Základy algoritmizace a programování

Přednáška 1 a 2

1. a 8.října 2018

Obsah

Úvodní informace

Olga Majlingová : ÚTM, Karlovo nám., budova D, 204,
olga@majling.eu http://olga.majling.eu

- web: http://marian.fsik.cvut.cz/zapg Informace k předmětu

(pro prezenční studium)

Podmínky udělení **klasifikovaného** zápočtu:
Bodované odevzdané příklady (prográmky+test):
maximální zisk **Z** bodů, student získal **Q** bodů.

- hodnocení **E** : 50% Z$\leq$ **Q** $<$ 60 % Z
- hodnocení **D** : 60% Z$\leq$ **Q** $<$ 70 % Z
- hodnocení **C** : 70% Z $\leq$ **Q** $<$ 80 % Z
- hodnocení **B** : 80% Z $\leq$ **Q** $<$ 90 % Z
- hodnocení **A** : 90% Z $\leq$ **Q** $<$ 100 % Z

Jak získat zápočet

Organizace předmětu: 5 úloh a jejich řešení v MATLABu.

1.10. Úloha první: Grafy a průsečíky

8.10. Úloha druhá: Plocha

15.10. Úloha třetí: Obyčejné diferenciální rovnice

22.10. Úloha čtvrtá:

29.10. Úloha pátá: Zpracování dat

Při účasti na méně než třech termínech zápočet udělen nebude (předmět nebyl absolvován).

5 domácích úloh:

- Odevzdání v zadaném termínu, vlastní vypracování 10 bodů
  bonusy a malusy
- bonusy: rozšíření zadání, originalita, pečlivost
- malusy: pozdní odevzdání, "podobnost"s dříve odevzdaným řešením
- Zlepšení hodnocení (pouze po odevzdání domácích úloh): zápočtový test ve zkouškovém období.

Cíle předmětu

NAUČIT SE
NAUČIT POČÍTAČ ŘEŠIT ÚLOHU,
KTEROU SAMI UMÍME ŘEŠIT

Počítačový program dělá jen to, co mu řeknete, nikdy však nedělá to, co byste chtěli, aby udělal.

- být schopen použít MATLAB pro řešení jednoduchých úloh
- být schopen číst a psát jednoduché programy
- rozumět základním konstrukcím a umět je použít

Název předmětu : Několik příkladů v MATLABu

Úlohy, které umíme řešit

- Najít "něco" v seznamu
  - **pojem "regulární" ve skriptech**
    Musíme pročítat skripta, slovo po slovu, dokud nenajdeme hledaný pojem.
  - **pojem "regulární" v rejstříku pojmů ve skriptech**
    Rejstřík pojmů bývá **uspořádaný** (podle abecedy).
    Nemusíme prohledávat pojmy od **a** do **z**, ale vyhledáme **r** (cca 18. písmeno z 26 písmen - hledáme ve druhé polovině),
    tj. mezi **n** a **z**, podíváme se do středu druhé poloviny (tam je **t**,
    tj. hledáme dál mezi **q** a **t**, ... najdeme podstatně rychleji.
- Čemu se přibližně rovná $\sqrt{2}$, když nemáme kalkulačku.
  Hledáme taková čísla $x_1, x_2 \in \mathbb{R}$, že platí: $x_1^2 \leq 2 \leq x_2^2$.
  Reálná čísla jsou **uspořádaná** (umíme určit zda $x_1 < x_2$).
  - $1 \leq 2 \leq 2^2$ - dál hledáme $\sqrt{2}$ mezi 1 a 1.5 nebo mezi 1.5 a 2?
  - $1 \leq 2 \leq 1.5^2$- dál ... $\sqrt{2} \in (1,\ 1.25)$ nebo $(1.25,\ 1.5)$?
  - $1.25 \leq 2 \leq 1.5^2$ ...
- Čemu se přibližně rovná $x \in \mathbb{R}$, pro které platí $f(x)=0$.
  Hledáme taková čísla $x_1, x_2 \in \mathbb{R}$, že platí:
  $f(x_1) \leq f(x) \leq f(x_2)$.

