Vysoká škola báňská – Technická univerzita Ostrava
Fakulta stavební

Přednáška z předmětu: Algoritmizace inženýrských výpočtů

Téma č.2: Základy algoritmizace, výpočet hodnot funkcí

prof. Ing. Martin Krejsa, Ph.D.

Obsah

Kapitola 1

Základy algoritmizace

Cíle

Kapitola by měla sloužit:

- k vysvětlení pojmu algoritmus,
- k položení základů pro tvorbu jednoduchých algoritmů.

Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy. Pojem algoritmu se nejčastěji objevuje při programování, kdy se jím myslí teoretický princip řešení problému (oproti přesnému zápisu v konkrétním programovacím jazyce). Obecně se ale algoritmus může objevit v jakémkoli jiném vědeckém odvětví. Jako jistý druh algoritmu se může chápat i např. kuchyňský recept. V užším smyslu se slovem algoritmus rozumí pouze takové postupy, které splňují požadavky, označované vlastnostmi algoritmu.

1.1. Vlastnosti algoritmu

Následující výčet vlastností algoritmu je uveden např. v [2]:

- **Konečnost** (finitnost)

  Každý algoritmus musí skončit v konečném počtu kroků. Tento počet kroků může být libovolně velký (podle rozsahu a hodnot vstupních údajů), ale pro každý jednotlivý vstup musí být konečný. Postupy, které tuto podmínku nesplňují, se mohou nazývat výpočetní metody. Speciálním příkladem nekonečné výpočetní metody je reaktivní proces, který průběžně reaguje s okolním prostředím. Někteří autoři však mezi algoritmy zahrnují i takovéto postupy.

- **Obecnost** (hromadnost, masovost, univerzálnost)

  Algoritmus neřeší jeden konkrétní problém (např. „jak spočítat $3 \cdot 7$“), ale obecnou třídu obdobných problémů (např. „jak spočítat součin dvou celých čísel“), takže má širokou množinu možných vstupů.

- **Determinovanost**

  Každý krok algoritmu musí být jednoznačně a přesně definován; v každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat, takže pro stejné vstupy lze pokaždé získat stejné výsledky. Protože běžný jazyk obvykle neposkytuje naprostou přesnost a jednoznačnost vyjadřování, byly pro zápis algoritmů navrženy programovací jazyky, ve kterých má každý příkaz jasně definovaný význam. Vyjádření výpočetní metody v programovacím jazyce se nazývá *program*. Některé algoritmy ale determinované nejsou, např. pravděpodobnostní algoritmy s (pseudo)náhodným výběrem.

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

- **Výstup** (resultativnost)

  Algoritmus má alespoň jeden výstup, veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který algoritmus řeší (algoritmus vede od zpracování hodnot k výstupu).

- **Elementárnost**

  Algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků.

V praxi jsou proto předmětem zájmu hlavně takové algoritmy, které jsou v nějakém smyslu kvalitní. Takové algoritmy splňují různá kritéria, měřená např. počtem kroků potřebných pro běh algoritmu, jednoduchost, efektivitu či eleganci. Problematikou efektivity algoritmů, tzn. metodami, jak z několika známých algoritmů řešících konkrétní problém vybrat ten nejlepší, se zabývají odvětví informatiky nazývané *algoritmická analýza* a *teorie složitosti*.

1.2. Elementární algoritmy

Následující výklad je zaměřen na několik elementárních algoritmů, které jsou základem mnoha výpočetních technik a postupů.

