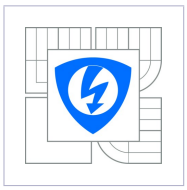

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV MATEMATIKY

Moderní numerické metody

(Cvičení z MATLABu)

Michal Novák

evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Ústav matematiky FEKT VUT v Brně, 2015
http://www.umat.feec.vutbr.cz

Tento text byl vytvořen v rámci realizace projektu CZ.1.07/2.2.00/28.0193,
Komplexní inovace studijních programů a zvyšování kvality výuky na FEKT VUT v Brně.

Tento učební materiál se odkazuje na software MATLAB. Uváděné algoritmy jsou zpracovány tak, aby je bylo možné spustit na co nejširším spektru verzí softwaru – jsou používány pouze nejzákladnější příkazy a nejjednodušší typy algoritmů. Některé z algoritmů využívají symbolické zápisy a výpočty. Pro jejich běh je tedy nutný Symbolic Math Toolbox.

Obsah

Předmluva

Předkládaný studijní materiál je doplňkovým studijním materiálem pro předmět *Moderní numerické metody*. Jeho plánované využití je zejména v počítačových cvičeních a při přípravě samostatných prací, ať už v prezenční nebo kombinované formě předmětu.

Důvod vzniku tohoto studijního materiálu je prostý: výpočty numerické matematiky jsou dnes natolik složité, že bez výpočetní techniky jsou v podstatě nemyslitelné. Co však znamená *použití výpočetní techniky*? Jakým způsobem je dobré / vhodné / správné / korektní výpočetní techniku – v našem případě software MATLAB – používat? Jaká je vlastně při řešení úloh z numerické matematiky role člověka a jaká je role výpočetní techniky?

Smyslem tohoto krátkého studijního materiálu je přimět čtenáře přemýšlet nad *praktickou realizací* teoretických poznatků popisovaných v ostatních studijních materiálech. Tedy ptát se, „jak zařídit, aby“, „jak ověřit, že“ apod. a přemýšlet nad tím, jak probírané metody pracují, jaké jsou jejich možnosti a jaká mají omezení. A dopracovat se poznání, které úkony při řešení probíraných problémů je vhodné přenechat stroji a u kterých je naopak nezastupitelná role člověka.

K tomuto účelu bylo sestaveno 13 algoritmů popisujících metody probírané v předmětu. Anonymně – bez uvedení metody, bez vysvětlujících komentářů, bez záruky správnosti, zpracováno člověkem, jehož přístup k programování nemusí vyhovovat všem (a o jehož schopnostech programovat není nic známo). Tedy přesně tak, jako kdyby tyto algoritmy čtenář nalezl naprosto náhodně na internetových zdrojích, jejichž kvalitu neumí posoudit ani si ji nemůže nechat nezávisle ověřit. Zdrojový kód algoritmů je psán tak, aby pracoval ve všech aktuálně dostupných verzích software MATLAB.

Ke každému z algoritmů se váže sada úkolů a cvičení. Prvním z nich je pochopitelně vždy identifikace dané numerické metody a ověření, že algoritmus dává správné výsledky. Posledním z úkolů by vždy mělo být sepsání vlastního algoritmu – a na čtenáři už ponechávám, jak detailně se danou numerickou metodu rozhodne zpracovat. Zde totiž neexistuje jediná předepsaná správná odpověď a detailnost i kvalita zpracování by měly být vždy adekvátní účelu, jemuž mají sloužit.

březen 2015 *autor*

1 Soustavy lineárních rovnic

U každého z následujících algoritmů odpovězte na následující otázky, resp. proveďte zadané úkoly. *Vždy zkontrolujte, že algoritmus dává správné výsledky a opravte případné chyby!*

1. Uvedené algoritmy popisují hledání řešení soustavy lineárních rovnic dvěma z následujících čtyř metod: *Jacobiho metody bez relaxačního parametru*, *Jacobiho metody s relaxačním parametrem*, *Gauss-Seidelovy metody bez relaxačního parametru*, *Gauss-Seidelovy metody s relaxačním parametrem*. Identifikujte je a stávající algoritmy upravte tak, abyste získali pro každou z uvedených metod jeden algoritmus.

