Vysoká škola báňská – Technická univerzita Ostrava
Fakulta stavební

Přednáška z předmětu: Algoritmizace inženýrských výpočtů

Téma č.3: Řešení nelineárních algebraických rovnic

doc. Ing. Martin Krejsa, Ph.D.

Obsah

Kapitola 1

Řešení nelineárních algebraických rovnic

Cíle

Kapitola umožňuje bližší seznámení:

- s principem iteračních metod,
- se základními algoritmy pro stanovení kořenů nelineárních algebraických rovnic,
- s využitím cyklu typu **while** při tvorbě algoritmů.

1.1. Iterace

Slovo iterace může být použito ve stejném významu jako opakování (lat. *iteretur* – opakovat). V matematice se iterací označuje řešení problému postupným opakováním, které vede k přiblížení se k žádoucímu výsledku.

1.1.1. Taylorova řada

Jednoduchý iterační výpočet je možno demonstrovat na Taylorově řadě, tedy zvláštní mocninné řadě, pojmenované po anglickém matematikovi Brooku Taylorovi, který ji publikoval v roce 1712 (metoda aproximace funkce mocninnou řadou byla objevena již roku 1671 Jamesem Gregorym).

Za určitých předpokladů o funkci $f(x)$ v okolí bodu a lze tuto funkci vyjádřit (rozvinout) jako mocninnou řadu. Toto vyjádření funkce prostřednictvím *Taylorovy řady* se označuje jako *Taylorův rozvoj*:

$$\begin{aligned} f(x) &= f(a) + \frac{f(a)'}{1!} \cdot (x-a) + \frac{f(a)''}{2!} \cdot (x-a)^2 + \frac{f(a)^{(3)}}{3!} \cdot (x-a)^3 + \cdots = \\ &= \sum_{k=0}^{\infty} \frac{f(a)^{(k)}}{k!} \cdot (x-a)^k \,. \end{aligned} \tag{1.1}$$

Pro přibližné vyjádření hodnot funkce není nutné vyjadřovat všechny členy Taylorovy řady, členy s vyššími derivacemi lze zanedbat. Tímto způsobem se získá tzv. Taylorův polynom, kterým lze aproximovat hodnoty funkce, která má v daném bodě derivaci, pomocí polynomu, jehož koeficienty závisí na derivacích funkce v tomto bodě.

Obsah
4. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.1. Stanovte s využitím prvních deseti členů Taylorova rozvoje hodnotu funkce e^x pro $x = 1$.

Řešení. Vztah pro Taylorův rozvoj, kterým lze stanovit hodnotu funkce e^x, má následující tvar:

$$f(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^{(n)}}{n!} \,. \tag{1.2}$$

Vztah (1.2) je platný pro $x \in < -\infty, \infty >$.

Algoritmicky lze daný výpočet vyjádřit pomocí postupu 1.

Vstup : n, x
Výstup: $f(x)$

$y \leftarrow 1$
for $i \leftarrow 1, 2, 3, \ldots, n-1$ **do**
  $y \leftarrow y + \frac{x^i}{i!}$
end
$f(x) \leftarrow y$

Algoritmus 1: Stanovení hodnoty funkce e^x s využitím Taylorova rozvoje

Obsah
5. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Skript programu MATLAB pak může obsahovat následující instrukce:

```
function y=ecko(n,x)
if n<2
  error('Počet členů Taylorova rozvoje n musí být nejméně 2!')
end
y=1;
for i=1:n-1
  y=y+(x^i)/factorial(i);
end
```

Po vyvolání této m-funkce:

```
ecko(10,1)
```

lze získat výsledek:

```
ans =
   2.718281525573192
```

Obsah

6. strana ze 44

Zavřít dokument

Konec

Celá obrazovka/Okno

Průběh postupného zpřesňování výsledku s přibývajícím počtem členů Taylorova rozvoje lze zobrazit pomocí následující tabulky:

```
   i     f(xi)          f(xi)-e^x
-------------------------------------
   1   1.00000000   -1.71828183e+000
   2   2.00000000   -7.18281828e-001
   3   2.50000000   -2.18281828e-001
   4   2.66666667   -5.16151618e-002
   5   2.70833333   -9.94849513e-003
   6   2.71666667   -1.61516179e-003
   7   2.71805556   -2.26272903e-004
   8   2.71825397   -2.78602051e-005
   9   2.71827877   -3.05861778e-006
  10   2.71828153   -3.02885853e-007
```

kde poslední sloupec představuje rozdíl dílčího výsledku s přesnou hodnotou e^x, vyvolanou např. s využitím funkce `exp(x)`.

▲

Poznámka 1.2. V m-souboru s názvem `ecko` byla využita funkce `error`, která zobrazí chybovou zprávu a ukončí činnost funkce.

Poznámka 1.3. Pro výpočet hodnoty faktoriálu čísla n byla použita funkce systému MATLAB s názvem `factorial`.

Obsah
7. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

1.1.2. Zastavovací podmínka cyklu

V příkladu 1.1 byl při výpočtu aproximace hodnoty funkce s využitím Taylorova rozvoje zadán přesný počet požadovaných členů rozvoje (tím pádem bylo použito cyklu `for`). V praxi se ale nejčastěji používá tzv. „zastavovací podmínka cyklu“, která může nabývat tvaru:

$$|x_k - x_{k-1}| < \varepsilon \,, \tag{1.3}$$

kde x_{k-1}, x_k jsou členy Taylorova rozvoje na $(k-1)$. a k. místě a ε dostatečně malé číslo, označované jako tolerance nepřesnosti výsledku.

1.1.3. Rekurentní vzorec

Rekurentní vzorec určuje členy posloupnosti pomocí znalosti jednoho nebo více předcházejících prvků. Součástí každého rekurentního vzorce musí být zadání prvního, případně několika prvních členů posloupnosti. Nevýhodou zadání pomocí rekurentního vzorce je skutečnost, že libovolný prvek posloupnosti lze určit jen tehdy, pokud jsou známy členy předcházející. První prvek dané posloupnosti, který bývá nazýván „nultou aproximací“, je potřeba vždy odhadnou.

Obsah
8. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.4. S využitím tzv. „Heronova vzorce“ stanovte hodnotu funkce $f(x) = \sqrt{x}$ pro $x = 2$ s tolerancí nepřesnosti výsledku $\varepsilon = 0,001$.

Řešení. Obecný Heronův vzorec pro určení hodnoty funkce $f(x) = \sqrt{x}$ je rekurentní a nabývá tvar:

$$f(x) = y_k = \frac{1}{2}\left(y_{k-1} + \frac{x}{y_{k-1}}\right) , \tag{1.4}$$

kde y_{k-1}, y_k jsou opět $(k-1)$. a k. členy řady. Vztah (1.4) je platný pro $x > 0$.