Řešení nelineární rovnice metodou půlení intervalu

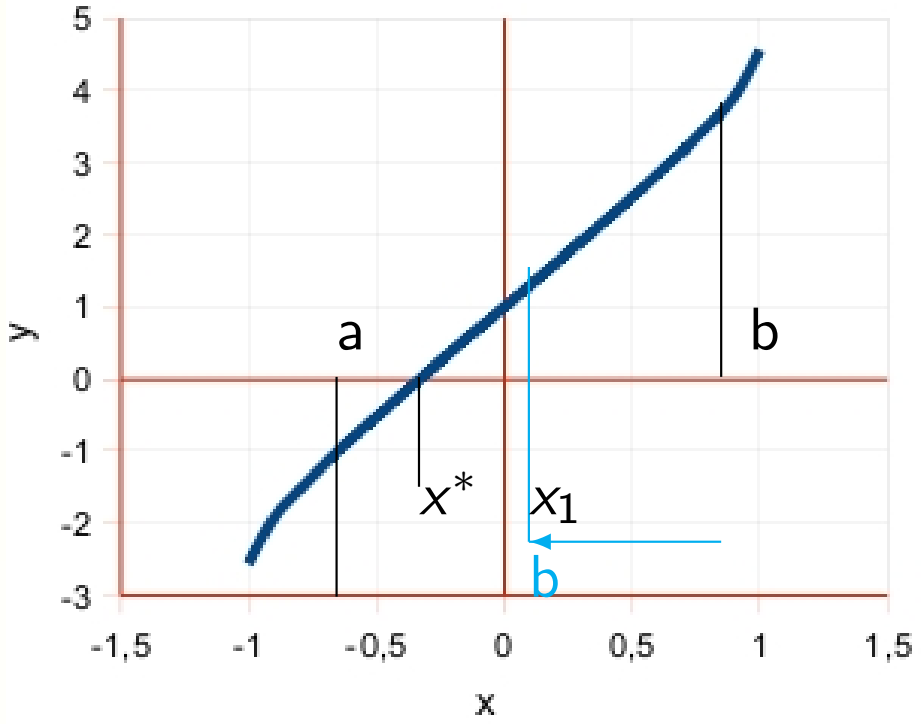
Předpokládáme, že

- v intervalu $< a, b >$ má rovnice $f(x) = 0$ **jediné** řešení (x^*)
- funkce $f(x)$ je spojitá v $< a, b >$
- v krajních bodech intervalu v platí: $f(a) \cdot f(b) < 0$

Určujeme přibližné řešení

$$x_k = \frac{a+b}{2}, \quad k = 1, 2, ...$$

Jestliže $f(x_k) = 0$, pak x_k je hledaným kořenem x^*.
Jinak hledáme v intervalu $< x_k, b >$ nebo $< a, x_k >$, takovém, že $f(a) \cdot f(b) < 0$



Jestliže $|b - a| < \varepsilon \Rightarrow x^* = \frac{a+b}{2}$

Přesnost aproximace

Po k krocích metody půlení najdeme přibližné řešení x_k, které se od přesného řešení x^* liší :

$$|x_k - x^*| < \frac{b-a}{2^k}$$

Pokud na zadaném intervalu **je jediný** kořen, cyklus **bude** ukončen, protože v každé iteraci se $|b-a|$ zmenší na polovinu, tedy po **konečném** počtu iterací bude $|b-a| < \varepsilon$

Jak určit interval?

- Z grafu.
- Z průběhu funkce.

Algoritmus

- Máme **dáno** : konkrétní funkci f(x), interval $\langle a, b \rangle$.
  1. Ověříme, že v bodech a, b mají funkční hodnoty **různá znaménka**. $f(a) \cdot f(b) < 0$
  2. Pokud to není pravda, **kořen nehledáme**.
- **Opakujeme, dokud** $|b - a| > \varepsilon$
  3. Vezmeme x uprostřed mezi a a b. $x = (a + b)/2$
  4. Zjistíme, kde je kořen:
     1. $|f(x)| < \varepsilon$ kořen je x, **končíme**
     jinak
     2. $f(x) \cdot f(a) < 0$ kořen je v intervalu $\langle a, x \rangle$ $b = x$
     jinak
     3. $(f(x) \cdot f(b) < 0)$ kořen je v intervalu $\langle x, b \rangle$ $a = x$
- Konec algoritmu. Pokud na intervalu **byl jediný** kořen , je v x.