2. Rozmyslete si, jak jinak lze vyřešit zadávání úlohy, tj. matice, vektoru pravých stran a počáteční aproximace. Algoritmus v tomto smyslu upravte.

3. Jaká kritéra konvergence daný algoritmus používá? Jaká jiná kritéria lze použít? Modifikujte algoritmus tak, abyste použili jiné kritérium konvergence. Je ověřování kritérií konvergence dostatečné? Udejte příklad soustavy rovnic, kterou algoritmus vyhodnotí jako nevhodnou, avšak člověk ji jednoduchým zásahem upraví do požadovaného tvaru. Tento „jednoduchý zásah“ naprogramujte.

4. U každého algoritmu určete, jakou ukončovací podmínku používá. Pokud to jde, přepište algoritmus tak, abyste použili jinou ukončovací podmínku.

5. Dohledejte příkaz MATLABu, který používá stejný algoritmus. Dopište použití tohoto příkazu do textu algoritmu a nechejte MATLAB vypsat a srovnat oba výsledky.

6. Modifikujte výpis výsledných hodnot a jiných textových prvků algoritmu. Upravte formát výpisu čísel.

1.1 Algoritmus č. 1

```
clear all
format short;
rad_matice = input('Zadej řád matice: ');
A=zeros(rad_matice,rad_matice + 1);
```

```
fprintf('Při zadávání matice zadávejte i pravou stranu rovnice;
koeficient na pravé straně ponechejte beze změn.
Např. rovnici 10x+5y+z=6 zadejte jako [10 5 1 6]')
for n=1:rad_matice
    hlaska = ['Zadej řádek č.' int2str(n) ' včetně pravé strany: '];
    A(n,:) = input(hlaska);
end
konverguje = true;
for m=1:rad_matice
   mezisoucet = 0;
   for n=1:rad_matice
      if (m ~= n)
          mezisoucet = mezisoucet + abs(A(m,n));
      end
   end
   if (abs(A(m,m)) < mezisoucet)
      konverguje = false;
   end
end
if (konverguje == false)
   fprintf('Matice není v iteračním tvaru. Končím.')
else
    x_pocatecni = input('Zadej počáteční aproximaci ve formátu [1 2 3]: ');
    relaxacni_parametr = input('Zadej relaxační parametr: ');
    x_stare = transpose(x_pocatecni);
    pocet_kroku = input('Zadej počet kroků: ');
    for iterace = 1:pocet_kroku
        for r=1:rad_matice
            cyklus = 0;
            for s = 1:rad_matice
               if (r ~= s)
                   cyklus = cyklus + A(r,s)*x_stare(s,1);
               end
            end
            x_nove(r,1) = (1-relaxacni_parametr)* x_stare(r,1) +
            relaxacni_parametr * ((A(r,rad_matice+1) - cyklus) / A(r,r));
        end
        x_stare = x_nove;
        dalsi_aproximace = transpose(x_nove)
    end
end
```

1.2 Algoritmus č. 2

```
clear all
format short;
rad_matice = input('Zadej řád matice: ');
A=zeros(rad_matice,rad_matice + 1);
fprintf('Při zadávání matice zadávejte i pravou stranu rovnice;
koeficient na pravé straně ponechejte beze změn.
Např. rovnici 10x+5y+z=6 zadejte jako [10 5 1 6]')
for n=1:rad_matice
    hlaska = ['Zadej řádek č.' int2str(n) ' včetně pravé strany: '];
    A(n,:) = input(hlaska);
end
konverguje = true;
for m=1:rad_matice
   mezisoucet = 0;
   for n=1:rad_matice
      if (m ~= n)
          mezisoucet = mezisoucet + abs(A(m,n));
      end
   end
   if (abs(A(m,m)) <= mezisoucet)
      konverguje = false;
   end
end
    x_pocatecni = input('Zadej počáteční aproximaci ve formátu [1 2 3]: ');
    x_stare = transpose(x_pocatecni);
    pocet_kroku = input('Zadej počet kroků: ');
    for iterace = 1:pocet_kroku
        for r=1:rad_matice
            cyklus = 0;
            for s = 1:rad_matice
               if (r ~= s)
                   cyklus = cyklus + A(r,s)*x_stare(s,1);
               end
            end
            x_nove(r,1)=(A(r,rad_matice+1) - cyklus)/A(r,r);
            x_stare(r,1) = x_nove(r,1);
        end
        dalsi_aproximace = transpose(x_nove)
    end
```

