahradit volanim procedury Nacti_pol_mat - viz #1 v programu
   - v unitech neni zatim treba nic menit
*)

Var
   vstup_d, koncit, videt  : char;
   vstup_n : shortint;
   je_mat_pol  : boolean;
   rozmer_matice, stupen_polynomu, pocet_vymen     : shortint;
   x          : word;
   pol1, pol2  : t_polynom;
   polmat1     : pol_mat;
   k, i, i1, a : shortint;

Procedure Nacti_zadanou_matici;
begin
nuluj_pol_mat(polmat1);
end;

{ ************************************************************************** }
begin {program}
   repeat
      clrscr;
      nuluj_polynom(pol1);
      nuluj_polynom(pol2);
      nuluj_pol_mat(polmat1);
      writeln( '****** Determinant maticoveho polynomu ****** ');
```

```
    writeln( 'Ukoncit program pri zadavani vstupu muzete zadanim znaku "x"
nebo "X" ');
    writeln( 'do znakoveho vstupu nebo cislem 99 do ciselneho vstupu. ');
    writeln;
    pocet_vymen := 0;
    koncit := 'A';
    videt := 'N';
{ zadani typu vstupnich udaju }
    repeat
      write ( 'Bude vstup M_aticovy polynom ' +
        'nebo P_olynomialni matice < M,P > ? ');
      readln(vstup_d);
    until (vstup_d = 'm')
      OR (vstup_d = 'p')
      OR (vstup_d = 'M')
      OR (vstup_d = 'P')
      OR (vstup_d = 'x')
      OR (vstup_d = 'X');
    if (vstup_d = 'x') OR (vstup_d = 'X') then exit ;
      if (vstup_d = 'm') OR (vstup_d = 'M') then
        je_mat_pol := True
      else
        je_mat_pol := False;

{ zadani rozmeru ctvercove matice }
    repeat
      if je_mat_pol then
      begin
        writeln( 'Zadejte rozmer ctvercovych matic v maticovem polynomu (pro
vsechny stejny - ');
        write( 'min. 2; max. 5 ) ');
      end
      else
        write( 'Zadejte rozmer ctvercove polynomialni matice ( min. 2; max. 5 )
');
      readln(vstup_d);
    until ((vstup_d < '6') AND (vstup_d > '1'))
      OR (vstup_d = 'x') OR (vstup_d = 'X') ;
    if (vstup_d = 'x') OR (vstup_d = 'X') then exit ;
    val(vstup_d, rozmer_matice, x);

{ zadani maximalniho stupne polynomu }
    repeat
      if je_mat_pol then
        write( 'Zadejte maximalni stupen maticoveho polynomu ( max. 5 ) ')
      else
```

```
        write( 'Zadejte maximalni stupen polynomu polynomialni matice ( max.
5 ) ');
        readln(vstup_d);
    until ((vstup_d < '6') AND (vstup_d > '0'))
      OR (vstup_d = 'x') OR (vstup_d = 'X');
    if (vstup_d = 'x') OR (vstup_d = 'X') then exit ;
    val(vstup_d, stupen_polynomu, x);

    if je_mat_pol then
      nacti_mat_pol(polmat1, rozmer_matice, stupen_polynomu)
    else
{ nacti_zadanou_matici;}
  nacti_pol_mat(polmat1, rozmer_matice, stupen_polynomu);

    clrscr;
    writeln( 'polynomni matice po zadani ');
    writeln;
    vypis_matici(polmat1, rozmer_matice, stupen_polynomu);
    writeln;

{ zobrazovat postup vypoctu ? }
    repeat
      write ( 'Chcete videt prubeh vypoctu <aAnN> ? ');
      readln(videt);
    until (videt = 'a')
      OR (videt = 'A')
      OR (videt = 'n')
      OR (videt = 'N');
    clrscr;