Vykoušíme algoritmus

Dříve než začneme "učit počítač", musíme to umět sami!

Příklad. Rovnici

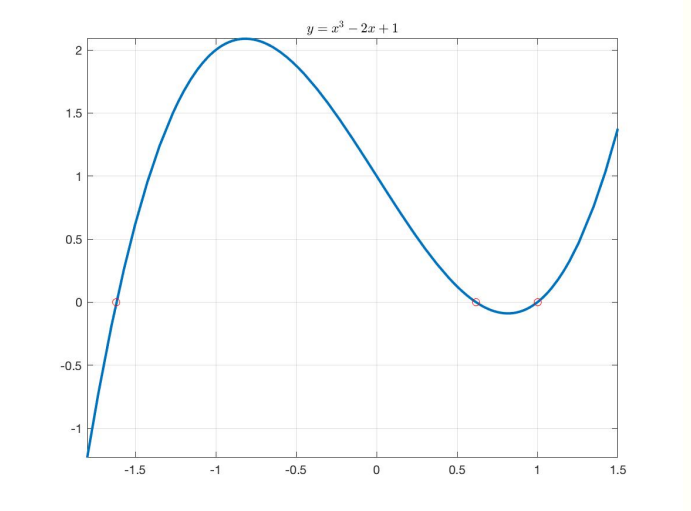
$x^3 - 2x + 1 = 0$ tj.

$(x - 1)(x^2 + x - 1) = 0$

umíme vyřešit,

$x_1 = 1$,

$x_2 = \frac{-1-\sqrt{5}}{2}, x_2 = \frac{-1+\sqrt{5}}{2}$



Postupujeme podle uvedeného algoritmu

1. Vybereme interval, např. $a = -2,\ b = -1$
2. Ověříme, zda $f(a) \cdot f(b) < 0 : f(-2) = -3, f(-1) = 2$, platí.

Opakujeme kroky 3 a 4 a mezivýsleky zapisujeme do tabulky:

				znaménko			měníme
k	a	b	x_k	$f(a)$	$f(b)$	$f(x)$	mez
1.	-2	-1	-1.5	-	+	+	b
2.	-2	-1.5	-1.75	-	+	-	a
3.	-1.5	-1.75	-1.625	-	+	-	a

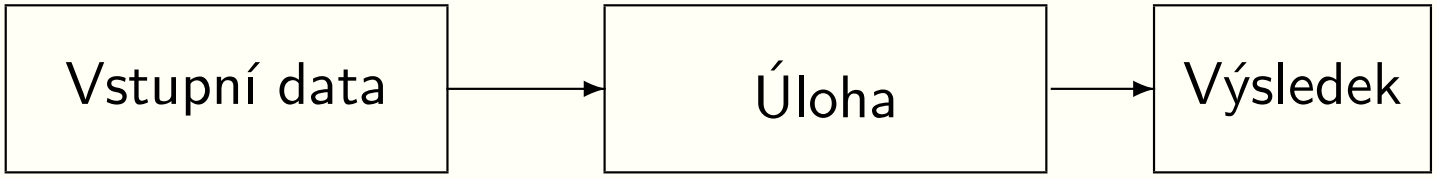
Po několika krocích máme pocit, že tomu rozumíme.

Algoritmy a programy

- Algoritmus . . .
  - . . . si počítač **sám** nevymyslí!
- Program . . . data+ příkazy
  - . . . data (co mám, co chci)
  - . . . příkazy: mění data a určují pořadí výpočtů
- Základní příkazy jsou:
  - přiřazení
  - větvení (podmíněný)
  - cyklus (opakování)
- V příkazech se vyskytují **proměnné**

  proměnné mají jméno, typ a hodnotu

Algoritmus



Úloha je zadána svými vstupními a výstupními daty.