1.2.1. Záměna obsahu dvou proměnných

Za nejjednodušší algoritmus lze považovat postup popisující záměnu obsahu dvou proměnných. K této operaci je potřeba třetí, pomocné proměnné. Samotný algoritmus lze pro proměnné a a b i pro pomocnou proměnnou c popsat následovně:

```
c=a;
a=b;
b=c;
```

Byť se jedná o skutečně elementární algoritmus, lze jej společně s logickým rozhodováním využít např. pro vzestupné setřídění vektoru d o 3 prvcích (c je opět pomocná proměnná):

```
function setrid(d)
  if length(d)==3
    if d(1)>d(2)
      c=d(1);
      d(1)=d(2);
      d(2)=c;
    end
    if d(2)>d(3)
      c=d(2);
```

```
      d(2)=d(3);
      d(3)=c;
    end
    if d(1)>d(2)
      c=d(1);
      d(1)=d(2);
      d(2)=c;
    end
    d
  end
end
```

Příklad 1.1. Proveďte vzestupné třídění vektoru `[8 24 2]` s využitím funkce pro třídění vektoru o třech prvcích, vytvořené v předchozím oddíle.

Řešení. Nejprve je potřeba přiřadit vstupní hodnoty vektoru d:

```
d=[8 24 2]
```

Pak je možno vyvolat funkci `setrid`:

```
setrid(d)
```

Výsledkem je:

```
d =
     2     8    24
```

▲

Tento způsob třídění je samozřejmě velmi neefiktivní a v žádném případě jej nelze považovat za univerzální. Je však vhodný pro vysvětlení základů algoritmizace. Kvalitním algoritmům, které jsou zaměřeny na třídění vektorů, bude věnována jedna z následujících kapitol.

Kapitola 2

Výpočet hodnot funkcí

Cíle

Kapitola má za cíl demonstrovat:

- základní přístupy k tvorbě algoritmů pro výpočet hodnot funkcí,
- odlišnosti mezi algoritmy z hlediska jejich efektivity,
- využití cyklů **for** při tvorbě algoritmů,
- provádění tzv. „tabelace funkce“.

2.1. Výpočet hodnoty polynomu

Následující ukázka jednoho z nejzákladnějších algoritmů pro výpočet polynomu byla převzata z [1]. Spočívá v nalezení nejlepšího způsobu určení hodnoty polynomu:

$$f(x) = 2 \cdot x^4 + 3 \cdot x^3 - 3 \cdot x^2 + 5 \cdot x - 1 \tag{2.1}$$

pro konkrétně zadanou hodnotu x, např. $x = \frac{1}{2}$. Nejlepším způsobem výpočtu je chápán algoritmus s nejmenším počtem matematických operací.

Metoda 1:

První způsob vede k přímému určení požadované hodnoty:

$$f(x = \frac{1}{2}) = 2 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} + 3 \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} - 3 \cdot \frac{1}{2} \cdot \frac{1}{2} + 5 \cdot \frac{1}{2} - 1 = \frac{5}{4}\,. \tag{2.2}$$

Při tomto výpočtu bylo provedeno 10 operací násobení a 4 pro součet/rozdíl.

Metoda 2:

Poněkud výhodnější řešení představuje postup, při němž jsou postupně určeny mocniny vstupního parametru x:

$$\frac{1}{2} \cdot \frac{1}{2} = \left(\frac{1}{2}\right)^2 \tag{2.3}$$

$$\left(\frac{1}{2}\right)^2 \cdot \frac{1}{2} = \left(\frac{1}{2}\right)^3 \tag{2.4}$$

$$\left(\frac{1}{2}\right)^3 \cdot \frac{1}{2} = \left(\frac{1}{2}\right)^4 \tag{2.5}$$

Nyní lze provést finální výpočet polynomu:

$$f(x = \frac{1}{2}) = 2 \cdot \left(\frac{1}{2}\right)^4 + 3 \cdot \left(\frac{1}{2}\right)^3 - 3 \cdot \left(\frac{1}{2}\right)^2 + 5 \cdot \left(\frac{1}{2}\right) - 1 = \frac{5}{4}\,. \tag{2.6}$$

Tento algoritmus je poněkud efektivnější než v prvním případě, neboť bylo provedeno celkem 7 operací násobení a stejný počet 4 operací pro součet/rozdíl.