{ cyklus nulovani sloupcu pod diagonalou }
{ cyklus nulovani sloupcu k, posledni sloupec neni v cyklu zahrnuty }
    k := 1;
    while k < rozmer_matice do
    begin
      if (videt = 'a') OR (videt = 'A') then
      begin
        write( 'nuluji sloupec '); writeln(k);
      end; { if }
      i1 := vyhl_pol_ns(polmat1, rozmer_matice, k);
      if i1 = 0 then
      begin
        writeln( 'sloupec je nulovy - det = 0 ');
        exit;
      end; { if }
```

```
if i1 <> k then
      begin
        if (videt = 'a ') OR (videt = 'A ') then
          writeln( 'vymena radku ',k:3, ' za ',i1:3);
        vymen_radky(polmat1, rozmer_matice, max_stupen_polynomu, k,i1);

        if (videt = 'a ') OR (videt = 'A ') then
        begin
          writeln( 'polynomni matice po vymene radku ');
          vypis_matici(polmat1, rozmer_matice, max_stupen_polynomu);
          readln;
        end; { if }
        inc(pocet_vymen);
      end; { if }
      i1 := pocet_nenul_pol(polmat1, rozmer_matice, k);
      if (videt = 'a ') OR (videt = 'A ') then
        writeln( 'pocet nenulovych polynomu = ', i1:3);
      if i1 = 0 then
      begin
        writeln( 'pocet nenul. polynomu = 0 !!! chyba ');
        exit;
      end; { if }
      if  i1 = 1 then
      begin
        if (videt = 'a ') OR (videt = 'A ') then
        begin
          writeln( 'konec cyklu v sloupci ',k:3);
          writeln;
        end ; { if }
        inc(k);
      end { if }
      else
      begin
        if (videt = 'a ') OR (videt = 'A ') then
          writeln( 'redukce slupce ',k:3);
        redukuj_pol_mat(polmat1, rozmer_matice, k, videt);
        if (videt = 'a ') OR (videt = 'A ') then
        begin
          writeln( 'Polynomni matice po redukci ');
          vypis_matici(polmat1, rozmer_matice, max_stupen_polynomu);
          readln;
        end; { if }
      end; {else}
    end; { while }
```

```
if (videt = 'a') OR (videt = 'A') then
    begin
      writeln('Polynomni matice konecna');
      vypis_matici(polmat1, rozmer_matice, max_stupen_polynomu);
      readln;
    end; { if }
    soucin_pol_diag(polmat1, rozmer_matice, pocet_vymen, pol2);


{ vynasobeni soucinu -1^pocet_vymen }
    for a := 0 to max_stupen_polynomu do
      pol2[a] := umocni(-1, pocet_vymen) * pol2[a];
    write('determinant = ');
    vypis_polynom(pol2);
    repeat
      write('Chcete koncit <A,N> ? ');
      readln(koncit);
    until (koncit = 'a')
      OR (koncit = 'A')
      OR (koncit = 'n')
      OR (koncit = 'N');
  until (koncit = 'A')
    OR (koncit = 'a') ;
  clrscr;
end. { program }
```

```
Unit PolMat;  { polynomialni matice }

{*********************************************************************}
Interface
{*********************************************************************}

Uses Crt, Polynom;

Const
   max_rozmer_matice = 5;

Type
   pol_mat = array[1..max_rozmer_matice, 1..max_rozmer_matice] of t_polynom;

Procedure nuluj_pol_mat(var pm1 : pol_mat);
{ vynuluje polynomialni matici pm1 }

Procedure nacti_pol_mat(var pm1 : pol_mat; rm, sp : shortint);
{ nacte polynomialni matici o rozmeru rm a stupni polynomu sp }

Procedure nacti_mat_pol(var pm1 : pol_mat; rm, sp : shortint);
{ nacte maticovy polynom o poctu sp a rozmerech rm matic a prevede jej do
  polynomialni matice pm1 }

Procedure vymen_radky(var pm1 : pol_mat; rm, sp, i , i1 : shortint);
{ v polynomialni matici pm1 o rozmeru rm a stupni polynomu sp vymeni radky
  i a i1 }

Procedure vypis_matici(pm1 : pol_mat; rm, sp : shortint);
{ vypise polynomialni matici o rozmeru rm a stupni polynomu sp }