Algoritmus chápeme jako přesný postup,
kterým lze vyřešit daný typ úlohy.

Algoritmus . . .

. . . za nás počítač nevymyslí!

Algoritmizace a programování

Přípravu programu obvykle dělíme na dvě etapy:

- **Algoritmizace**
  Musíme přemýšlet!
  JAK ze vstupních dat dostaneme výsledky?
  Postup řešení úlohy si představujeme jako
  změnu hodnot proměnných
  . . . postupně upřesňujeme
  . . . až se dostaneme k příkazům, kterým "rozumí" vykonavatel algoritmu
  . . . můžeme znázornit např. graficky, např. ve formě vývojového diagramu
- **Programování**
  Podle zápisu algoritmu sestavíme program v programovacím jazyku.

Data a příkazy

- DATA: proměnné a konstanty

  jméno, typ, hodnota

  Jméno konstanty, proměnné, struktury, souboru...

  i d e n t i f i k á t o r

  je posloupnost písmen a číslic, která **začíná písmenem.**

  písmena : a..z A..Z _ (znak podtržítka)

  (bez diakritiky), (NIC JINÉHO !)

  malá a velká písmena se rozlišují

- PŘÍKAZY

  co umí vykonavatel algoritmu

  mění hodnoty proměnných (přiřazení)

proměnná = hodnota_výrazu

- POŘADÍ PŘÍKAZŮ

  za sebou

  skoky (větvení, opakování)

Změnu hodnot proměnných provádíme přiřazovacím příkazem.
Podmíněný příkaz, přepínač a příkazy cyklu určují **pořadí příkazů**.

MATLAB

- stažení a instalace: `download.cvut.cz`,...
- VPN
- spuštění
- přikazové okno a editor
- nápověda (dokumentace)!!!

Co potřebujeme k řešení $f(x) = 0$

1. Zapsat funkci $f(x)$ (aritmetický výraz).
2. Zadat hodnoty proměnných a, b.
3. Vypočítat $f(a), f(b), x, f(x)$.
4. Zobrazit graf $y = f(x)$.
5. Realizovat (zapsat "matlabovsky") podmíněný říkaz, tj.:
   Jestli platí, že $f(x) \cdot f(a) < 0$, *tak zaměň b za x, jinak ...*
6. Realizovat opakování výpočtu, tj.
   Dokud platí, že ... , opakuj ....
7. Vypsat výsledek.

Aritmetické výrazy

Výrazy se skládají z jmen proměnných, čísel, operátorů , funkcí .

Operátory

- sčítání +
  odčítání -
- násobení *
- dělení /
  funkce `floor()`,
  `ceil()`
- umocňování ^
- kulaté závorky ( )

Matematické funkce

odmocnina: `sqrt(x)`

goniometrické: `sin(x); cos(x); tan(x);`

inverzní: `asin(x); acos(x); atan(x);`

logaritmy: `log(x); log10(x);`

e^x: `exp(x);`

absolutní hodnota: `abs(x)`

zbytek po dělení: `rem(x), mod(x)`

Vyčíslování: zleva doprava (zpravidla), obvyklá priorita :
závorky, funkce, mocnění, násobení/dělení, sčítání/odčítání

ZLOMKY! `a/b*c` nebo `a/(b*c)`!

Aritmetické výrazy v Matlabu

MATLAB: operátory

- transponování '
- :

  od **:** krok **:** do

  5 : 0.5 : 7

  5 : 1 : 7 je-li krok roven 1, lze vynechat 5 : 7

"tečkované" operátory

- násobení .∗
- dělení ./ .\
- umocňování (.^)

"tečkované" operace: **.*** , **./** , **.^** ... po prvcích

obyčejné : ***** , **/** , **** , **^** ... pravidla pro matice!