Výpočetní postup pro danou úlohu lze vyjádřit pomocí algoritmu 2.

Vstup : x, ε
Výstup: $f(x)$

$y_1 \leftarrow x$
$y_2 \leftarrow \frac{1}{2} \cdot \left(y_1 + \frac{x}{y_1}\right)$
while $|y_1 - y_2| \geqq \varepsilon$ **do**
  $y_1 \leftarrow y_2$
  $y_2 \leftarrow \frac{1}{2} \cdot \left(y_1 + \frac{x}{y_1}\right)$
end
$f(x) \leftarrow y_2$

Algoritmus 2: Stanovení hodnoty funkce $f(x) = \sqrt{x}$ s využitím Heronova vzorce

Obsah
9. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Skript programu MATLAB pak může obsahovat následující instrukce:

```
function y=odmocnina(x,tol)
if x<=0
  error('Není splněna podmínka x>0!')
end
y1=x; y2=1/2*(y1+x/y1);
while abs(y1-y2)>tol
  y1=y2;
  y2=1/2*(y1+x/y1);
end
y=y2;
```

Po vyvolání této m-funkce, např.:

```
odmocnina(2,0.001)
```

lze získat odpovídající výsledek:

```
ans =
   1.414213562374690
```

Obsah

10. strana ze 44

Zavřít dokument

Konec

Celá obrazovka/Okno

Průběh postupného zpřesňování výsledku je zobrazen v následující tabulce:

```
   i     f(xi)         f(xi)-x^0.5
-----------------------------------
   1   2.00000000   5.85786438e-001
   2   1.50000000   8.57864376e-002
   3   1.41666667   2.45310429e-003
   4   1.41421569   2.12390141e-006
   5   1.41421356   1.59472435e-012
```

Poslední sloupec tabulky obsahuje hodnoty rozdílu dílčího výsledku s přesnou hodnotou $\sqrt{x}$, vyvolanou např. s využitím funkce programu MATLAB s názvem `sqrt(x)`.

Odchylku od přesného řešení lze rovněž zobrazit např. vyvoláním příkazu:

```
odmocnina(2,0.001)^2
```

který zobrazí:

```
ans =
   2.000000000004511
```

▲

Obsah

11. strana ze 44

Zavřít dokument

Konec

Celá obrazovka/Okno

Příklad 1.5. Ověřte platnost Taylorova rozvoje pro výpočet hodnoty funkce $\sin(x)$, jenž he uváděn ve tvaru:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{(2n+1)}}{(2n+1)!}\,, \tag{1.5}$$

s požadovanou tolerancí nepřesnosti výsledku $\varepsilon = 0,001$. Vztah (1.5) je platný pro $x \in < -\infty, \infty >$.

Příklad 1.6. Ověřte platnost Taylorova rozvoje pro výpočet hodnoty funkce $\cos(x)$, který nabývá tvar:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n}}{(2n)!}\,. \tag{1.6}$$

s požadovanou tolerancí nepřesnosti výsledku $\varepsilon = 0,001$. Vztah (1.6) je platný pro $x \in < -\infty, \infty >$.

Obsah
12. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2. Iterační metody řešení nelineárních algebraických rovnic

Principů, popsaných v kap. 1.1, lze využít například při řešení nelineárních algebraických rovnic. Jednotlivé metody, popsané v následujícím textu, se liší zejména v rychlosti konvergence a univerzálnosti použití.

Konvergence je pojem označující sbíhání, sbíhavost, sbližování, popř. vývoj, který vede ke sblížení. O daných vlastnostech, které se sbližují, lze tvrdit, že konvergují. Objekty, procesy, vlastnosti apod., které se účastní konvergence, se označují jako konvergentní (výjimečně též jako konvergenční), např. konvergentní řada v matematice. V matematice je konvergence úzce spojena s pojmem limita. Opakem konvergence je **divergence**.

1.2.1. Prostá iterace

Řešenou rovnici:

$$f(x) = 0 \,, \tag{1.7}$$

je nutno nejprve upravit na tvar:

$$x = g(x) \,, \tag{1.8}$$

což lze většinou provést několika způsoby. Předpokladem výpočtu je skutečnost, že existuje interval $< a_0, b_0 >$, který spadá do definičního oboru i oboru spojitosti funkcí $f(x)$ a $g(x)$, jenž rovněž obsahuje společný kořen obou rovnic. Z tohoto intervalu je nutno zvolit i hodnotu nulté aproximace.

Obsah
13. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Poznámka 1.7. Nevýhodou této metody je v případě nevhodně zvolené funkce $g(x)$ konvergence řešení ke kořenu, který neleží v intervalu $< a_0, b_0 >$. Vyřešený kořen pak není společný pro obě rovnice, nýbrž je kořenem pouze rovnice $g(x)$.

Algoritmicky lze výpočet pro k iteračních kroků vyjádřit postupem algoritmu 3.

Vstup : x_0, k
Výstup: $f(x)$

for $i \leftarrow 1, 2, 3, \ldots, k$ **do**
| $x_i \leftarrow g(x_{i-1})$
end

$f(x) \leftarrow x_k$

Algoritmus 3: Stanovení kořene rovnice $f(x)$ s využitím metody prosté iterace

Zápis ve formě m-funkce pak vypadá následovně:

```
function xc=prosta_it(g,x0,k)
x(1)=x0;
for i=1:k
  x(i+1)=g(x(i));
end
x'
xc=x(k+1);
```

V seznamu parametrů m-funkce `prosta_it` je obsažena i řešená funkce g. Tu lze definovat prostřednictvím proměnné pomocí funkce programu MATLAB s názvem `inline`.

Obsah
14. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.8. Metodou prosté iterace stanovte aproximaci kořene rovnice:

$$f(x) = \left(\frac{x}{2}\right)^2 - \sin(x) = 0\ . \tag{1.9}$$

Vstupní parametry zvolte: počet iteračních cyklů $k = 10$, nultá aproximace $x_0 = 1,5$.

Řešení. Rovnici (1.9) lze upravit na tvar:

$$x = 2\sqrt{\sin(x)}\ . \tag{1.10}$$

Do programu MATLAB pak lze výraz (1.10) zadat příkazem:

```
g=inline('2*sqrt(sin(x))')
```

Nyní lze spustit výpočet povelem:

```
xc=prosta_it(g,1.5,10)
```

Seznam prvních deseti iterací je následující (desátou, poslední iteraci lze považovat za vyřešený kořen nelineární rovnice):

```
1.500000000000000
1.997493415863046
1.908232350897023
1.942788324690179
1.930393907098011
1.934981663979237
1.933302091730397
```

Obsah
15. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

```
    1.933919512286077
    1.933692884811449
    1.933776115524553
    1.933745554580009
```

Po dosazení vypočteného kořene 1.933745554580009 do původní rovnice (1.9) vychází:

```
ans =
   -1.085057641792009e-005
```

což dostatečně vypovídá o nepřesnosti daného řešení. ▲

Příklad 1.9. Proveďte výpočet kořene nelineární rovnice z příkladu 1.8 s využitím metody prosté iterace s nultou aproximací $x_0 = 2,0$ a $k = 20$ iteračními cykly. Zobrazte dosaženou přesnost řešení.