Metoda 3:

Poslední způsob výpočtu hodnoty polynomu, označovaný jako *Hornerova metoda*, předpokládá následující úpravu řešeného polynomu (2.1):

$$\begin{aligned} f(x) &= -1 + x \cdot (5 - 3 \cdot x + 3 \cdot x^2 + 2 \cdot x^3) = \\ &= -1 + x \cdot (5 + x \cdot (-3 + 3 \cdot x + 2 \cdot x^2)) = \\ &= -1 + x \cdot (5 + x \cdot (-3 + x \cdot (3 + 2 \cdot x)))\,. \end{aligned} \tag{2.7}$$

Sled výpočetních operací vypadá pro $x = \frac{1}{2}$ následovně (výpočet probíhá zevnitř směrem ven):

$$\frac{1}{2} \cdot 2 + 3 = 4 \tag{2.8}$$

$$\frac{1}{2} \cdot 4 - 3 = -1 \tag{2.9}$$

$$\frac{1}{2}\cdot(-1)+5=\frac{9}{2} \tag{2.10}$$

$$\frac{1}{2}\cdot\frac{9}{2}-1=\frac{5}{4}\,. \tag{2.11}$$

Celkem je pro určení požadované hodnoty polynomu potřeba pouze čtyř operací násobení i součtu/rozdílu. Jedná se tedy o nejvíce efektivní algoritmus, který lze pro obecně vyjádřený polynom schématicky vyjádřit algoritmem 1.

Vstup : n, $\mathbf{c}=\{c_1,c_2,c_3,\ \dots,\ c_n,c_{n+1}\}$, x
Výstup: $f(x)$

$y \leftarrow c_{n+1}$
for $i \leftarrow n, n-1,\ \dots,\ 2, 1$ **do**
| $y \leftarrow y\cdot x + c_i$
end
$f(x) \leftarrow y$

Algoritmus 1: Hornerova metoda

Algoritmus 1 lze zapsat prostřednictvím m-souboru i v programu Matlab:

```
function y=horner(d,c,x)
y=c(d+1);
for i=d:-1:1
  y=y*x+c(i);
end
```

Vstupní parametry jsou následující: d je stupeň zadávaného polynomu, c je vektor s $d+1$

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

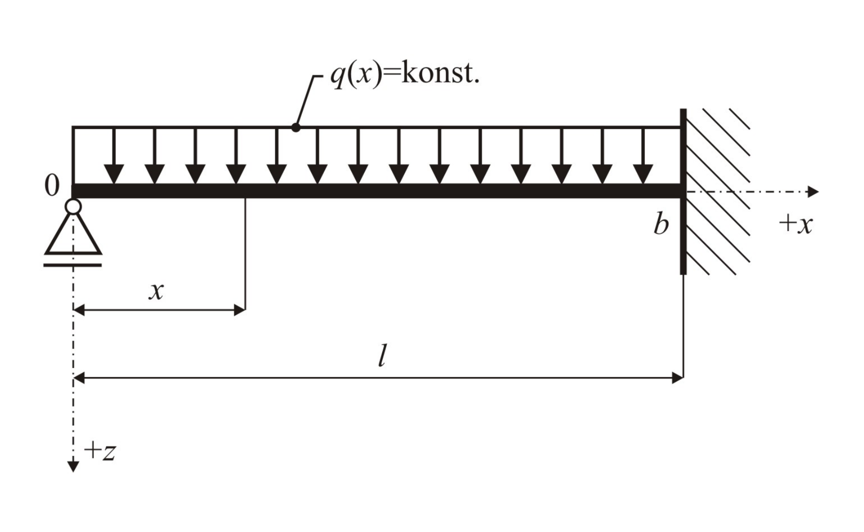
konstantními koeficienty polynomu a x je hodnota, pro níž se polynom určuje. Funkce se pro polynom (2.1) a konkrétně zadanou hodnotu $x = \frac{1}{2}$ bude vyvolávat příkazem:

```
horner(4,[-1 5 -3 3 2],1/2)
```