2 Řešení rovnic

U každého z následujících algoritmů odpovězte na následující otázky, resp. proveďte zadané úkoly. *Vždy zkontrolujte, že algoritmus dává správné výsledky a opravte případné chyby!*

1. Identifikujte metodu, kterou daný algoritmus používá.
2. Algoritmy využívají pro práci příkaz na dosazování. Je to nutné? Zkuste navrhnout alternativní způsoby.
3. Jaká kritéra konvergence daný algoritmus používá? Jaká jiná kritéria lze použít? Modifikujte algoritmus tak, abyste použili jiné kritérium konvergence.
4. U každého algoritmu určete, jakou ukončovací podmínku používá. Pokud to jde, přepište algoritmus tak, abyste použili jinou ukončovací podmínku.
5. Lze daný algoritmus použít jen na hledání reálných nebo i na hledání komplexních řešení? Jak docílíte toho, abyste našli komplexní řešení? Změní se v tomto případě nějak ověřování podmínek konvergence?
6. Dohledejte příkaz MATLABu, který používá stejný algoritmus. Dopište použití tohoto příkazu do textu algoritmu a nechejte MATLAB vypsat a srovnat oba výsledky.
7. Modifikujte výpis výsledných hodnot a jiných textových prvků algoritmu. Upravte formát výpisu čísel.

Další otázky a úkoly naleznete u příslušných algoritmů.

2.1 Algoritmus č. 1

```
clear all
format short;
x=sym('x');
funkce = input('Zadej levou stranu rovnice v promenne x: ');
presnost = input('Zadej požadovanou přesnost: ');
z0 = input('Zadej počáteční aproximaci: ');
```

```
aproximace(1)=z0;
derivace=diff(funkce)
c = subs(derivace,z0);
for n=1:100
    z1=z0-subs(funkce,z0)/c;
    if abs(z1-z0)<presnost
        aproximace(n+1)=z1;
        break
    end
    aproximace(n+1)=z1;
    z0=z1;
end
aproximace
```

2.2 Algoritmus č. 2

Otázky a úkoly specifické pro tento algoritmus:

1. Zadejte interval $\langle a; b\rangle$ na jednom řádku. Upravte algoritmus, aby pracoval i s tímto vstupem.
2. Proč se vypisuje text `Nevyhovujici zadani`? Co algoritmus v danou chvíli ošetřuje?

```
clear all
format short;
x=sym('x');
f = input('Zadej levou stranu rovnice v promenne x: ');
a = input('Zadej dolní mez intervalu, v nemz lezi koren: ');
b = input('Zadej horní mez intervalu, v nemz lezi koren: ');
eps = input('Zadej pozadovanou presnost: ');
k=1;
aproximace_a(k)=a+b;
aproximace_b(k)=a+b;
derivace=diff(f);
if abs(b-a)<eps;
    fprintf('Nevyhovující zadání')
else
    if subs(f,a)*subs(f,b)>=0
        fprintf('Zvol jiný interval')
    else
        druhader = subs(diff(f,2),(a+b)/2);
        if subs(f,a)*druhader>0
            a_n=a;
```

```
            b_n=b;
        end
        if subs(f,b)*druhader>0
            a_n=b;
            b_n=a;
        end
        while (abs(a_n-aproximace_a(k))>eps | abs(b_n-aproximace_b(k))>eps)
            a_n1 = a_n-subs(f,a_n)/subs(diff(f),a_n);
            b_n1 = (b_n*subs(f,a_n1)-a_n1*subs(f,b_n))/(subs(f,a_n1)-subs(f,b_n));
            k=k+1;
            aproximace_a(k)=a_n;
            aproximace_b(k)=b_n;
            a_n=a_n1;
            b_n=b_n1;
        end
        for i=1:length(aproximace_a)-1
            vysledek_a(i)=aproximace_a(i+1);
            vysledek_b(i)=aproximace_b(i+1);
        end
        vysledek_a
        vysledek_b
    end
end
```