Příklad 1.10. Upravte popsaný algoritmus metody prosté iterace tak, aby byl cyklus zakončen podmínkou (1.3). Příklad 1.8 pak vyřešte s parametrem $\varepsilon = 0,05$, vyjadřujícím toleranci nepřesnosti výsledku.

1.2.2. Metoda bisekce (půlení intervalů)

Touto metodou lze přibližně (se zadanou tolerancí ε) vyřešit kořen reálné nelineární rovnice $f(x) = 0$, která je spojitá pro $x \in < a_0; b_0 >$. Také se předpokládá, že platí: $f(a_0) \cdot f(b_0) < 0$, tj. sign $f(a_0) = -$ sign $f(b_0)$.

Postup výpočtu kořene nelineární rovnice $f(x) = 0$ metodou bisekce je vyjádřen algoritmem 4.

Po n výpočetních krocích bude mít zkoumaný interval s hledaným kořenem rovnice $f(x) = 0$ šířku:

$$b_n - a_n = \frac{1}{2}\,(b_{n-1} - a_{n-1}) = \frac{1}{2^2}\,(b_{n-2} - a_{n-2}) = \ \ldots\ = \frac{1}{2^n}\,(b_0 - a_0)\ . \tag{1.11}$$

Pro odhad chyby (nepřesnosti), kterou je zatížen výsledek, pak platí:

$$|a_n - \alpha| \leqq \frac{b_0 - a_0}{2^n}\ , \tag{1.12}$$

resp.

$$|b_n - \alpha| \leqq \frac{b_0 - a_0}{2^n}\ , \tag{1.13}$$

kde α představuje přesnou hodnotu kořene rovnice $f(x) = 0$.

Poznámka 1.11. Metoda bisekce konverguje velice pomalu (v [1] se uvádí zlepšení přesnosti o 1 desetinné místo po 3,3 výpočetních krocích, neboť $10^{-1} \approx 2^{-3,3}$). Na rychlost konvergence však nemá vliv řešená funkce $f(x)$. Výhodou je jednoduchost aplikace i možnost přesného odhadu počtu kroků, potřebných k dosažení požadované přesnosti.

Obsah
17. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Vstup : $\varepsilon > 0$, a_0, b_0
Výstup: x_c

$a \leftarrow a_0$
$b \leftarrow b_0$
while $|a - b| \geqq \varepsilon$ **do**
  $c \leftarrow \frac{a+b}{2}$
  if $f(c) = 0$ **then**
    $x_c \leftarrow c$ /* c je výsledný kořen rovnice */
    konec;
  end
  if $\text{sign}(f_c) \cdot \text{sign}(f_a) < 0$ **then**
    $b \leftarrow c$ /* nové hranice intervalu jsou $< a, c >$ */
  else
    $a \leftarrow c$ /* nové hranice intervalu jsou $< c, b >$ */
  end
end
$x_c \leftarrow \frac{a+b}{2}$

Algoritmus 4: Algoritmus metody bisekce (půlení intervalů)

Obsah
18. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.12. Metodou bisekce stanovte aproximaci kořene rovnice:

$$f(x) = x^3 + x - 1\ . \tag{1.14}$$

Vstupní parametry zvolte: $\varepsilon = 0,001$, $a_0 = 0$ a $b_0 = 1$.

Řešení. Funkci pro výpočet kořene nelineární rovnice metodou bisekce lze naprogramovat prostřednictvím m-souboru následovně:

```
function xc=bisect(f,a,b,tol)
if sign(f(a))*sign(f(b))>=0
  error('Není splněna podmínka f(a)*f(b)<0!')
end
fa=f(a);
fb=f(b);
while(b-a)>tol
  c=(a+b)/2;
  fc=f(c);
  if fc==0
    xc=c
    break
  end
  if sign(fc)*sign(fa)<0  %nové hranice intervalu jsou <a,c>
    b=c;
    fb=fc;
  else                    %nové hranice intervalu jsou <c,b>
```

Obsah
19. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

```
    a=c;
    fa=fc;
  end
end
xc=(a+b)/2;
```

V seznamu parametrů je obsažena i řešená funkce f. Tu lze definovat prostřednictvím proměnné pomocí funkce programu MATLAB s názvem `inline`:

```
f=inline('x^3+x-1')
```

Nyní lze spustit výpočet zadáním příkazu:

```
xc=bisect(f,0,1,0.001)
```

Výsledek je pak:

```
xc =
    0.6821
```

Obsah
20. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Pro demonstraci práce funkce bisect je možno zobrazit jak se mění obsah proměnných a, b a c během iterace. Sloupce označené $f(a)$, $f(b)$ a $f(c)$ pak obsahují hodnotu -1 nebo $+1$ podle toho, jestli je příslušná hodnota funkce $f(a)$, $f(b)$ a $f(c)$ záporná či kladná.

```
  i      a     f(a)     c     f(c)     b     f(b)
-----------------------------------------------
  0   0.0000   -1   0.5000   -1   1.0000    1
  1   0.5000   -1   0.7500    1   1.0000    1
  2   0.5000   -1   0.6250   -1   0.7500    1
  3   0.6250   -1   0.6875    1   0.7500    1
  4   0.6250   -1   0.6563   -1   0.6875    1
  5   0.6563   -1   0.6719   -1   0.6875    1
  6   0.6719   -1   0.6797   -1   0.6875    1
  7   0.6797   -1   0.6836    1   0.6875    1
  8   0.6797   -1   0.6816   -1   0.6836    1
  9   0.6816   -1   0.6826    1   0.6836    1
 10   0.6816   -1   0.6821   -1   0.6826    1
```

▲

Příklad 1.13. Metodou bisekce stanovte aproximaci kořene rovnice (1.9) z příkladu 1.8. Vstupní parametry zvolte: $\varepsilon = 0,05$, $a_0 = 1,5$ a $b_0 = 2,0$.

Obsah
21. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.14. Pro staticky neurčitý nosník z příkladu předchozí přednášky stanovte největší průhyb na konstrukci s využitím metody bisekce.