Function pocet_nenul_pol(pm1 : pol_mat; rm, k : shortint) : shortint;

Function vyhl_pol_ns(pm1 : pol_mat; rm, k : shortint) : shortint;

Procedure redukuj_pol_mat(var pm1 : pol_mat; rm, k : shortint; vi : char);

Procedure soucin_pol_diag(pm1 : pol_mat; rm, pv : shortint; var p1 : t_polynom);
{ vypocte soucin polynomu na diagonale matice pm1 }
```

```
{*************************************************************************}
implementation
{*************************************************************************}

Procedure nuluj_pol_mat;
var k, i, a : integer;
begin
    for k := 1 to max_rozmer_matice do
    begin
        for i := 1 to max_rozmer_matice do
            nuluj_polynom(pm1[i,k]);
    end;
end;

Procedure nacti_pol_mat;
var k, i, a : shortint;
   s1, s2, s3  : string[255];
begin
   nuluj_pol_mat(pm1);
   for i := 1 to rm do
      for k := 1 to rm do
         for a := sp downto 0 do
         begin
            Str(k:1,s1);
            Str(i:1,s2);
            Str(a:1,s3);
            write( 'Zadejte koeficient u clenu  p^ ');
            write(s3 + ' prvku [ ' + s2 + ', ' + s1 +  '] polynomialni matice  ');
            readln(pm1[k,i,a]);
            if pm1[k,i,a] = 99 then
              halt;
         end;
end;


Procedure nacti_mat_pol;

type
   matice_1 = array[1..max_rozmer_matice, 1..max_rozmer_matice] of real;
   polynom_n = array[0..max_stupen_polynomu] of matice_1;
var
   pol1 : polynom_n;
   k, i, a : shortint;
   s1, s2, s3 : string;
```

```
begin
  nuluj_pol_mat(pm1);
  for a := sp downto 0 do
    for i := 1 to rm do
      for k := 1 to rm do
        begin
          Str(k:1,s1);
          Str(i:1,s2);
          Str(a:1,s3);
          write( 'Zadejte v matici u clenu  p^' + s3 + ' prvek [ ' + s2 + ',' + s1 +
'] ');
          readln(pol1[a,k,i]);
          if pol1[a,k,i] = 99 then
            halt;
        end;
{ !!!!!! - procedura ! transformace maticoveho polynomu do polynomialni matice }
  for k := 1 to rm do
    for i := 1 to rm do
      for a := 0 to sp do
        pm1[k,i,a] := pol1[a,k,i];
end;

Procedure vymen_radky;
var a, k : shortint;
  b : real;
begin
  for k := 1 to rm do
    for a := 0 to sp do
    begin
      b := pm1[k,i1,a];
      pm1[k,i1,a] := pm1[k,i,a];
      pm1[k,i,a] := b;
    end;
end;

Procedure vypis_matici;
var k, i, a : shortint;
  s1, s2, s3  : string[255];
begin
  for i := 1 to rm do
    for k := 1 to rm do
    begin
      Str(k:1,s1);
      Str(i:1,s2);
      write( 'A[ ' +  s2 + ',' + s1 + '] - ');
      vypis_polynom(pm1[k,i]);
    end; end;
```

```
Function pocet_nenul_pol;{ (pm1 : pol_mat; rm, k : shortint) : shortint; }
var
  a, i, pocet : shortint;
begin
   pocet := 0;
   for i := k to rm do
   begin
      a := vrat_stupen_polynomu(pm1[k,i]);
      if a >= 0 then                {je nenulovy ?}
        inc(pocet);
   end;
   pocet_nenul_pol := pocet;
end;

Function vyhl_pol_ns; { (pm1 : pol_mat; rm, k : shortint) : shortint; }
var
  riadok, stupen, i, a  : shortint;
begin
  riadok := 0;
  stupen := 11;

  for i := k to rm do
  begin
     a := vrat_stupen_polynomu(pm1[k,i]);
     if a >= 0 then               {je nenulovy nebo byl konec polynomu ?}
        if a < stupen then        {je nizsiho stupne jak predchazejici ?}
           begin
              stupen := a;
              riadok := i;
           end;
  end;
  vyhl_pol_ns := riadok;
end;