Výstup z programu MATLAB pak bude:

```
ans =
    1.2500
```

Příklad 2.1. S využitím předchozího postupu určete průhyb v polovině rozpětí staticky neurčitého nosníku, jehož statické schéma je zobrazeno na obr. 2.1. Pro stanovení rovnic polynomů ohybové čáry a pootočení využijte metodu přímé integrace diferenciální rovnice 4. řádu $EI_y w_z(x)'''' = q_z(x)$, kde E je modul pružnosti v tahu a tlaku [MPa], I_y příslušný moment setrvačnosti [m^4] a q_z spojité silové zatížení, působící ve svislém směru [kN/m]. Konkrétní vstupní údaje jsou uvedeny v tabulce 2.1.



Obr. 2.1 Statické schéma řešeného staticky neurčitého nosníku

Spojité silové zatížení q_z :	4 kN/m
Rozpětí nosníku l :	6 m
Šířka obdélníkového průřezu b :	0, 02 m
Výška obdélníkového průřezu h :	0, 15 m
Moment setrvačnosti I_y :	$\frac{1}{12} \cdot 0,02 \cdot 0,15^3 = 5,625 \cdot 10^{-6}$ m^4
Modul pružnosti v tahu a tlaku E :	$2,1 \cdot 10^{11}$ Pa

Tab. 2.1 Vstupní údaje příkladu 2.1

Řešení. V diferenciální rovnici 4. řádu

$$EI_y w_z(x)'''' = q_z(x) \tag{2.12}$$

bude na pravé straně člen odpovídající rovnoměrnému spojitému zatížení, tedy $q(x) = q =$ $=$ konst.

Vztah (2.12) pak lze postupně integrovat:

$$EI_y w_z(x)'''' = q_z \ , \tag{2.13}$$

$$EI_y w_z(x)''' = -V_z(x) = q_z \cdot x + C_1 \ , \tag{2.14}$$

$$EI_y w_z(x)'' = -M_y(x) = q_z \cdot \frac{x^2}{2} + C_1 \cdot x + C_2 \ , \tag{2.15}$$

$$EI_y w_z(x)' = q_z \cdot \frac{x^3}{6} + C_1 \cdot \frac{x^2}{2} + C_2 \cdot x + C_3 \ , \tag{2.16}$$

$$EI_y w_z(x) = q_z \cdot \frac{x^4}{24} + C_1 \cdot \frac{x^3}{6} + C_2 \cdot \frac{x^2}{2} + C_3 \cdot x + C_4 \ . \tag{2.17}$$

Integrační konstanty $C_1, \ldots, C_4$ se určí z okrajových podmínek:

a) Pro $x = 0$ je $M_y(x) = 0$ a platí tedy:

$$M_y(x=0) = -q \cdot \frac{0^2}{2} - C_1 \cdot 0 - C_2 = 0 \,, \tag{2.18}$$

z čehož vyplývá, že $C_2 = 0$,

b) Pro $x = 0$ je $w_z(x) = 0$ a platí tedy:

$$w_z(x=0) = \frac{1}{EI_y} \cdot \left(q_z \cdot \frac{0^4}{24} + C_1 \cdot \frac{0^3}{6} + 0 \cdot \frac{0^2}{2} + C_3 \cdot 0 + C_4 \right) = 0 \,, \tag{2.19}$$

z čehož plyne, že $C_4 = 0$,

c) Pro $x = l$ je $w_z'(x) = 0$ a platí tedy:

$$w_z'(x=l) = \frac{1}{EI_y} \cdot \left(q_z \cdot \frac{l^3}{6} + C_1 \cdot \frac{l^2}{2} + 0 \cdot l + C_3 \right) = 0 \,, \tag{2.20}$$

d) Pro $x = l$ je $w_z(x) = 0$ a platí tedy:

$$w_z(x=l) = \frac{1}{EI_y} \cdot \left(q_z \cdot \frac{l^4}{24} + C_1 \cdot \frac{l^3}{6} + 0 \cdot \frac{l^2}{2} + C_3 \cdot l + 0 \right) = 0 \,. \tag{2.21}$$