Řešení. V průřezu s největším průhybem platí: $w_z(x)' = \varphi_y(x) = 0$ a pro jeho určení je tedy nutno vyřešit kořeny polynomu 3. stupně. Vztáhne-li se odvozená rovnice pro pootočení k obecnému vyjádření polynomem n-tého stupně, vektor $\mathbf{c}$ s prvky $[c_n, c_{n-1}, \ldots, c_1, c_0]$ pak nabývá tvar:

$$\mathbf{c} = \left[\frac{q_z}{6}\,, \quad -\frac{3}{16}\,q_z l\,, \quad 0\,, \quad \frac{1}{48}\,q_z l^3 \right]\,. \tag{1.15}$$

Výpočet kořene polynomu metodou bisekce lze provést s nepatrně upravenou m-funkcí z příkladu 1.12:

```
function xc=bisekce(a,b,d,tol)
if sign(horner(3,d,a))*sign(horner(3,d,b))>=0
  error('Není splněna podmínka f(a)*f(b)<0!')
end
fa=horner(3,d,a);
fb=horner(3,d,b);
while b-a>tol
  c=(a+b)/2;
  fc=horner(3,d,c);
  if fc==0
    xc=c
    break
  end
  if sign(fc)*sign(fa)<0  %nové hranice intervalu jsou <a,c>
    b=c;
```

Obsah
22. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

```
    fb=fc;
  else                    %nové hranice intervalu jsou <c,b>
    a=c;
    fa=fc;
  end
end
xc=(a+b)/2
```

Největší průhyb se pak získá dosazením výsledného kořene polynomu pootočení z intervalu $(0; l)$ do rovnice ohybové čáry.

Celá posloupnost příkazů, potřebných pro určení průřezu s nulovým pootočením i největšího průhybu na nosníku, může být následující:

```
format long
qz=4000;
l=6;
b=0.02;
h=0.15;
Iy=1/12*b*h^3;
E=2.1*10^11;
c=[0 1/(48*E*Iy)*qz*l^3 0 -3/(48*E*Iy)*qz*l qz/(24*E*Iy)];
fi=[1/(48*E*Iy)*qz*l^3 0 -3/(16*E*Iy)*qz*l qz/(6*E*Iy)];
xmax=bisekce(0,l-0.00001,fi,0.00000000000001)
poot=horner(3,fi,xmax)
wmax=horner(4,c,xmax)*1000
```

Obsah
23. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Pro dané zadání pak vychází největší průhyb v průřezu se souřadnicí:

```
xmax =
   2.529210992451759
```

Pootočení je v tomto řezu:

```
poot =
   1.734723475976807e-017
```

a samotný maximální průhyb pak v milimetrech vychází:

```
wmax =
   23.769036533008371
```

▲

Poznámka 1.15. Pro znázornění principu fungování algoritmu metody bisekce byl na obr. 1.1 vytvořen graf pootočení nosníku z předchozího příkladu posloupností příkazů:

```
x=linspace(0,l,100);
plot([0,l],[0,0],'r-')
for i=1:100, y(i)=horner(3,fi,x(i)); end
plot(x,y,'b-');
title('Graf pootočení fi(x)');
xlabel('x');
ylabel('fi(x)');
```

Obsah

24. strana ze 44

Zavřít dokument

Konec

Celá obrazovka/Okno

V grafu je vyznačena cesta iteračního výpočtu od středu intervalu směrem k řešenému kořenu rovnice pootočení.

Poznámka 1.16. V grafu na obr. 1.1 lze pozorovat, že hraniční hodnota pootočení v bodě b je nulová. Pokud by tedy byly při vyvolání funkce bisekce zvoleny meze $a_0 = 0$ a $b_0 = l$, nebyla by splněna požadovaná podmínka pro řešení touto metodou: $f(a_0) \cdot f(b_0) < 0$. Z tohoto důvodu byla hodnota vstupního parametru $b_0 = l$ nepatrně upravena (na hodnotu $b_0 = l - 0.00001$):

```
xmax=bisekce(0,l-0.00001,fi,0.00000000000001,1000)
```

1.2.3. Metoda regula falsi

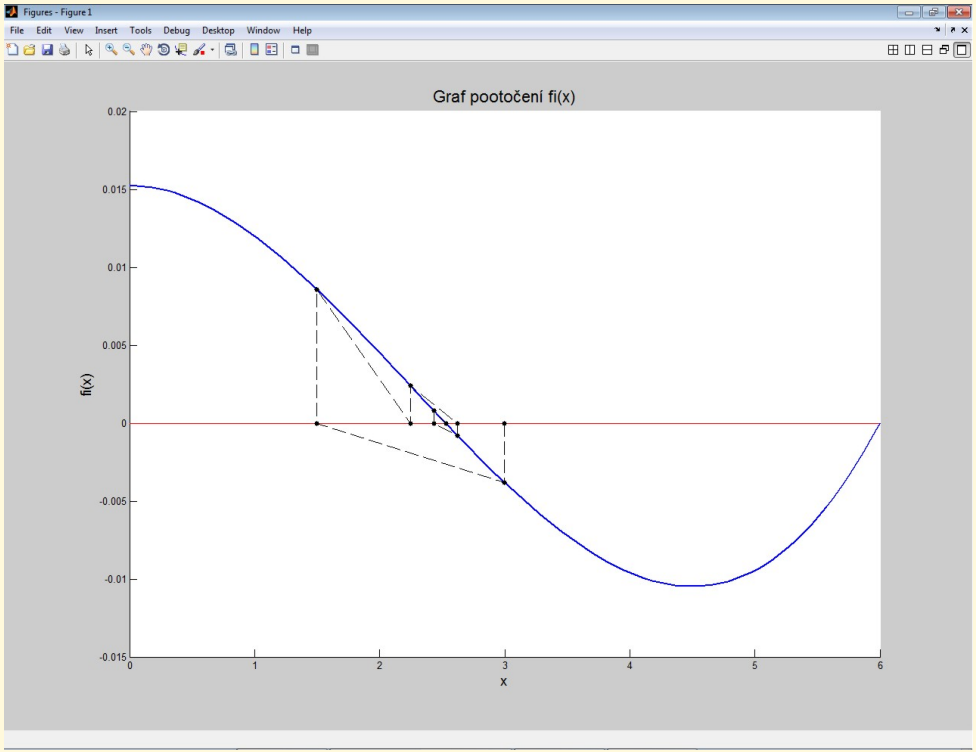
Touto metodou lze přibližně (se zadanou tolerancí ε) rovněž vyřešit kořen reálné nelineární rovnice $f(x) = 0$, která je spojitá pro $x \in < a; b >$. Předpoklad: $f(a_0) \cdot f(b_0) < 0$, tj. $\operatorname{sign} f(a_0) = -\operatorname{sign} f(b_0)$ zůstává stále v platnosti, neboť odpovídá skutečnosti, že v intervalu $< a; b >$ existuje reálný kořen uvedené rovnice.