Procedure redukuj_pol_mat;{ (pm1 : pol_mat; rm, k : shortint; vi : char); }
var
   i, j            : shortint;
   k1, k2, dx      : shortint;
   kk, ki, lx      : real;
   pol2            : t_polynom;
begin
  for i := k+1 to rm do { cyklus pro radky v sloupci k }
  begin
     k1 := vrat_stupen_polynomu(pm1[k,k]);
     kk := pm1[k,k,k1];
     k2 := vrat_stupen_polynomu(pm1[k,i]);
```

```
    if k2 >= 0 then
    begin
       ki := pm1[k,i,k2];
       if k2-k1 < 0 then
       begin
          writeln( 'Chyba - zaporny exponent d ! ');
          halt;
       end;
       if (vi = 'a ') OR (vi = 'A ') then
       begin
          write( 'koeficient li = ');
          write(ki/kk:3:3);
          write( '; koeficient di = ');
          writeln(k2-k1:3);
       end; { if }
       for j := k to rm do
       begin
          lx := ki / kk;
          dx := k2 - k1;
          nasob_polynom(pm1[j,k], lx, dx, pol2);
          odecti_polynomy(pm1[j,i], pol2, pm1[j,i]);
       end; { for }
    end; { if }
  end; { for }
end;

Procedure soucin_pol_diag; {pm1 : pol_mat; rm, pv : shortint; var p1 : t_polynom}
var
a, e   : shortint;
begin
  secti_polynomy(pm1[1,1], p1, p1);
  for a := 2 to rm do
     soucin_polynomu(pm1[a,a], p1, p1);
end;

end.  { of Unit PolMatice }
```

PROGRAM S POUŽITÝMI PROCEDURAMI A FUNKCEMI

```
Unit Pouz_Fce;

{*******************************}
interface
{*******************************}

uses crt;

Const
    max_exponent = 11;
    max_stupen_polynomu = 5;


Type
   Polynom = array[0..max_exponent] of real;

Var
{   n : integer;}
  cykl_x : integer;

Procedure Otoc_Polynom(vstup : Polynom; var vystup : Polynom; Zadany_max :
integer);

Procedure Nasob_polynom(vstup : Polynom; nasob : real; stupen : integer; n :
integer; var vystup : Polynom);

Procedure Del_polynom(vstup : Polynom; nasob : real; stupen : integer; n :
integer; var vystup : Polynom);

Procedure nuluj_polynom(var vstup : Polynom);

Function Exponent(vstup : Polynom) : integer;

Function Coeficient(vstup : Polynom; exponent : integer) : real;

Procedure Odecti_Polynomy(vstup1, vstup2 : Polynom; n : integer; var vystup :
Polynom);
{Procedure Soucin_Polynomu(vstup1, vstup2 : Polynom; var vystup : Polynom);}

Procedure Kopiruj_Polynom(vstup : Polynom; var vystup : Polynom);

Procedure Secti_Polynomy(vstup1, vstup2 : Polynom; n : integer; var vystup :
Polynom);
```

```
Function MStri(r : real) : string;
{ zdokonalena procedura Str }

Procedure Ukaz_polynom(vstup : Polynom; max : integer);

Procedure Napis_stupen(a : integer);

Procedure Deleni_polynomu(c, b :polynom; dc, db :integer; var vystup :polynom);

Function Exponent_0(vstup : polynom) : integer;

{ ***************************************************************************}
implementation
{ ***************************************************************************}

Procedure Otoc_Polynom;
      var a : integer;
begin
   nuluj_polynom(vystup);
   for a := 0 to zadany_max do
      vystup[a] := vstup[zadany_max - a];
end;