Zápis výrazů, přiřazovací příkaz : příklady

- Zapište v MATLABu:

$$\frac{a+\frac{b}{c}}{\frac{c}{d^2}} \qquad \frac{a+bc}{d^3+\sqrt{e}}$$

- Určete hodnotu výrazů zapsaných v MATLABu

```
a=1, b=2, c=3, d=4;
sqrt(d)/(a+b)*c +d;
```

```
e=1,f=2,g=3,h=4;
mod(g,f) * h + e / f * g * h;
```

- A=5
  B=3.14
  A=B
  B=A

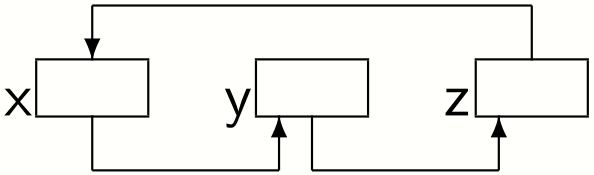
- A=5
  B=3.14
  B=A
  A=B

- A=5
  B=A
  B=B+1

- A=5
  A=B
  A=A-1

Jaké jsou hodnoty A,B v jednotlivých sloupcích?

- Zapište příkazy, pomocí kterých se hodnoty proměnných x,y,z změní tak, že v proměnné x bude hodnota z, v proměnné y původní hodnota x a v proměnné z původní hodnota y.



Zadání funkce, která počítá hodnotu výrazu

Výraz, do kterého chceme vícekrát dosazovat různé hodnoty **lze** zapsat jako tzv. řádkovou funkci (viz `Anonymous Functions`).
Taková funkce může obsahovat pouze jeden výraz.
(Více o funkcích bude v dalších přednáškách.)
Pravidlo pro zadání takové funkce je

```
jmeno_funkce @(parametry) vyraz
```

Například pro $y = x^3 - 2x + 1$ můžeme
pojmenovat funkci `f`, parametr `x`, zapsat:

```
f = @(x) x^3 - 2*x +1
```
nebo
```
f = @(x) x.^3 - 2.*x +1
```

a dále používat, např.: y=f(1.5) , (za `x` ve funkci se dosadí 1.5, vypočte se výraz a jeho hodnota se uloží do proměnné `y`);
nebo `a=-2; b=-1; f(a)*f(b)` (takto se hodnota vypočítaného výrazu nikam neuloží)

Zadání hodnot proměnným

Zobrazení grafu pomocí funkce plot

Funkce `plot` kreslí graf zadaný pomocí hodnot ve vektorech.
K tomu si otevře nové grafické okno (pokud žádné neexistovalo)
nebo začne kreslit do aktivního okna.

Syntaxe:

`plot(y)`	vykreslí hodnoty vektoru `y` v závislosti na jejich pořadí
`plot(x,y)`	hodnotách vektoru `x`
`plot(x,y,str)`	hodnotách vektoru `x` a pomocí řetězce `str` ovlivní výsledný vzhled grafu: barvu značky a čáry, typ značky a typ čáry

Proměnná str obsahuje až tři hodnoty vlastností, v pořadí barva,
typ značky a typ čáry. Implicitně je nastavena barva `b` (modrá),
značka žádná a plná čára. (více viz `help plot`)

Příklad - graf polynomu černou plnou čárou:

```
x=a:0.01:b; plot(x,f(x),'k')
```

Někdy je možné použít funkci `fplot`.

Někdy to, a někdy něco jiného

PODMÍNĚNÝ PŘÍKAZ

Někdy jedno, jindy druhé

Začátek

- zadej funkci (f), načti vstup (a,b)

je $f(a) \cdot f(b)$ záporné?

ano

- vypočítej $x = \frac{a+b}{2}$, $f(x)$

je $f(a) \cdot f(x) < 0$?

ano

$b = x$ (změníme b)

ne

$a = x$ (změníme a)

je $|b - a| < \varepsilon$?