Metoda regula falsi stanoví nejprve průsečík sečny křivky $f(x)$ sestrojené v bodech $[a, f(a)]$ a $[b, f(b)]$ podle vztahu:

$$s = a - \frac{f(a)}{f(b) - f(a)} \cdot (b - a) \; . \tag{1.16}$$

V případě, že $\operatorname{sign}(f(s)) = \operatorname{sign}(f(a))$, změní se zkoumaný interval na $< s, b >$, v opačném případě na $< a, s >$ a výpočet průsečíku sečny křivky $f(x)$ pokračuje s využitím rekurentního vzorce (1.16) dokud není splněna zastavovací podmínka.

Obsah
25. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno



Obr. 1.1 Graf pootočení staticky neurčitého nosníku a iterace k průřezu s největším průhybem metodou bisekce

Obsah
26. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Postup výpočtu kořene nelineární rovnice $f(x) = 0$ metodou regula falsi je vyjádřen algoritmem 5.

Vstup : $\varepsilon > 0$, a, b
Výstup: x_c

$s \leftarrow a - \dfrac{f(a)}{f(b) - f(a)} \cdot (b - a)$

while $|f(s)| \geqq \varepsilon$ **do**

  if $\text{sign}(f(s)) = \text{sign}(f(a))$ **then**

    $a \leftarrow s$ `/* nové hranice intervalu jsou` $< s, b >$ `*/`

  else

    $b \leftarrow s$ `/* nové hranice intervalu jsou` $< a, s >$ `*/`

  end

  $s \leftarrow a - \dfrac{f(a)}{f(b) - f(a)} \cdot (b - a)$

end

$x_c \leftarrow s$

Algoritmus 5: Algoritmus metody regula falsi

Příklad 1.17. Metodou regula falsi stanovte aproximaci kořene rovnice (1.14), která byla popsána v příkladu 1.12. Vstupní parametry zvolte obdobné $\varepsilon = 0,001$, $a_0 = 0$ a $b_0 = 1$.

Řešení. Zápis funkce pro výpočet kořene nelineární rovnice metodou regula falsi ve formě skriptu programu MATLAB může být následující:

Obsah
27. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

```
function xc=regula(f,a,b,tol)
if sign(f(a))*sign(f(b))>=0
  error('Není splněna podmínka f(a)*f(b)<0!')
end
fa=f(a);
fb=f(b);
s=a-fa/(fb-fa)*(b-a);
fs=f(s);
while abs(fs)>=tol
  if sign(fs)==sign(fa)  %nové hranice intervalu jsou <s,b>
    a=s;
    fa=fs;
  else                   %nové hranice intervalu jsou <a,s>
    b=s;
    fb=fs;
  end
  s=a-fa/(fb-fa)*(b-a);
  fs=f(s);
end
xc=s;
```

V seznamu parametrů je opět obsažena i řešená funkce f, kterou lze definovat již známou funkcí `inline`. Výpočet bude spuštěn příkazem:

```
xc=regula(f,0,1,0.001)
```

Obsah
28. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Výsledek je pak:

```
xc =
   0.682175815962540
```

Činnost funkce regula v průběhu iterace se dá zobrazit v následující tabulce. Lze z ní vyčíst měnící se obsah proměnných a, b a s během iterace i zda-li je příslušná hodnota $f(a)$, $f(s)$ a $f(s)$ záporná či kladná.

```
  i      a     f(a)     s     f(s)     b     f(b)
-----------------------------------------------
  0   0.0000   -1   0.5000   -1   1.0000    1
  1   0.5000   -1   0.6364   -1   1.0000    1
  2   0.6364   -1   0.6712   -1   1.0000    1
  3   0.6712   -1   0.6797   -1   1.0000    1
  4   0.6797   -1   0.6817   -1   1.0000    1
  5   0.6817   -1   0.6822   -1   1.0000    1
```

▲

Poznámka 1.18. Metoda regula falsi konverguje k výslednému kořenu vždy, pokud je jeho existence zaručena vstupními parametry. Konvergence metodou regula falsi probíhá rychleji než tomu bylo u metody bisekce, čemuž odpovídá i průběh výpočtů v příkladu 1.12 a 1.17. Zatímco při požadované toleranci nepřesnosti výsledku $\varepsilon = 0.001$ bylo metodou bisekce dosaženo řešení po 10 krocích, metoda regula falsi nalezla hledaný kořen již po 5 iteracích.

Obsah
29. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.19. Pro staticky neurčitý nosník z příkladu předchozí přednášky stanovte podobně jako v příkladu 1.14 největší průhyb na konstrukci, tentokráte ovšem s využitím metody regula falsi.

Řešení. Pro vstupní údaje $a = 0$, $b = l - 0,001$ a $\varepsilon = 0,0001$ vychází s využitím metody regula falsi největší průhyb v průřezu se souřadnicí:

```
xmax =
   2.529471870690230
```

Pootočení je v tomto řezu:

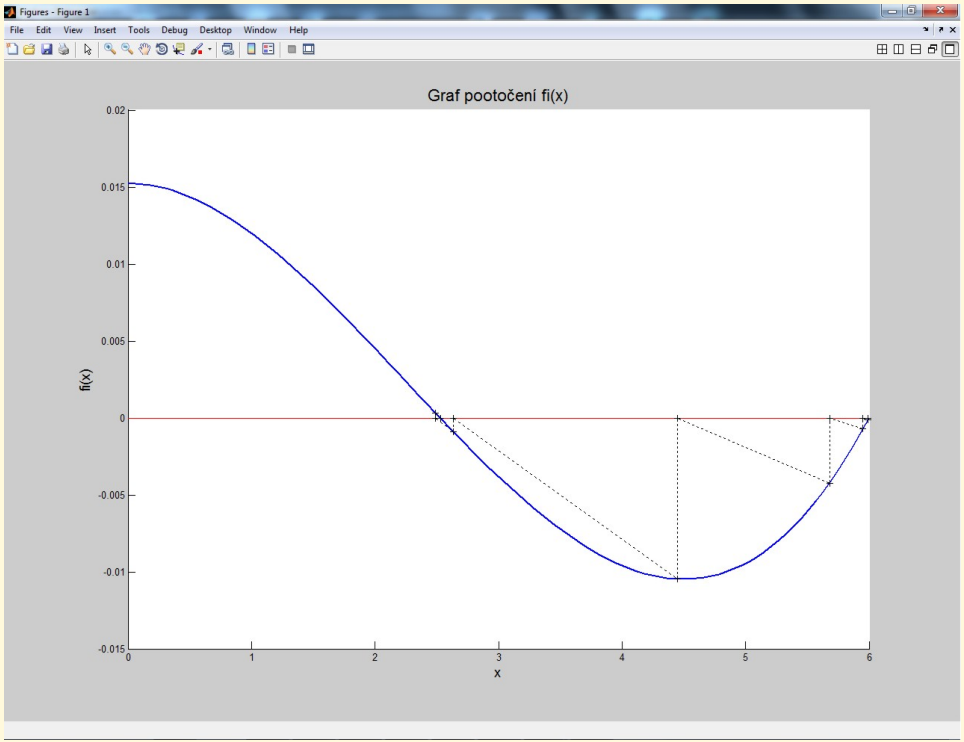
```
poot =
   -2.201632719885452e-006
```

a samotný maximální průhyb pak v milimetrech vychází:

```
wmax =
   23.769036245827937
```