{ ***************************************************************************}
Function Exponent;
   var a : integer;
      vystup : integer;
begin
   vystup := 0;
   for a := 0 to max_exponent do
   begin
      if vstup[max_exponent - a] <> 0.00 then
      begin
         if vystup = 0 then
         begin
            vystup := max_exponent - a;
         end;
      end;
   end;
   Exponent := vystup;
end;
```

```
{ ***************************************************************************}
Function Coeficient;
      var vystup : real;
begin
      Coeficient := vstup[exponent];
end;


{ ***************************************************************************}
Procedure Nasob_Polynom;
       var a : integer;
begin
       nuluj_polynom(vystup);
       for a := 0 to n do
              vystup[stupen + a] := vstup[a] * nasob;
end;

{ ***************************************************************************}
Procedure Del_polynom;
       var a : integer;
begin
       nuluj_polynom(vystup);
       for a := 0 to n do
      begin
              vystup[a - stupen] := vstup[a] / nasob;
      end;
end;

{ ***************************************************************************}
Procedure Nuluj_polynom;
       var a : integer;
begin
       for a := 0 to max_exponent do
              vstup[a] := 0;
end;

{***************************************************************************}
Procedure Odecti_Polynomy;
       var a : integer;
begin
       nuluj_polynom(vystup);
       for a := 0 to n do
          vystup[a] := vstup1[a] - vstup2[a];
end;
```

```
{*************************************************************************}
{Procedure Sucin_polynomu;
        var a, b : integer;
begin
        nuluj_polynom(vystup);
        for a := 0 to n do
           for b := 0 to n do
           begin
              if (a+b > n) and (vstup[a] <> 0) and (vstup2 <> 0) then
              begin
                 writeLn( 'Chyba - doslo k prekroceniu maximalneho stupnu
polynomu. ');
                 halt;
              end;
              if a+b <= n then
                vystup[a+b] := vystup[a+b] + vstup1[a] * vstup2[a];
           end;
end;


{ ************************************************************************}
Procedure Kopiruj_polynom;
        var a : integer;
begin
    Nuluj_Polynom(vystup);
    for a := 0 to Exponent(vstup) do
       vystup[a] := vstup[a];
end;

{*****************************************}
Procedure Secti_Polynomy;
        var a : integer;
begin
        nuluj_polynom(vystup);
        for a := 0 to n do
           vystup[a] := vstup1[a] + vstup2[a];
end;
{*********************************************}
Function MStri;
var
  a    : shortint;
  s    : string;
begin
  str(r:10:16,s);
  a:= length(s);
```

```
      while (a > 0) AND (s[a] = '0') do
      begin
          s[0] := chr(a);
          dec(a);
      end;
   if s[a] = '.' then
     dec(a);
     s[0] := chr(a);
     MStri := s;
end;

{*****************************************}
Procedure Ukaz_Polynom;
          var a, max_p : integer;
begin
    if wherey = 1 then
    begin
      writeLn;
    end;
    max_p := Exponent(vstup);
    write('polynom = ');
    for a := 0 to max_p do
    begin
        if vstup[a] < 0 then
        begin
            write(' - ');
        end
        else
        begin
            write(' + ');
        end;
        write(MStri(ABS(vstup[a])), ' * q');
        Napis_stupen(a);
    end;
    writeln; write('<Enter>');
    readln;
end;
{*********************************************}
Procedure Napis_stupen;
begin
{    gotoxy(wherex, wherey - 1);}
    write('^',a);
{    gotoxy(wherex, wherey + 1); }
end;
```

```
{*******************************************}
Procedure Deleni_polynomu;