2.3 Algoritmus č. 3

Otázky a úkoly specifické pro tento algoritmus:

1. Popište účel části vyznačené v textu.
2. Pro jaké stupně polynomů funguje dělení pomocí příkazu `deconv`?

```
clear all
format short;
polynomy(1,:)=input('Polynomu ve tvaru [x1 x2 ... xn]: ');
derivace=polyder(polynomy(1,:));
polynomy(2,1)=0;
for i=1:size(polynomy,2)-1
    polynomy(2,i+1)=derivace(i);
end
delenec=[1 2 3]; % jenom pomocne
j=1;
while length(delenec)>2
```

```
    k=1;
    clear delenec
    while polynomy(j+1,k)==0
        k=k+1;
    end
    % popište účel této části
    if (j>1 & abs(zbytek(k))<10^(-7))
        k=k+1;
    end
    for n=1:k-1
        polynomy(j+1,n)=0;
    end
    % konec
    for m=1:length(polynomy(j+1,:))-k+1
        delenec(m)=polynomy(j+1,k+m-1);
    end
    delenec;
    [nasobek,zbytek]=deconv(polynomy(j,:),delenec);
    polynomy(j+2,:)=-zbytek;
    j=j+1;
end
polynomy
```

3 Vlastní čísla

U každého z následujících algoritmů odpovězte na následující otázky, resp. proveďte zadané úkoly. *Vždy zkontrolujte, že algoritmus dává správné výsledky a opravte případné chyby!*

1. Identifikujte metodu, kterou daný algoritmus používá.
2. Uvedené algoritmy pracují s maticemi. Navrhněte alternativní způsob jejich zadávání a – je-li to nutné – upravte daný algoritmus tak, aby pracoval s tímto novým vstupem.
3. U každého algoritmu určete, jakou ukončovací podmínku používá. Pokud to jde, přepište algoritmus tak, abyste použili jinou ukončovací podmínku.
4. Dohledejte příkaz MATLABu, který používá stejný algoritmus. Dopište použití tohoto příkazu do textu algoritmu a nechejte MATLAB vypsat a srovnat oba výsledky.
5. Modifikujte výpis výsledných hodnot a jiných textových prvků algoritmu. Upravte formát výpisu čísel.

Další otázky a úkoly naleznete u příslušných algoritmů.

3.1 Algoritmus č. 1

```
clear all
format short;
rad_matice = input('Zadej řád matice: ');
A=zeros(rad_matice,rad_matice);
for n=1:rad_matice
    hlaska = ['Zadej řádek č.' int2str(n) ' matice: '];
    A(n,:) = input(hlaska);
end
En=eye(rad_matice);
p(1)=1;
p(2)=trace(A)
```

```
B_stare=A;
for m=3:rad_matice+1
    B_nove=A*(B_stare-p(m-1)*En)
    p(m)=trace(B_nove)/(m-1);
    B_stare=B_nove;
end
p(1)=-1;
polynom=-1*p
```

3.2 Algoritmus č. 2

Otázky a úkoly specifické pro tento algoritmus:

1. Proč se vypisuje text `Zadana matice neni symetricka`? Čemu tato skutečnost vadí? Navrhněte, jak postupovat v případě, že matice symetrická není.

```
clear all
format short;
rad_matice = input('Zadej řád matice: ');
A=zeros(rad_matice,rad_matice);
for n=1:rad_matice
    hlaska = ['Zadej řádek č.' int2str(n) ' matice: '];
    A(n,:) = input(hlaska);
end
if isequal(A,transpose(A))==1
   pocatecni_aproximace=input('Zadej počáteční aproximaci
   ve tvaru [1 2 3 ... n]: ');
    iterace_x=transpose(pocatecni_aproximace);
    pocet_iteraci=input('Zadej počet iterací: ');
    for m=1:pocet_iteraci
        x_interim=A*iterace_x
        lambda=(transpose(iterace_x)*x_interim)/(transpose(iterace_x)*iterace_x)
        iterace_x=x_interim;
        posloupnost(m)=lambda;
    end
    posloupnost
else
    fprintf('Zadaná matice není symetrická.')
end
```

3.3 Algoritmus č. 3

Otázky a úkoly specifické pro tento algoritmus:

1. Kdy může tento algoritmus havarovat? Jak tomu lze zabránit?

```
clear all
format short;
rad_matice = input('Zadej řád matice: ');
A=zeros(rad_matice,rad_matice);
for n=1:rad_matice
    hlaska = ['Zadej řádek č.' int2str(n) ' matice: '];
    A(n,:) = input(hlaska);
end
pocatecni_aproximace=input('Zadej počáteční aproximaci
ve tvaru [1 2 3 ... n]: ');
iterace_x=transpose(pocatecni_aproximace);
pocet_iteraci=input('Zadej počet iterací: ');
for m=1:pocet_iteraci
    x_interim=A*iterace_x
    lambda=max(abs(x_interim))
    x_nove=1/lambda*x_interim
    iterace_x=x_nove
    posloupnost(m)=lambda;
end
posloupnost
```

4 Diferenciální rovnice a jejich soustavy

U každého z následujících algoritmů odpovězte na následující otázky, resp. proveďte zadané úkoly. *Vždy zkontrolujte, že algoritmus dává správné výsledky a opravte případné chyby!*

1. Identifikujte metodu, kterou daný algoritmus používá.
2. Uvedené algoritmy pracují s počátečními, resp. okrajovými podmínkami. Navrhněte alternativní způsoby jejich zadávání. Každý algoritmus modifikujte tak, aby korektně pracoval s tímto novým vstupem, resp. způsobem zadání. Proveďte totéž u zadávání intervalu.
3. Algoritmy využívají pro práci příkaz na dosazování. Je to nutné? Zkuste navrhnout alternativní způsoby. Ověřte na různých typech rovnic.
4. Pracují algoritmy korektně pro všechny přípustné typy rovnic? Srovnejte chování algoritmů pro rovnice obsahující / neobsahující na pravé straně rovnice $y' = f(x, y)$ jako sčítanec funkci $g(x)$.
5. Dohledejte příkaz MATLABu, který používá stejný algoritmus. Dopište použití tohoto příkazu do textu algoritmu a nechejte MATLAB vypsat a srovnat oba výsledky.
6. Navrhněte způsoby vizualizace řešení získaných danými algoritmy.
7. Modifikujte výpis výsledných hodnot a jiných textových prvků algoritmu. Upravte formát výpisu čísel.

Další otázky a úkoly naleznete u příslušných algoritmů.

4.1 Algoritmus č. 1

```
clear all
x=sym('x'); y=sym('y');
derivace=input('Zadejte pravou stranu rovnice y''=');
```

```
dolni_mez=input('Zadejte pocatecni bod intervalu: ');
horni_mez=input('Zadejte koncovy bod intervalu: ');
h=input('Zadejte krok: ');
pocatecni_podminka=input('Zadejte pocatecni podminku (y(x0)=): ');
pocet_opakovani=(horni_mez-dolni_mez)/h+1;
xi(1)=dolni_mez;
yi(1)=pocatecni_podminka;
for i=2:pocet_opakovani
    x=xi(i-1);
    y=yi(i-1);
    yi(i)=yi(i-1)+h*subs(derivace);
    xi(i)=xi(i-1)+h;
end
vysledky(1,:)=xi;
vysledky(2,:)=yi;
for i=2:pocet_opakovani
    x=xi(i-1);
    y=yi(i-1);
    k(1)=subs(derivace);
    x=xi(i-1)+1/2*h;
    y=yi(i-1)+1/2*h*k(1);
    k(2)=subs(derivace);
    yi(i)=yi(i-1)+h*k(2);
    xi(i)=xi(i-1)+h;
end
vysledky(3,:)=yi;
for i=2:pocet_opakovani
    x=xi(i-1);
    y=yi(i-1);
    k(1)=subs(derivace);
    x=xi(i-1)+h;
    y=yi(i-1)+h*k(1);
    k(2)=subs(derivace);
    yi(i)=yi(i-1)+1/2*h*(k(1)+k(2));
    xi(i)=xi(i-1)+h;
end
vysledky(4,:)=yi;
for i=2:pocet_opakovani
    x=xi(i-1);
    y=yi(i-1);
    k(1)=subs(derivace);
    x=xi(i-1)+1/2*h;
    y=yi(i-1)+1/2*h*k(1);
    k(2)=subs(derivace);
```

```
    x=xi(i-1)+1/2*h;
    y=yi(i-1)+1/2*h*k(2);
    k(3)=subs(derivace);
    x=xi(i-1)+h;
    y=yi(i-1)+h*k(3);
    k(4)=subs(derivace);
    yi(i)=yi(i-1)+1/6*h*(k(1)+2*k(2)+2*k(3)+k(4));
    xi(i)=xi(i-1)+h;
end
vysledky(5,:)=yi;
clear xi;
xi=vysledky(1,:)
vysledky(2,:)
vysledky(3,:)
vysledky(4,:)
vysledky(5,:)
```