▲

Poznámka 1.20. Pro znázornění principu fungování algoritmu metody regula falsi byl na obr. 1.2 vytvořen graf pootočení nosníku z předchozího příkladu. V grafu je vyznačena cesta iteračního výpočtu od počáteční aproximace směrem k řešenému kořenu rovnice pootočení.



Obr. 1.2 Graf pootočení staticky neurčitého nosníku a iterace k průřezu s největším průhybem metodou regula falsi

Obsah
31. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2.4. Metoda sečen

Při uplatnění tohoto výpočetního postupu se řešená funkce $f(x)$ nahradí lineární funkcí:

$$g(x) = \frac{f(x_k - f(x_{k-1}))}{x_k - x_{k-1}} \cdot (x - x_k) + f(x_k)\,. \tag{1.17}$$

Určení kořene rovnice $g(x)$ pak spočívá v uplatnění rekurentního vzorce:

$$x_{k+1} = x_k - f(x_k) \cdot \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\,. \tag{1.18}$$

Kořen x_{k+1} lze považovat za aproximaci kořene rovnice $f(x) = 0$. Vztah (1.18) představuje dvoukrokovou iterační formuli, neboť k zahájení iteračního výpočtu je potřeba určit dvě počáteční aproximace x_0 a x_1.

Poznámka 1.21. Pokud dvě počáteční aproximace x_0 a x_1 nebudou vhodně zvoleny, nemusí metoda sečen konvergovat. Z tohoto důvodu je nutné algoritmus výpočtu opatřit kontrolou konvergence, příp. počáteční aproximace stanovit s využitím jiné metody.

Příklad 1.22. Metou sečen vyřešte kořen rovnice z příkladu 1.17. Vstupní parametry zvolte: $x_0 = 0$, $x_1 = 1$ a $\varepsilon = 0,001$.

Řešení. Při tvorbě algoritmu i instrukcí m-souboru je možno vyjít z výpočetního postupu metody regula falsi, který je nutné upravit v souvislosti s odlišným rekurentním vzorcem (1.18) metody sečen.

M-funkci s názvem např. `m_secen` pak lze vyvolat příkazem:

```
xc=m_secen(f,0,1,0.001)
```

Obsah
32. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Pro dané vstupní parametry vychází výsledek:

```
xc =
   0.682020419648186
```

Výpočetní postup metody sečen lze nejlépe sledovat postupným výpisem hodnot x_{k-1}, x_k a x_{k+1} i jejích funkčních hodnot:

```
  i   x(k-1)  f(x(k-1))     x(k)     f(x(k))   x(k+1)  f(x(k+1))
-------------------------------------------------------------------
  0   0.0000  -1.000000   1.0000   1.000000   0.5000  -0.375000
  1   1.0000   1.000000   0.5000  -0.375000   0.6364  -0.105935
  2   0.5000  -0.375000   0.6364  -0.105935   0.6901   0.018636
  3   0.6364  -0.105935   0.6901   0.018636   0.6820  -0.000737
```

Pro dosažení výsledku s tolerancí *varepsilon* $= 0,001$ postačily pouze 3 iterační cykly. ▲

Příklad 1.23. Pro staticky neurčitý nosník z příkladu předchozí přednášky stanovte podobně jako v příkladech 1.14 nebo 1.19 největší průhyb na konstrukci, tentokráte ovšem s využitím metody sečen.

Řešení. Pro vstupní údaje $a = 0$, $b = l/2$ a $\varepsilon = 0,0001$ vychází s využitím metody sečen největší průhyb v průřezu se souřadnicí:

```
xmax =
   2.534161490683230
```

Obsah
33. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Pootočení je v tomto řezu:

```
poot =
   -4.176775334049400e-005
```

a samotný maximální průhyb pak v milimetrech vychází:

```
wmax =
   23.768933137770031
```

▲

Poznámka 1.24. Pro znázornění principu fungování algoritmu metody sečen byly na obr. 1.3 a 1.4 vytvořeny grafy pootočení nosníku z předchozího příkladu. V grafech je vyznačena cesta iteračního výpočtu od počáteční aproximace x_k směrem k řešenému kořenu rovnice pootočení. Oba grafy se liší nepatrně pozměněnou hodnotou vstupního parametru b, který byl zadán $b = l - 0,85$, resp. $b = l - 1,0$. Jak již bylo řečeno v pozn. 1.21, metoda je velice citlivá na hodnoty prvních dvou aproximací x_0 a x_1.