var d,a : polynom;
   i, j, k, m, n, JO : integer;
   konec : boolean;
begin
    nuluj_polynom(vystup);
    nuluj_polynom(a);
    konec := false;
    m := dc; n := db;
{    writeln( 'Deleni polynomu ');
    writeln( 'm = ',m:2, '     n = ',n:2);readln;                }
    while konec = false do
    begin
        k := m - n;
        a[k] := c[m]/b[n];
   {     writeln( 'k = ',k:2, '   a ',k:1, ' = ',a[k]:4:3); readln;}
        for i := 0 to n do
        begin
            d[i+k] := b[i]*a[k];
            c[i+k] := c[i+k] - d[i+k];
            if ABS(c[i+k])< 0.0001 then c[i+k] := 0;
            {writeln( ' c ',i+k:1, ' = ',c[i+k]:5:5, '    d ',i+k:1, ' = ',d[i+k]:4:3);
            readln;}
        end;
        if (c[m] = 0) and (m >0) then m := m - 1;
        {writeln( 'm = ',m:1);readln;}
        for j := 0 to m do
        begin
            if (c[j]=0) AND (k=0) then begin konec := true; JO := 1 end;
            if (k=0) AND (c[j] <> 0) then begin konec := true; JO := 0 end;
{            writeln ( 'jo = ',jo:1);}
        end;
    end;
    if JO = 1 then
    begin for i := 0 to dc-db do vystup[i] := a[i]; end;
    if JO = 0 then
    begin for i := 0 to dc-db do vystup[i] := 0; end;
end;

{*******************************************}
Function Exponent_0;
    var cykl : integer;
       vystup : integer;
begin
    vystup := 0;
```

```
  if vstup[0] = 0.00 then
  begin
      for cykl := 1 to max_exponent do
      begin
          { write( ' a := ', cykl);}
          if vstup[cykl] <> 0.00 then
          begin
              {writeLn( 'vystup = ', vystup);}
              if (vystup = 0) then
              begin
                  { write( ' Vkladam = ', cykl);}
                  vystup := cykl;
              end;
          end;
      end;
    end
    else
    begin
        vystup := 0;
    end;
    Exponent_0 := vystup;
end;

end.  {Unit Pouz_Fce}
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Rozmery maticovych polynomu A, B          **********

 Pozn1.: Matice A a B maji myt shodny pocet sloupcu.
 Pozn2.: Program muzete ukoncit zadanim znaku "X"
----------------------------------------------------------------------

ejte sloupce polynomialni matice A i B ( max.= 5 ) 2

ejte radky polynomialni matice A ( max.= 5 ) 2

ejte radky polynomialni matice B ( max.= 5 ) 2


imalni stupne polynomu v A, B  ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Stupne maticovych polynomu A, B           **********

 Pozn.: Program muzete ukoncit zadanim znaku "X"
----------------------------------------------------------------------

ejte maximalni stupen polynomu polynomialni matice A ( max. 5 ) 2

ejte maximalni stupen polynomu polynomialni matice B ( max. 5 ) 1


teni prvku matic A, B  ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

nomni matice A po zadani :

.] - polynom = 1
2] - polynom = 7*q-3
.] - polynom = 0.5
2] - polynom = 1*q^2
ter>

nomni matice B po zadani :

.] - polynom = 1*q-0.5
2] - polynom = 0
.] - polynom = 0
2] - polynom = 1*q-0.5


matice P  ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

nomni matice P po zadani :

.] - polynom = 1
2] - polynom = 0
3] - polynom = 0
4] - polynom = 0
.] - polynom = 0
2] - polynom = 1
3] - polynom = 0
4] - polynom = 0
.] - polynom = 0
2] - polynom = 0
3] - polynom = 1
4] - polynom = 0
ter>
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

ynomni matice M vytvorena z A, B :

1] - polynom = 1
2] - polynom = 7*q-3
1] - polynom = 0.5
2] - polynom = 1*q^2
1] - polynom = 1*q-0.5
2] - polynom = 0
1] - polynom = 0
2] - polynom = 1*q-0.5


ete videt prubeh vypoctu <A/N> ?
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Vysledek = matice D1, F1, Q1, R1, S1          **********

                ** Spolecny pravy delitel matic A, B -> D1 **

ice D1 :
1] - polynom = 1
2] - polynom = 7*q-3
1] - polynom = 0
2] - polynom = 1*q-0.5
ter>
```

```
**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu *********
**********************************************************************