Poslední dvě rovnice představují soustavu dvou lineárních rovnic o dvou neznámých integračních konstantách C_1 a C_2. Řešením soustavy lze získat:

$$C_1 = -\frac{3}{8}\, q_z l \ , \tag{2.22}$$

a

$$C_3 = \frac{1}{48}\, q_z l^3 \ . \tag{2.23}$$

Dosazením výsledných hodnot integračních konstant $C_1, \ldots, C_4$ do vztahů 2.14 až 2.17 je možno získat výsledné rovnice pro dvojici statických veličin (posouvající sílu V_z a ohybový moment M_y) i pro obě deformační veličiny - průhyb w_z a pootočení $w_z^{'} = \varphi_y$:

$$V_z(x) = -\left(q_z x - \frac{3}{8}\, q_z l\right) = \frac{3}{8}\, q_z l - q_z x \ , \tag{2.24}$$

$$M_y(x) = -\left(q_z \frac{x^2}{2} - \frac{3}{8}\, q_z l x + 0\right) = \frac{3}{8}\, q_z l x - q_z \frac{x^2}{2} \ , \tag{2.25}$$

$$\begin{aligned} w_z(x)' = \varphi_y(x) &= \frac{1}{EI_y} \cdot \left(q_z \frac{x^3}{6} - \frac{3}{8}\, q_z l\, \frac{x^2}{2} + 0 \cdot x + \frac{1}{48}\, q_z l^3\right) = \\ &= \frac{1}{EI_y} \cdot \left(q_z \frac{x^3}{6} - \frac{3}{16}\, q_z l x^2 + \frac{1}{48}\, q_z l^3\right) \ , \end{aligned} \tag{2.26}$$

$$\begin{aligned} w_z(x) &= \frac{1}{EI_y} \cdot \left(q_z \frac{x^4}{24} - \frac{3}{8}\, q_z l\, \frac{x^3}{6} + 0 \cdot \frac{x^2}{2} + \frac{1}{48}\, q_z l^3 x + 0\right) = \\ &= \frac{1}{2EI_y} \cdot \left(q_z \frac{x^4}{12} - \frac{3}{24}\, q_z l x^3 + \frac{1}{24}\, q_z l^3 x\right) \ . \end{aligned} \tag{2.27}$$

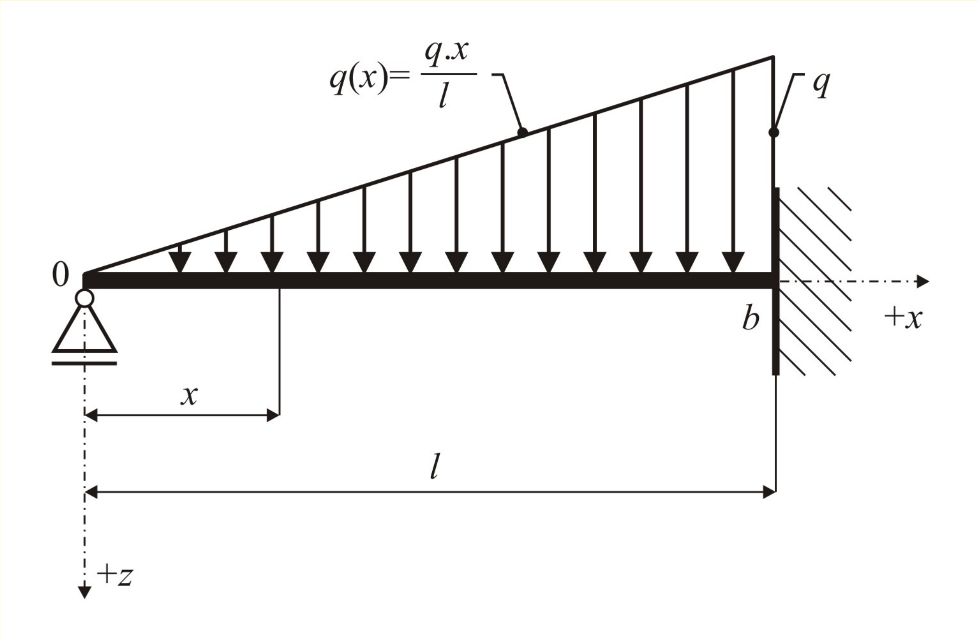
Při využití programu MATLAB i vytvořené m-funkce `horner` pro stanovení požadovaného průhybu bude sled příkazů následující:

```
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)];
horner(4,c,l/2)*1000
```