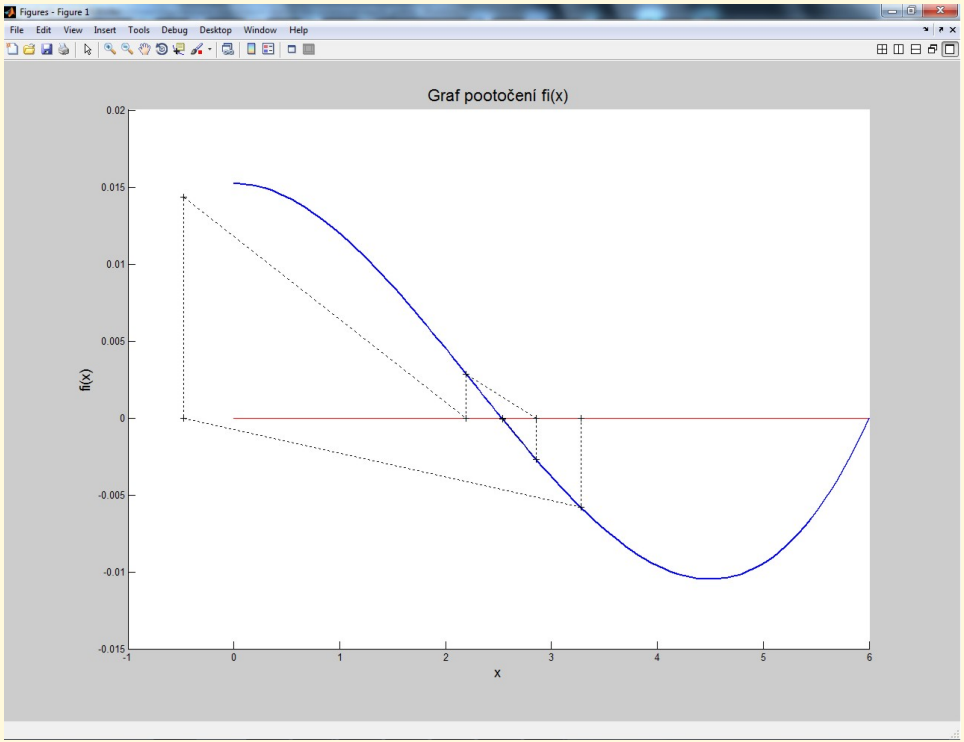
Obsah

34. strana ze 44

Zavřít dokument

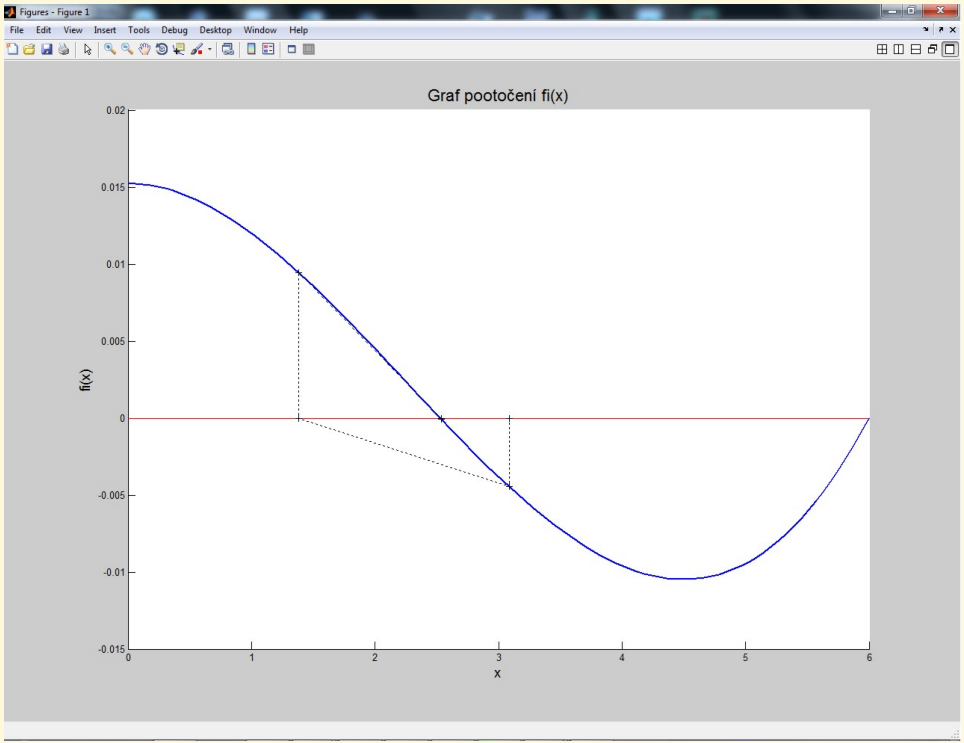
Konec

Celá obrazovka/Okno



Obr. 1.3 Graf pootočení nosníku a iterace k průřezu s největším průhybem metodou sečen se vstupními parametry $a = 0$, $b = l - 0,85$ a $\varepsilon = 0,0001$.

Obsah
35. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno



Obr. 1.4 Graf pootočení nosníku a iterace k průřezu s největším průhybem metodou sečen se vstupními parametry $a = 0$, $b = l - 1,0$ a $\varepsilon = 0,0001$.

Obsah
36. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

1.2.5. Newtonova metoda (metoda tečen)

Pokud jednoduchý reálný kořen rovnice $f(x) = 0$ leží v intervalu $< a, b >$, ve kterém existují i derivace $f'(x), f''(x)$, lze vyjádřit funkci f ve tvaru Taylorova rozvoje v bodě x_0:

$$f(x) = f(x_0) + f'(x_0) \cdot (x - x_0) + \frac{1}{2} \cdot f''(\xi_0) \cdot (x - x_0)^2 \,. \tag{1.19}$$

Rovnici $f(x) = 0$ lze aproximovat lineární rovnicí, kterou tvoří první dva členy tohoto rozvoje (1.19):

$$f(x_0) + f'(x_0) \cdot (x - x_0) = 0 \,. \tag{1.20}$$

Kořen lineární rovnice (1.20) se dá stanovit:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \,. \tag{1.21}$$

Pokud se rovnice (1.20) vyjádří obecně pro x_k:

$$f(x_k) + f'(x_k) \cdot (x - x_k) = 0 \,, \tag{1.22}$$

při řešení lze získat posloupnost, definovanou rekurentní formulí

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \,, \tag{1.23}$$

která vyjadřuje základní myšlenku Newtonovy metody.

Obsah
37. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Poznámka 1.25. Podle [2] je vhodné volit nultou aproximaci x_0 mezi krajními body a, b tak, aby $\text{sign}(f(x_0)) = \text{sign}(f''(x_0))$. Pokud mezi skutečným kořenem počáteční aproximací leží inflexní bod, může posloupnost konvergovat k některému z ostatních kořenů řešené funkce, příp. divergovat nebo oscilovat.

Poznámka 1.26. Algoritmus Newtonovy metody odpovídá výpočetnímu postupu metody prosté iterace pro funkci:

$$\varphi(x) = x - \frac{f(x)}{f'(x)} \,. \tag{1.24}$$

Poznámka 1.27. Oproti předchozím postupům obsahuje rekurentní vzorec posloupnosti derivaci funkce $f(x)$. Numerické derivování bude vysvětleno v některé z dalších kapitol, proto je následující příklad zaměřen pouze na úlohu, kde je derivace řešené funkce $f'(x)$ známa.

Obsah
38. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.28. Pro staticky neurčitý nosník z příkladu předchozí přednášky stanovte podobně jako v příkladech 1.14, 1.19 nebo 1.23 největší průhyb na konstrukci. Pro určení průřezu s největším průhybem použijte Newtonovu metodu.