4.2 Algoritmus č. 2

Otázky a úkoly specifické pro tento algoritmus:

1. V algoritmu je použita proměnná `metoda`. O jakou metodu se jedná?

```
clear all
format short
x=sym('x'); y=sym('y');
derivace=input('Zadejte pravou stranu rovnice y''=');
dolni_mez=input('Zadejte dolni mez intervalu: ');
horni_mez=input('Zadejte horni mez intervalu: ');
h=input('Zadejte krok: ');
hodnota=dolni_mez;
pocatecni_podminka=input('Zadejte pocatecni podminku (y(x)=): ');
pocet_opakovani=(horni_mez-dolni_mez)/h+1;
xi(1)=hodnota;
yi(1)=pocatecni_podminka;
for i=2:pocet_opakovani
    x=xi(i-1);
    y=yi(i-1);
    k(1)=subs(derivace);
    x=xi(i-1)+1/2*h;
    y=yi(i-1)+1/2*h*k(1);
    k(2)=subs(derivace);
    x=xi(i-1)+1/2*h;
```

```
    y=yi(i-1)+1/2*h*k(2);
    k(3)=subs(derivace);
    x=xi(i-1)+h;
    y=yi(i-1)+h*k(3);
    k(4)=subs(derivace);
    yi(i)=yi(i-1)+1/6*h*(k(1)+2*k(2)+2*k(3)+k(4));
    xi(i)=xi(i-1)+h;
    k;
end
vysledky=yi;
metoda=vysledky
yi=vysledky;
for n=4:length(xi)
    for j=1:3
        x=xi(n-4+j);
        y=yi(n-4+j);
        f(j)=subs(derivace);
    end
    yi(n)=yi(n-1)+1/12*h*(5*f(1)-16*f(2)+23*f(3));
end
tretiho_radu=yi
yi=vysledky;
for n=5:length(xi)
    for j=1:4
        x=xi(n-5+j);
        y=yi(n-5+j);
        f(j)=subs(derivace);
    end
    yi(n)=yi(n-1)+1/24*h*(-9*f(1)+37*f(2)-59*f(3)+55*f(4));
end
ctvrteho_radu=yi
yi=vysledky;
for n=6:length(xi)
    for j=1:5
        x=xi(n-6+j);
        y=yi(n-6+j);
        f(j)=subs(derivace);
    end
    yi(n)=yi(n-1)+1/720*h*(251*f(1)-1274*f(2)+2616*f(3)-2774*f(4)+1901*f(5));
end
pateho_radu=yi
```

4.3 Algoritmus č. 3

Otázky a úkoly specifické pro tento algoritmus:

1. Lze zadat vstupní údaje algoritmu, resp. pravé strany rovnic, maticově? Modifikujte algoritmus, aby pracoval i s tímto alternativním způsobem zadání.

```
clear all
format short
fprintf('Pri zadavani pouzivejte u,v,w pro funkce promenne x \n')
u=sym('u'); v=sym('v'); w=sym('w'); x=sym('x');
derivace1=input('Zadejte pravou stranu 1. rovnice u''=');
xi(1)=input('Zadejte hodnotu x0: ');
ui(1)=input('Zadejte pocatecni podminku (u(x0)=): ');
derivace2=input('Zadejte pravou stranu 2. rovnice v''=');
vi(1)=input('Zadejte pocatecni podminku (v(x0)=): ');
derivace3=input('Zadejte pravou stranu 3. rovnice w''=');
wi(1)=input('Zadejte pocatecni podminku (w(x0)=): ');
dolni_mez=xi(1);
horni_mez=input('Zadejte koncovy bod intervalu: ');
h=input('Zadejte krok: ');
pocet_opakovani=(horni_mez-dolni_mez)/h+1;
x=xi(1); u=ui(1); v=vi(1); w=wi(1);
for i=2:pocet_opakovani
    ui(i)=ui(i-1)+h*subs(derivace1);
    vi(i)=vi(i-1)+h*subs(derivace2);
    wi(i)=wi(i-1)+h*subs(derivace3);
    xi(i)=xi(i-1)+h;
    u=ui(i);
    v=vi(i);
    w=wi(i);
    x=xi(i);
end
xi
ui
vi
wi
```