**********          Vysledek = matice D1, P1, Q1, R1, S1          *********

                    ** Pomocne matice splnujici vztahy **
                    **           P1*A+Q1*B=D1             **
                    **           R1*A+S1*B=0              **

ice P1 :
1] - polynom = 1
2] - polynom = 0
1] - polynom = 0
2] - polynom = 0

ice Q1 :
1] - polynom = 0
2] - polynom = 0
1] - polynom = 0
2] - polynom = 1

ter>

**********************************************************************
********** Nejvetsi pravy spolecny delitel maticovych polynomu *********
**********************************************************************

**********          Vysledek = matice D1, P1, Q1, R1, S1          *********

                    ** Pomocne matice splnujici vztahy **
                    **           P1*A+Q1*B=D1             **
                    **           R1*A+S1*B=0              **

ice R1 :
1] - polynom = -1*q+0.5
2] - polynom = 0
1] - polynom = -0.5
2] - polynom = 1

ice S1 :
1] - polynom = 1
2] - polynom = 7*q-3
1] - polynom = 0
2] - polynom = -1*q+3

ete skoncit <A/N> ?
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Rozmery maticovych polynomu A, B          **********

Pozn1.: Program muzete ukoncit zadanim znaku "X".
Pozn2.: Polynomni matice A a B maji myt shodny pocet radku.
```

```
ejte radky polynomialni matice A a B ( max.= 5 ) 2

ejte sloupce polynomialni matice A ( max.= 5 ) 2

ejte sloupce polynomialni matice B ( max.= 5 ) 2


one maticovych polynomu A, B ?   ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Maximalni stupne polynomu v A, B          **********

Pozn.: Program muzete ukoncit zadanim znaku "X".
```

```
ejte maximalni stupen polynomu polynomialni matice A ( max. 5 ) 2

ejte maximalni stupen polynomu polynomialni matice B ( max. 5 ) 1


teni prvku matic A, B ?   ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

*********          Polynomni matice A po zadani          *********

1] : polynom = 2
1] : polynom = -1*q+2
1] : polynom = 4*q
2] : polynom = -2*q^2-2*q-1


r matice B  ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

*********          Polynomni matice B po zadani          *********

1] : polynom = 1*q+1
2] : polynom = 1
1] : polynom = 2*q+1
2] : polynom = -1*q-3


r matice Q  ->  <Enter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********              Polynomni matice G po zadani              **********

 ] : polynom = 1
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 1
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 1
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 0
 ] : polynom = 1

 matice N  ->  <Enter>

**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Polynomni matice N vytvorena z A, B          **********

 ] : polynom = 2
 ] : polynom = -1*q+2
 ] : polynom = 1*q+1
 ] : polynom = 1
 ] : polynom = 4*q
 ] : polynom = -2*q^2-2*q-1
 ] : polynom = 2*q+1
 ] : polynom = -1*q-3

te videt prubeh vypoctu <A/N> ?
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu *********
**********************************************************************

**********          Vysledek = matice D2, P2, Q2, R2, S2          *********

                  ** Spolecny levy delitel matic A, B -> D2 **
.] : polynom = 2
2] : polynom = 0
.] : polynom = 4*q
2] : polynom = -2.5

ter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu *********
**********************************************************************

**********          Vysledek = matice D2, P2, Q2, R2, S2          *********

                  ** Pomocne matice splnujici vztahy **
                  **            A*P2+B*Q2=D2            **
                  **            A*R2+B*S2=0             **

ice P2 :
1] : polynom = 1
2] : polynom = -0.25*q
1] : polynom = 0
2] : polynom = -0.5

ice Q2 :
1] : polynom = 0
2] : polynom = 0
1] : polynom = 0
2] : polynom = 1

ter>
```

```
**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Vysledek = matice D2, P2, Q2, R2, S2          **********

                    ** Pomocne matice splnujici vztahy **
                    **          A*P2+B*Q2=D2             **
                    **          A*R2+B*S2=0              **

ce P2 :
] : polynom = 1
] : polynom = -0.25*q
] : polynom = 0
] : polynom = -0.5

ce Q2 :
] : polynom = 0
] : polynom = 0
] : polynom = 0
] : polynom = 1

er>

**********************************************************************
********** Nejvetsi levy spolecny delitel maticovych polynomu **********
**********************************************************************

