VZOR

Řešení 1. písemné práce z Numerických metod

Jméno a příjmení: .... **Osobní číslo:** .....

Část 1.

A) Pro řešení soustavy rovnic $\mathbf{Ax} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 1 & 5 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix},$$

máme použít metodu SOR.
Pro ruční počítání je vhodné nejprve najít předpis pro Gauss-Seidelovu metodu.

GS:

$$\begin{aligned} x_1^{K+1} &= \frac{1}{3}\left(1 - x_2^K\right) \\ x_2^{K+1} &= \frac{1}{4}\left(2 - x_1^{K+1}\right) \\ x_3^{K+1} &= \frac{1}{5}\left(3 - x_2^{K+1}\right) \end{aligned}$$

Vztahy pro metodu SOR získáme pomocí faktu, že $K+1$-ní iterace vektoru řešení $\mathbf{x}^{K+1}$ je lineární kombinací předchozí K-té iterace a $K+1$ iterace získané Gauss-Seidelovou metodou, tj.

$$\mathbf{x}^{K+1} = \omega\, \mathbf{x}_{\mathbf{G}S}^{K+1} \;+\; (1-\omega)\, \mathbf{x}^K.$$

SOR:

$$\begin{aligned} x_1^{K+1} &= \frac{1.05}{3}\left(1 - x_2^K\right) &&- 0.05\, x_1^K \\ x_2^{K+1} &= \frac{1.05}{4}\left(2 - x_1^{K+1}\right) &&- 0.05\, x_2^K \\ x_3^{K+1} &= \frac{1.05}{5}\left(3 - x_2^{K+1}\right) &&- 0.05\, x_3^K \end{aligned}$$

Čísla budeme zapisovat na 4 desetinná místa. V následující tabulce jsou získané iterace.

K	$\mathbf{x}^K$	$\mathbf{x}^K - \mathbf{x}^{K-1}$	$\|\mathbf{x}^K - \mathbf{x}^{K-1}\|$
0	0.0000 0.0000 0.0000		
1	0.3500 0.4331 0.5390	0.3500 0.4331 0.5390	0.5390
2	0.1809 0.4558 0.5073	−0.1690 0.0227 −0.0317	0.1690
3	0.1814 0.4545 0.5091	0.0004 −0.0012 0.0018	0.0018

Pro jednoduchost byla použita maximová norma, tj.

$$\|\mathbf{x}\| = \|\mathbf{x}\|_\infty = \max_i |x_i|$$

Přibližné řešení je $\underline{\mathbf{x}^3 = [0.1814,\ 0.4545,\ 0.5091]^T}$.

B) Aitkenova extrapolační formule slouží k urychlení konvergence iteračních metod, které mají lineární rychlost konvergence, tj. platí

$$\exists q \in (0,1), L \geq 0 : \underbrace{\|\mathbf{x}^{K+1} - \mathbf{x}^*\|}_{\|\mathbf{e}^{K+1}\|} \leq q \underbrace{\|\mathbf{x}^K - \mathbf{x}^*\|}_{\|\mathbf{e}^K\|}, \forall K \geq L.$$

Po vydělení dostaneme

$$\frac{\|\mathbf{e}^{K+1}\|}{\|\mathbf{e}^K\|} \leq q.$$

Předpokládejme, že je tento podíl přibližně konstantní pro $K \geq L$. Potom lze odvodit vztah (pro jednotlivé složky - pro jednoduchost neuvádíme index i, příslušná složka přesného řešení je označena α)

$$\frac{x^{K+1} - \alpha}{x^K - \alpha} \approx \frac{x^K - \alpha}{x^{K-1} - \alpha}.$$

Odtud po úpravě

$$(x^{K+1} - \alpha)(x^{K-1} - \alpha) \approx (x^K - \alpha)^2$$

$$x^{K+1}\, x^{K-1} - (x^{K+1} + x^{K-1})\alpha + \underline{\alpha^2} \approx {x^K}^2 - 2x^K\alpha + \underline{\alpha^2}$$

$$x^{K+1}\,x^{K-1} - x^{K\,2} \;\approx\; \alpha\,(x^{K+1} - 2x^K + x^{K-1})$$

$$\alpha \;\approx\; \frac{x^{K+1}\,x^{K-1} - x^{K\,2}}{x^{K+1} - 2x^K + x^{K-1}}.$$

Část 2.