Výsledný průhyb v milimetrech pak v polovině rozpětí vychází:

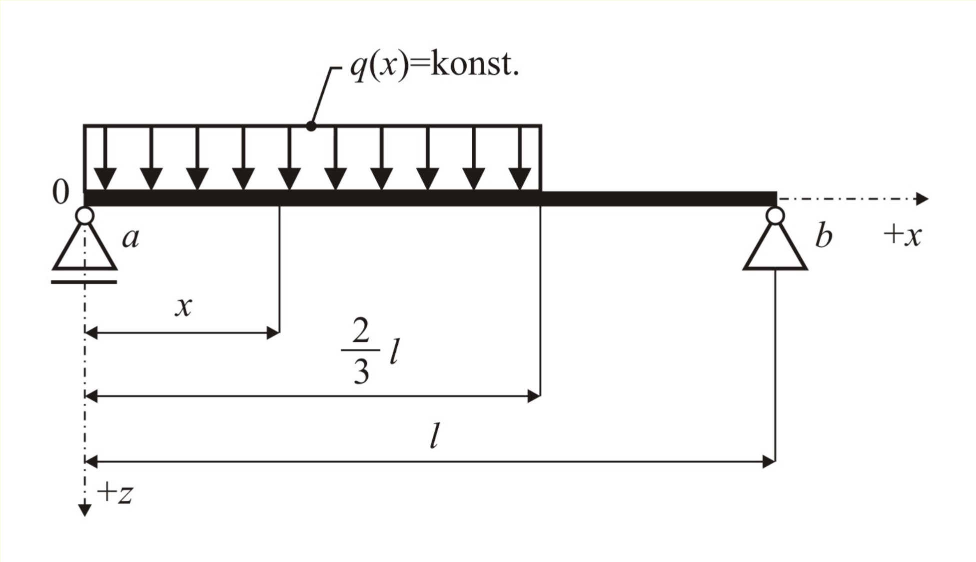
```
ans =
  22.857142857142865
```

Příklad 2.2. Výše vysvětleným postupem určete průhyb v polovině rozpětí staticky neurčitého nosníku, který je schématicky zobrazen na obr. 2.2.



Obr. 2.2 Statické schéma řešeného staticky neurčitého nosníku

Příklad 2.3. Stanovte průhyb v polovině rozpětí staticky určitého nosníku, jehož schéma je zobrazeno na obr. 2.3. Pro stanovení rovnice ohybové čáry ve zkoumaném intervalu $< 0; \frac{2}{3}\, l >$ využijte Clebschovu metodu.



Obr. 2.3 Statické schéma řešeného staticky určitého nosníku

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

2.1.1. Tabelace řešené funkce

Výpis hodnot funkce s parametrem x lze nejlépe provést s využitím cyklu `for` a funkcemi pro výpis na obrazovku v předepsaném formátu `disp` a `sprintf`.

Poznámka 2.4. Funkce `disp(x)` zobrazí obsah proměnné x typu `text`.

Poznámka 2.5. Funkce `sprintf` převede data do textového řetězce v požadovaném formátu pomocí tzv. „konverzních specifikátorů“, které začínají znakem %. Běžnou konverzi je možno provést specifikátorem `%f` pro převod numerické hodnoty do tvaru s pevnou desetinnou čárkou, specifikátorem `%e` pro vyjádření číselné hodnoty v exponenciálním tvaru, příp. specifikátorem `%g` pro automatickou volbu. Mezi znak `%` a specifikátor `f`, `e` nebo `g` lze navíc vložit počet znaků požadované šířky formátu, případně tečku a počet požadovaných znaků za desetinnou čárkou. Parametr `\n` zajistí přechod na nový řádek.