**********          Vysledek = matice D2, P2, Q2, R2, S2          **********

                    ** Pomocne matice splnujici vztahy **
                    **          A*P2+B*Q2=D2             **
                    **          A*R2+B*S2=0              **

ce R2 :
] : polynom = -0.2*q^2-0.26666667*q-0.5
] : polynom = 0.6*q^2+0.6*q-1
] : polynom = -0.4*q-0.2
] : polynom = 1.2*q+1.2

ce S2 :
] : polynom = 1
] : polynom = 0
] : polynom = 0.13333333*q+0.4
] : polynom = -2.4*q-0.4

te skoncit <A/N> ?
```

```
      ************************************************************
      ********** Determinant maticoveho polynomu **********
      ************************************************************

      Pozn.: Program muzete ukoncit zadanim znaku "X"
```

```
e vstup M_aticovy polynom nebo P_olynomialni matice < M/P > ? P

ejte rozmer ctvercove polynomialni matice ( min. 2; max. 5 ) 3

ejte maximalni stupen polynomu polynomialni matice ( max. 5 ) 2

ter>
```

```
      ************************************************************
      ********** Determinant maticoveho polynomu **********
      ************************************************************

ynomni matice po zadani :

1] - polynom = 0
2] - polynom = 0
3] - polynom = 1*q^2-1*q+1
1] - polynom = 1
2] - polynom = -1*q+1
3] - polynom = 0
1] - polynom = 1*q^2+1
2] - polynom = -1*q-1
3] - polynom = -1*q-1

ete videt prubeh vypoctu <A/N> ?
```

```
        ********************************************************************
        *********   Prubeh vypoctu determinantu   **********

olynomni matice konecna :
1,1] - polynom = 1
1,2] - polynom = -1*q+1
1,3] - polynom = 0
2,1] - polynom = 0
2,2] - polynom = -2
2,3] - polynom = -1*q-1
3,1] - polynom = 0
3,2] - polynom = 0
3,3] - polynom = 1*q^2-1*q+1

Enter>
```

```
        ********************************************************************
        ********** Determinant maticoveho polynomu **********
        ********************************************************************

eterminant = polynom = -2*q^2+2*q-2

hcete koncit <A/N> ?
```

```
**********************************************************
********** Faktorizace skalarniho polynomu **********
**********************************************************

**********  Zadavani koeficientu polynomu  **********

Pozn.: Program muzete ukoncit zadanim cisla "99"
```

```
Zadejte koeficient polynomu pri stupni 0 : 2
Zadejte koeficient polynomu pri stupni 1 : -4.5
Zadejte koeficient polynomu pri stupni 2 : 1

Zadany polynom?  => <Enter>
```

```
**********************************************************
********** Faktorizace skalarniho polynomu **********
**********************************************************

Zadany polynom = + 2 * q^0 - 4.5 * q^1 + 1 * q^2
<Enter>

Zadejte pocet iteracnich kroku v pripadne rexlexi : 5
```

```
**********************************************************
********** Faktorizace skalarniho polynomu **********
**********************************************************

Zadany polynom ma neznamou nestabilni cast. Musim pocitat jeho reflexi.
<Enter>
```

```
**********************************************************
********** Faktorizace skalarniho polynomu **********
**********************************************************

Reflexe k polynomu zadanemu :
ra= polynom = + 4.00000001 * q^0 - 3 * q^1 + 0.5 * q^2
<Enter>
```

```
**********************************************************
********** Faktorizace skalarniho polynomu **********
**********************************************************

Puvodni polynom a :
 polynom = + 2 * q^0 - 4.5 * q^1 + 1 * q^2
<Enter>
Stabilni cast polynomu :
a+ = polynom = - 4 * q^0 + 1 * q^1
<Enter>
Nestabilni cast polynomu :
a- = polynom = - 0.5 * q^0 + 1 * q^1
<Enter>
Konec faktorizace. -> <Enter>

Chcete koncit <A,N> ?
```