C) Pro nalezení řešení můžeme použít připravené algoritmy v Matlabu. Zadáme data:

```
>> A=[9 3 1; 4 2 1; 1 3 6]

A =
     9     3     1
     4     2     1
     1     3     6

>> b=[3 2 7]'

b =
     3
     2
     7

>> x0=[0 0 0]'

x0 =
     0
     0
     0

>> epsilon=0.001

epsilon =
   1.0000e-03

>> max_iter=50

max_iter =
    50
```

A spustíme algoritmus Jacobiovy metody:

```
>> jacobi(A, b, x0, epsilon, max_iter);

1. iterace = [ 0.333333], rozdíl iterací [ 0.333333]
             [ 1.000000],                [1.000000]
             [ 1.166667],                [1.166667]

2. iterace = [ -0.129630], rozdíl iterací [ -0.462963]
             [ -0.250000],                 [-1.250000]
             [ 0.611111],                [-0.555556]

3. iterace = [ 0.348765], rozdíl iterací [ 0.478395]
             [ 0.953704],                [1.203704]
             [ 1.313272],                [0.702160]

4. iterace = [ -0.130487], rozdíl iterací [ -0.479252]
             [ -0.354167],                 [-1.307870]
             [ 0.631687],                [-0.681584]

5. iterace = [ 0.381201], rozdíl iterací [ 0.511688]
             [ 0.945130],                [1.299297]
             [ 1.365498],                [0.733811]

6. iterace = [ -0.133432], rozdíl iterací [ -0.514634]
             [ -0.445152],                 [-1.390282]
             [ 0.630568],                [-0.734930]

...


49. iterace = [ 1.305159], rozdíl iterací [ 2.210049]
             [ 3.041675],                [5.802555]
             [ 2.697922],                [3.161710]

50. iterace = [ -0.980327], rozdíl iterací [ -2.285486]
             [ -2.959279],                 [-6.000954]
             [ -0.571697],                 [-3.269619]
```

Nyní spustíme algoritmus Gauss-Seidelovy metody:

```
>> gauss_seidel(A, b, x0, epsilon, max_iter);

1. iterace = [ 0.333333], rozdíl iterací [ 0.333333]
             [ 0.333333],                [0.333333]
             [ 0.944444],                [0.944444]

2. iterace = [ 0.117284], rozdíl iterací [ -0.216049]
             [ 0.293210],                [-0.040123]
             [ 1.000514],                [0.056070]

3. iterace = [ 0.124428], rozdíl iterací [ 0.007144]
             [ 0.250886],                [-0.042324]
             [ 1.020486],                [0.019971]

4. iterace = [ 0.136317], rozdíl iterací [ 0.011889]
             [ 0.217122],                [-0.033764]
             [ 1.035386],                [0.014900]

5. iterace = [ 0.145916], rozdíl iterací [ 0.009599]
             [ 0.190474],                [-0.026648]
             [ 1.047110],                [0.011724]

6. iterace = [ 0.153496], rozdíl iterací [ 0.007580]
             [ 0.169452],                [-0.021022]
             [ 1.056358],                [0.009248]

...

18. iterace = [ 0.180173], rozdíl iterací [ 0.000440]
              [ 0.095471],                [-0.001221]
              [ 1.088902],                [0.000537]

19. iterace = [ 0.180521], rozdíl iterací [ 0.000347]
              [ 0.094508],                [-0.000963]
              [ 1.089326],                [0.000424]

20. iterace = [ 0.180795], rozdíl iterací [ 0.000274]
              [ 0.093748],                [-0.000760]
              [ 1.089660],                [0.000334]
```

A na závěr spustíme algoritmus metody SOR, kde zvolíme relaxační parametr např. $\omega = 1.3$:

```
>> sor(A, b, x0, 1.3, epsilon, max_iter)

1. iterace = [ 0.433333], rozdíl iterací [ 0.433333]
             [ 0.173333],                [0.173333]
             [ 1.310111],                [1.310111]

2. iterace = [ 0.038984], rozdíl iterací [ -0.394349]
             [ 0.295070],                [0.121736]
             [ 0.923392],                [-0.386719]

3. iterace = [ 0.160396], rozdíl iterací [ 0.121412]
             [ 0.194245],                [-0.100824]
             [ 1.078637],                [0.155246]

4. iterace = [ 0.145238], rozdíl iterací [ -0.015157]
             [ 0.162992],                [-0.031253]
             [ 1.055662],                [-0.022975]

5. iterace = [ 0.166647], rozdíl iterací [ 0.021409]
             [ 0.131639],                [-0.031353]
             [ 1.078296],                [0.022634]

6. iterace = [ 0.170542], rozdíl iterací [ 0.003894]
             [ 0.116207],                [-0.015431]
             [ 1.080692],                [0.002397]
...

10. iterace = [ 0.180445], rozdíl iterací [ 0.000883]
              [ 0.094152],                [-0.002150]
              [ 1.089694],                [0.000770]

11. iterace = [ 0.181000], rozdíl iterací [ 0.000556]
              [ 0.092852],                [-0.001300]
              [ 1.090188],                [0.000494]

12. iterace = [ 0.181326], rozdíl iterací [ 0.000325]
              [ 0.092075],                [-0.000777]
              [ 1.090474],                [0.000286]
```

Nyní budeme interpretovat získané výsledky.