4.4 Algoritmus č. 4

Otázky a úkoly specifické pro tento algoritmus:

1. Lze zadat vstupní údaje algoritmu, resp. pravé strany rovnic, maticově? Modifikujte algoritmus, aby pracoval i s tímto alternativním způsobem zadání.
2. Modifikujte algoritmus tak, aby pracoval s libovolným počtem rovnic.

```
clear all
fprintf('Rovnice zadavejte pomoci funkci y,z,u promenne x. \n')
x=sym('x'); y=sym('y'); u=sym('u'); z=sym('z');
derivace1=input('Zadejte pravou stranu rovnice y''=');
xi(1)=input('Zadejte hodnotu x0: ');
yi(1)=input('Zadejte pocatecni podminku (y(x0)=): ');
derivace2=input('Zadejte pravou stranu rovnice z''=');
zi(1)=input('Zadejte pocatecni podminku (z(x0)=): ');
derivace3=input('Zadejte pravou stranu rovnice u''=');
ui(1)=input('Zadejte pocatecni podminku (u(x0)=): ');
dolni_mez=input('Zadejte dolni mez intervalu: ');
horni_mez=input('Zadejte horni mez intervalu: ');
h=input('Zadejte krok: ');
pocet_opakovani=(horni_mez-dolni_mez)/h+1;
x=xi(1); y=yi(1); z=zi(1); u=ui(1);
for i=2:pocet_opakovani
    k(1)=h*subs(derivace1); l(1)=h*subs(derivace2); m(1)=h*subs(derivace3);
    x=xi(i-1)+1/2*h; y=yi(i-1)+1/2*k(1); z=zi(i-1)+1/2*l(1); u=ui(i-1)+1/2*m(1);
    k(2)=h*subs(derivace1); l(2)=h*subs(derivace2); m(2)=h*subs(derivace3);
    x=xi(i-1)+1/2*h; y=yi(i-1)+1/2*k(2); z=zi(i-1)+1/2*l(2); u=ui(i-1)+1/2*m(2);
    k(3)=h*subs(derivace1); l(3)=h*subs(derivace2); m(3)=h*subs(derivace3);
    x=xi(i-1)+h; y=yi(i-1)+k(3); z=zi(i-1)+l(3); u=ui(i-1)+m(3);
    k(4)=h*subs(derivace1); l(4)=h*subs(derivace2); m(4)=h*subs(derivace3);
    delta_y=1/6*(k(1)+2*k(2)+2*k(3)+k(4));
    yi(i)=yi(i-1)+delta_y;
    y=yi(i);
    delta_z=1/6*(l(1)+2*l(2)+2*l(3)+l(4));
    zi(i)=zi(i-1)+delta_z;
    z=zi(i);
    delta_u=1/6*(m(1)+2*m(2)+2*m(3)+m(4));
    ui(i)=ui(i-1)+delta_u;
    u=ui(i);
    xi(i)=xi(i-1)+h;
    x=xi(i)
    k
    delta_y
    y
    l
    delta_z
    z
```

```
    m
    delta_u
    u
end
xi
yi
zi
ui
```

4.5 Algoritmus č. 5

Otázky a úkoly specifické pro tento algoritmus:

1. Připouští algoritmus libovolný krok, resp. libovolný počet částí intervalu? Co musejí části intervalu, se kterými se pracuje, splňovat?
2. Popište, co dělá cyklus `while`. Je nutné tuto část kódu zacyklit? Proč ano, příp. proč ne?
3. Jaký je význam cyklu `if k<3`? Je v kódu nutný? Proč ano, příp. proč ne?
4. Modifikujte algoritmus tak, aby pracoval s libovolným počtem rovnic.

```
clear all
format short
y=sym('y'); x=sym('x');
fprintf('Uvazujme rovnici ve tvaru y''''+f1(x)y''+f2(x)*y = f3(x) \n')
f1=input('Zadejte funkci f1(x): ');
f2=input('Zadejte funkci f2(x): ');
f3=input('Zadejte funkci f3(x): ');
fprintf('Uvazujeme okrajove podminky ve tvaru y(a) = alpha, y(b) = beta \n')
a=input('a = ');
alf=input('y(a) = ');
b=input('b = ');
bet=input('y(b) = ');
n=input('Zadejte pocet casti intervalu: n = ');
eps=input('Zadejte pozadovanou presnost: eps = ');
px=exp(int(f1,x))
qx=-f2*px
fx=-f3*px
norma=2*eps;
k=0;
h=(b-a)/n;
while (norma>eps)
```

```
k=k+1;
clear vypis;
xi(1)=a; xi(2)=a+h; pi(1)=subs(px,x,xi(2)-h/2); pi(2)=subs(px,x,xi(2)+h/2);
qi(1)=subs(qx,x,xi(2)); fi(1)=subs(fx,x,xi(2));
A(1,1)=pi(1)+pi(2)+h^2*qi(1); A(1,2)=-pi(2); B(1,1)=h^2*fi(1)+pi(1)*alf;
if n>3
    for i=2:n-2
        xi(i+1)=xi(i)+h;
        pi(2*i-1)=subs(px,x,xi(i+1)-h/2);
        pi(2*i)=subs(px,x,xi(i+1)+h/2);
        qi(i)=subs(qx,x,xi(i+1));
        fi(i)=subs(fx,x,xi(i+1));
        A(i,i-1)=-pi(2*(i)-1);
        A(i,i)=pi(2*i-1)+pi(2*i)+h^2*qi(i);
        A(i,i+1)=-pi(2*i);
        B(i,1)=h^2*fi(i);
    end
    xi(i+3)=b;
end
xi(n)=xi(n-1)+h;
pi(2*(i+1)-1)=subs(px,x,xi(n)-h/2); pi(2*(i+1))=subs(px,x,xi(n)+h/2);
qi(n-1)=subs(qx,x,xi(n)); fi(n-1)=subs(fx,x,xi(n));
A(n-1,n-2)=-pi(2*(i+1)-1);
A(n-1,n-1)=pi(2*(i+1)-1)+pi(2*(i+1))+h^2*qi(n-1);
B(n-1,1)=h^2*fi(n-1)+pi(2*(i+1))*bet;
iterace=A\B;
vypis(1,:)=xi;
vypis(2,1)=alf;
for i=1:length(iterace)
    reseni(k,i)=iterace(i);
    vypis(2,i+1)=iterace(i);
end
vypis(2,i+2)=bet;
A
B
iterace
if k<3
    norma=2*eps;
else
    suma=0;
    for j=1:rem(length(iterace)+1,2)
        suma=suma+(reseni(k,j)-reseni(k-1,2*j))^2;
    end
    norma=sqrt(suma);
```

```
    end
    h=h/2; n=2*n;
    vypis
    pi;
    qi;
end
reseni;
```

Literatura

[1] J. Baštinec, M. Novák, *Moderní numerické metody (Elektrotechnika, elektronika, komunikační a řidicí technika)*, Brno, VUT, 2014. (učební text)

[2] J. Baštinec, M. Novák, *Moderní numerické metody, Sbírka příkladů.*, Brno, VUT, 2010. (dostupné po přihlášení do IS VUT)

[3] B.Fajmon, I. Hlavičková, M. Novák, *Matematika 3 (Elektrotechnika, elektronika, komunikační a řidicí technika)*, Brno, VUT, 2014. (učební text)

[4] M. Novák, *Matematika 3: Sbírka příkladů z numerických metod*, Brno, VUT, 2010. (dostupné po přihlášení do IS VUT)

[5] M. Novák, *Mathematics 3 (Numerical Methods: Exercise Book)*, Brno, VUT, 2014. (dostupné po přihlášení do IS VUT)