Příklad 2.6. Proveďte tabelizaci funkce ohybové čáry nosníku z příkladu 2.1. Výsledné průhyby určete v průřezech s roztečí 10 cm.

Řešení. Sled příkazů pro výpis výsledných průhybů ve sledovaných bodech x by mohl vypadat následovně:

```
format long
disp('  x [mm]    w(x) [mm]')
disp('_____________________')
for x=0:.1:1
disp(sprintf('%8.1f %12.4f',x*1000,horner(4,c,x)*1000))
end
```

Výpis pak bude mít následující vzhled:

```
  x  [mm]     w(x)  [mm]

---------------------
     0.0        0.0000
   100.0        1.5226
   200.0        3.0377
   300.0        4.5383
   400.0        6.0176
    ...          ...
  5700.0        0.6297
  5800.0        0.2881
  5900.0        0.0741
  6000.0        0.0000
```

▲

Příklad 2.7. Výpis z předchozího příkladu doplňte i o hodnotu posouvající síly V_z, ohybového momentu M_y a pootočení φ_y.

2.1.2. Vykreslení grafu řešené funkce

Graf řešené funkce lze vykreslit s využitím postupu, popsaném v kap. 1.4 předchozí přednášky.

Příklad 2.8. Vykreslete graf ohybové čáry nosníku z příkladu 2.1.

Řešení. Sled příkazů, s využitím cyklu **for**, odvozeného vektoru c z příkladu 2.1 a m-funkce `horner` může být následující:

```
x=linspace(0,1,100);
for i=1:100, y(i)=horner(4,c,x(i))*1000; end
plot(x,y,'b-');
title('Graf ohybové čáry w(x)');
xlabel('x');
ylabel('w(x)');
```

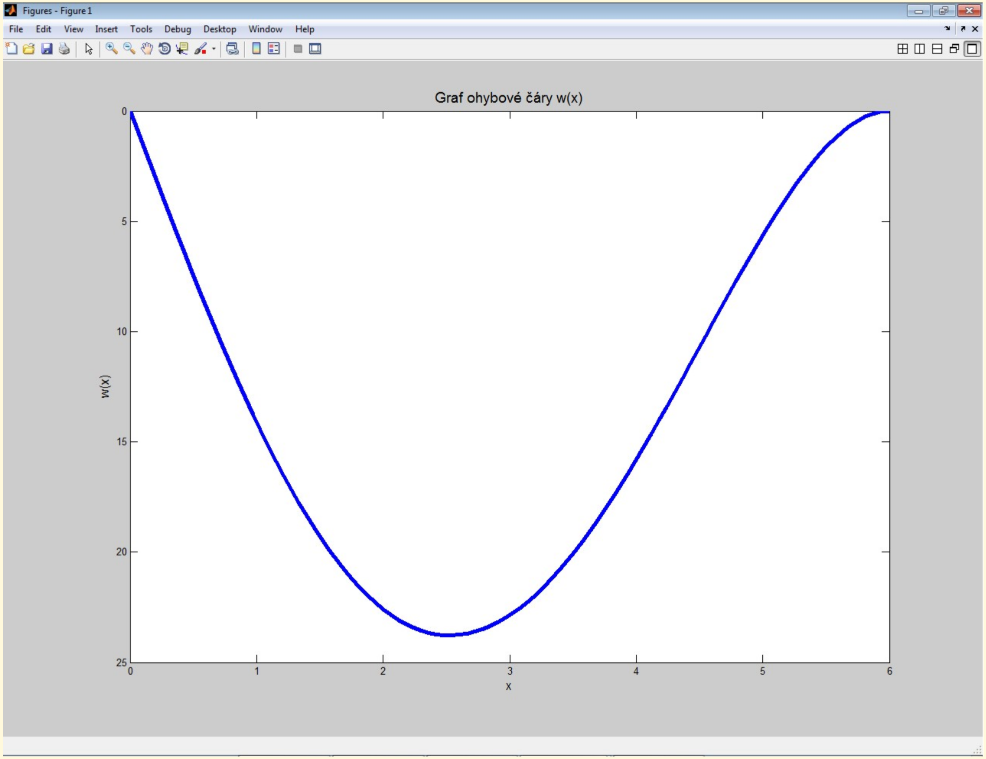
Výsledný graf ohybové čáry je pak zobrazen na obr. 2.4.