Jacobiova metoda ukončila výpočet po 50-ti iteracích a vypsala
$\mathbf{x}_{50} = [-0.980327, -2.959279, -0.571697]^T$

a pro rozdíl posledních dvou iterací
$\mathbf{x}_{50} - \mathbf{x}_{49} = [-2.285486, -6.000954, -3.269619]^T$,

je tedy zřejmé, že výpočet byl ukončen podmínkou na maximální počet iterací a ne při splnění zastavovací podmínky neboť $\|\mathbf{x}_{50} - \mathbf{x}_{49}\|_\infty = 6.000954 > 0.001$.

Z posloupnosti rozdílů dvou po sobě jdoucích iterací je zřejmé, že se hodnoty složek v absolutní hodnotě zvětšují a **Jacobiova metoda nekonverguje**.

Pro kontrolu si samozřejmě můžeme nechat vypsat řešení získané přímo Matlabem

```
>> A\b

ans =
   0.181818181818182
   0.090909090909091
   1.090909090909091
```

Gauss-Seidelova metoda ukončila výpočet po 20-ti iteracích a vypsala
$\mathbf{x}_{20} = [0.180795, 0.093748, 1.089660]^T$

a pro rozdíl posledních dvou iterací
$\mathbf{x}_{20} - \mathbf{x}_{19} = [0.000274, -0.000760, 0.000334]^T$,

je tedy zřejmé, že výpočet byl ukončen zastavovací podmínkou neboť
$\|\mathbf{x}_{20} - \mathbf{x}_{19}\|_2 \approx 8.74203.10^{-4} < 0.001$

a posloupnost norem rozdílů dvou po sobě jdoucích iterací se zmenšovala, tj. **Gauss-Seidelova metoda konverguje**.

Metoda SOR ukončila výpočet po 12-ti iteracích a vypsala
$\mathbf{x}_{12} = [0.181326, 0.092075, 1.090474]^T$

a pro rozdíl posledních dvou iterací
$\mathbf{x}_{12} - \mathbf{x}_{11} = [0.000325, -0.000777, 0.000286]^T$.

Podobně jako v Gauss-Seidelově metodě byl výpočet ukončen zastavovací podmínkou
$\|\mathbf{x}_{12} - \mathbf{x}_{11}\|_2 \approx 8.89466.10^{-4} < 0.001$

a posloupnost norem rozdílů dvou po sobě jdoucích iterací se zmenšovala, tj. **metoda SOR konverguje**.

Lze tedy konstatovat, že pro naše zadání Jacobiova metoda nekonvergovala a Gauss-Seidellova metoda konvergovala pomaleji než metoda SOR pro volbu parametru $\omega = 1.3$.

Pro odhad chyby přibližného řešení získaného pomocí Gauss-Seidelovy metody potřebujeme znát tvar iterační matice $\mathbf{H}_{GS}$. Tu není třeba odvozovat, protože je vypsána v okně Matlabu po spuštění metody:

```
H =
     0  -0.333333333333333  -0.111111111111111
     0   0.666666666666667  -0.277777777777778
     0  -0.277777777777778   0.157407407407407
```

Máme za úkol odhadnou chybu získaného přibližného řešení pro Gauss-Seidelovu metodu, pokud použijeme zastavovací podmínku $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_\infty \leq 10^{-3}$, tj. použít maximovou normu $\|\mathbf{r}\|_\infty = \max_i |r_i|$.

Z vypsané posloupnosti rozdílů dvou po sobě jdoucích iterací je patrné, že zastavovací podmínka s touto normou je splněna již v 19. iteraci (pozn. v algoritmu je implementována zastavovací podmínka s Euklidovskou normou $\|\mathbf{r}\|_2 = \sqrt{\sum_i r_i^2}$, a proto byl výpočet ukončen až ve 20. iteraci).

Pro odhad chyby platí následující vztah:

$$\|\mathbf{x}_k - \mathbf{x}^*\| \leq \frac{\|\mathbf{H}\|}{1 - \|\mathbf{H}\|} \cdot \|\mathbf{x}_k - \mathbf{x}_{k-1}\|.$$

Pro vektory máme použít maximovou normu, tudíž pro matici $\mathbf{H}$ musíme použít řádkovou maticovou normu $\|\mathbf{H}\|_R = \max_i \sum_j |h_{ij}|$, která je maximovou vektorovou normou generována:

```
>> max(sum(abs(H)'))

ans =
   0.944444444444445
```

nebo jinak

```
>> norm(H,inf)

ans =
   0.944444444444445
```

Dostáváme tedy

$$\|\mathbf{x}_{19} - \mathbf{x}^*\|_\infty \leq \frac{\|\mathbf{H}\|_R}{1 - \|\mathbf{H}\|_R} \cdot \underbrace{\|\mathbf{x}_{19} - \mathbf{x}_{18}\|_\infty}_{\leq 0.001},$$

tj.

$$\|\mathbf{x}_{19} - \mathbf{x}^*\|_\infty \leq \frac{0.944444444444445}{1 - 0.944444444444445} \cdot 0.001 = 17.000000000000171 \cdot 0.001 = 0.017.$$

Tento odhad lze interpretovat tak, že přibližné řešení $\mathbf{x}_{19}$ se v každé složce může od přesného řešení lišit až o ± 0.017.