Řešení. Jak již bylo odvozeno v předchozí přednášce, derivace pootočení je dána:

$$w_z(x)'' = \varphi_y'(x) = -\frac{1}{EI_y} \cdot M_y(x) \,, \tag{1.25}$$

konkrétně tedy:

$$\varphi_y'(x) = \frac{1}{EI_y} \cdot \left(q_z \frac{x^2}{2} - \frac{3}{8} q_z l x \right) \,, \tag{1.26}$$

Vztáhne-li se odvozená rovnice pro pootočení k obecně vyjádřenému polynomu n-tého stupně, vektor c s prvky $[c_n, c_{n-1}, \ldots, c_1, c_0]$ pak nabývá tvar:

$$\mathbf{c} = \left[\; \frac{q_z}{2EI_y} \,, \quad -\frac{3q_z l}{8EI_y} \,, \quad 0 \; \right] \,. \tag{1.27}$$

Sled příkazů pro daný výpočet může vypadat následovně:

```
format long
qz=4000;
l=6;
b=0.02;
h=0.15;
Iy=1/12*b*h^3;
E=2.1*10^11;
c=[0 1/(48*E*Iy)*qz*l^3 0 -3/(48*E*Iy)*qz*l qz/(24*E*Iy)];
```

Obsah
39. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

```
fi=[1/(48*E*Iy)*qz*l^3 0 -3/(16*E*Iy)*qz*l qz/(6*E*Iy)];
m=[0 -3*qz*l/(8*E*Iy) qz/(2*E*Iy)];
xmax=newton_it(3,fi,m,0.0001)
horner(3,fi,xmax)
horner(4,c,xmax)*1000
```

Příkaz `newton_it` vyvolá m-funkci se vstupním parametrem nulté aproximace $x_0 = 3$ metry a tolerancí nepřesnosti výsledku $\varepsilon = 0,0001$, obsahující např. příkazy:

```
function xc=newton_it(a,fi,m,tol)
f1=horner(2,m,a);
f2=horner(3,fi,a);
s=a-f2/f1;
fs=horner(3,fi,s);
while abs(fs)>=tol
  a=s;
  f1=horner(2,m,a);
  f2=horner(3,fi,a);
  s=a-f2/f1;
  fs=horner(3,fi,s);
end
xc=s;
```

Obsah
40. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Řešením je pak pro dané vstupní údaje:

```
xmax =
   2.529166666666667
```

Pootočení je v tomto řezu:

```
poot =
   3.740855052600245e-007
```

Maximální průhyb pak byl v milimetrech určen hodnotou:

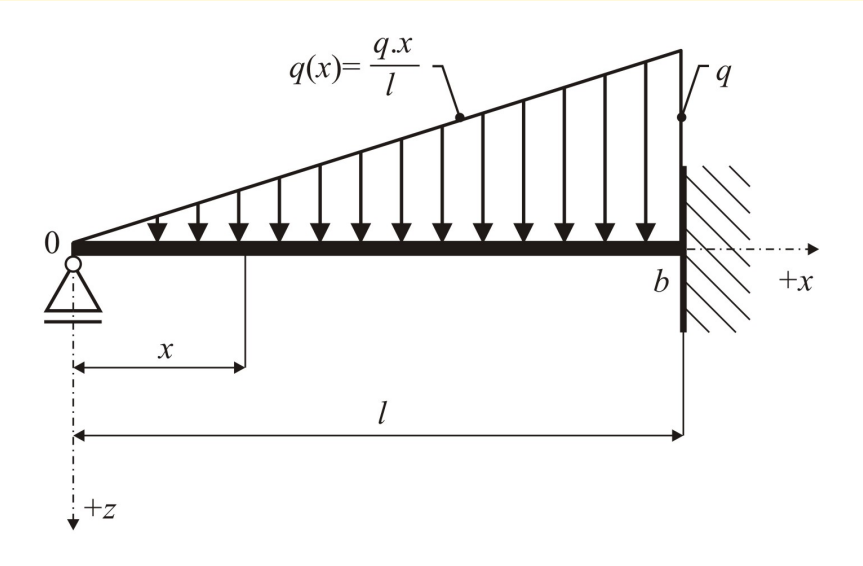
```
wmax =
   23.769036524717556
```

Pokud se provede výpis iterovaných hodnot x_i, $f(x_i)$, $f'(x_i)$, x_{i+1} a $f(x_{i+1})$, lze se přesvědčit, že Newtonova metoda konverguje velice rychle. Dostatečně přesné řešení bylo dosaženo již po prvním iteračním kroku:

```
  i     x(i)     f(x(i))    df(x(i))    x(i+1)   f(x(i+1))
-------------------------------------------------------
  0   3.0000   -0.007619   -0.003810    2.5000    0.000247
  1   2.5000   -0.008466    0.000247    2.5292    0.000000
```

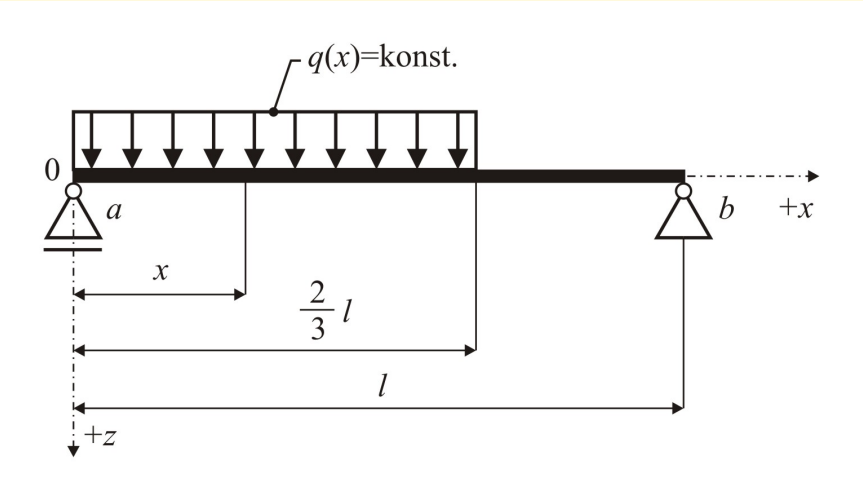
Obsah
41. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Příklad 1.29. Výše vysvětlenými iteračními postupy určete největší průhyb staticky neurčitého nosníku, který je schématicky zobrazen na obr. 1.5.



Obr. 1.5 Statické schéma řešeného staticky neurčitého nosníku

Příklad 1.30. Stanovte největší průhyb staticky určitého nosníku, jehož schéma je zobrazeno na obr. 1.6.



Obr. 1.6 Statické schéma řešeného staticky určitého nosníku

Obsah
43. strana ze 44
Zavřít dokument
Konec
Celá obrazovka/Okno

Literatura

[1] Mika, S. *Numerické metody algebry*. Matematika pro vysoké školy technické. 2. vydání. SNTL - Nakladatelství technické literatury, Praha, 1985. (176 s). (Citováno na s 17.)

[2] Olehla, M. — Tišer, J. *Praktické použití Fortranu*. 2. upravené vydání. Nakladatelství dopravy a spojů, Praha, 1979. (432 s). (Citováno na s 38.)