▲

Příklady k procvičení

1. Obdobně sestrojte graf ohybové čáry staticky neurčitém nosníku, který je schématicky zobrazen na obr. 2.2.

2. Vykreslete graf ohybové čáry staticky určitého nosníku, jehož schéma je zobrazeno na obr. 2.3.



Obr. 2.4 Ohybová čára staticky neurčitého nosníku z příkladu 2.1

2.1.3. Určení extrému diskretizované funkce

Zjednodušený výpočet největší hodnoty funkce v předem definovaném intervalu lze provést ve třech krocích nejprve diskretizací osy x, stanovením hodnot funkce pro všechna x v požadovaném rozsahu a algoritmem 2 pro stanovení největšího čísla ve vektoru.

Vstup : n, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$
Výstup: *maximum*

$maximum \leftarrow b_1$
for $i \leftarrow 2, 3, \ldots, n-1, n$ **do**
  if $b_i > maximum$ **then**
    $maximum \leftarrow b_i$
  end
end

Algoritmus 2: Algoritmus pro stanovení největšího čísla ve vektoru

Při programování uvedeného algoritmu v systému Matlab je možno vytvořit m-funkci `maximum` s následující posloupností příkazů:

```
function m=maximum(b)
n=length(b)
m=b(1);
if n>1
  for i=2:n
    if b(i)>m
      m=b(i);
```

```
    end
  end
end
```

Poznámka 2.9. Funkce `length` vrací rozměr vektoru, obsaženého ve vstupním parametru.

Příklad 2.10. Určete hodnotu největšího průhybu nosníku z příkladu 2.1 s využitím tabelizovaných hodnot ohybové čáry z příkladu 2.6.

Řešení. Nejprve je nutno vytvořit vektor $b = b_1, b_2, \ldots, b_n$ s hodnotami průhybu v sledovaných průřezech ($b_i = w_z(x_i)$) o souřadnicích x_i s roztečí 10 cm (n tedy bude při rozpětí $l = 6$ m rovno hodnotě 61):

```
i=0;
for x=0:.1:l
  i=i+1;
  b(i)=horner(4,c,x)*1000;
end
```

Nyní lze vyvolat funkci pro hledání největšího čísla ve vektoru b. Po zadání příkazu `maximum(b)` bude výpis m-funkce a výsledek největšího průhybu v mm následující:

```
n =
    61

ans =
   23.7654
```

▲

Poznámka 2.11. Tento způsob výpočtu největší hodnoty funkce je pouze přibližný, neboť je zatížen chybou danou diskretizací osy x. Způsob výpočtu, který povede k přesnému řešení, bude ukázán v následující kapitole.

Příklady k procvičení

1. S využitím m-funkce `maximum` určete i největší průhyb na staticky neurčitém nosníku, který je schématicky zobrazen na obr. 2.2.

2. Stanovte největší průhyb na staticky určitém nosníku, jehož schéma je zobrazeno na obr. 2.3.

Literatura

[1] Sauer T. *Numerical Analysis.* George Mason University. Pearson Education, Inc., 2006. (669 s). ISBN 0-321-26898-9. (Citováno na s 10.)

[2] *Wikipedia.* Otevřená internetová encyklopedie. Webové stránky. [on-line]. <http://cs.wikipedia.org>. (Citováno na s 4.)