ano

Konec, máme x

ne

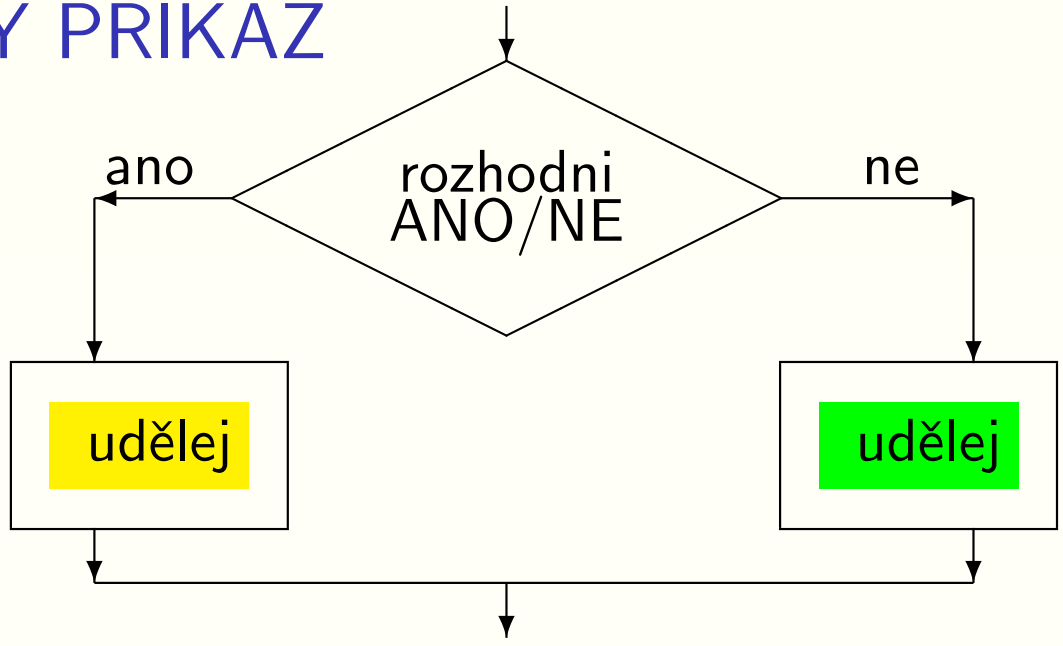
ještě není konec

ne

Na intervalu $\langle a,\ b \rangle$ graf funkce f osu x neprotíná, tj. rovnice $f(x) = 0$ nemá řešení

různé cesty, ale bez návratu "dozadu"

PODMÍNĚNÝ PŘÍKAZ



otestuje se **podmínka** :
pokud platí, výpočet pokračuje po cestě "ano",
pokud neplatí, výpočet jde cestou "ne";
konec rozvětvení výpočtu je označen **end**

MATLAB:

```
if podmínka
... cesta "ano"
else
... cesta "ne"
end
```

ZKRÁCENÁ VERZE:

```
if podmínka
... cesta "ano"
end
```

Logické výrazy

Podmínka – logický výraz : má hodnotu "pravda" nebo "lež"

Porovnání (relační operátory)

<	(menší)	>	(větší)
<=	(menší nebo rovno)	>=	(větší nebo rovno)
==	(ROVNOST)	~=	(nerovnost)

Logické operace

Konjunkce	&&
Disjunkce	||
Negace	~

Podmíněný příkaz : příklady

```
if f(a)*f(b) < 0
   x=(b-a)/2
   if f(a)*f(x) < 0
      b = x
   else
      a = x
   end
else
   disp('mezi a,b koren neni')
end
```

```
if f(a)*f(b) >= 0
 disp('mezi a,b koren neni')
else
   x=(b-a)/2
   if f(a)*f(x) > 0
      a = x
   else
      a = x
   end
end
```

Opakování realizujeme pomocí

C Y K L U

Příkazy cyklu

Otázky, které si klademe:

1. Co se bude opakovat(jaké příkazy)?
2. Čím se od sebe liší (co se v jednotlivých opakováních mění)?
3. Čím začneme a čím skončíme?

Ujasníme si, zda

- se příkazy budou opakovat určitý počet krát
  nebo
- činnost se bude opakovat, dokud bude platit nějaká podmínka

Příklady:
tisk tabulky funkčních hodnot
součet funkčních hodnot
"pohyby peněz na účtu"
výpočet hodnoty, dokud se poslední dva výsledky liší o více než ε

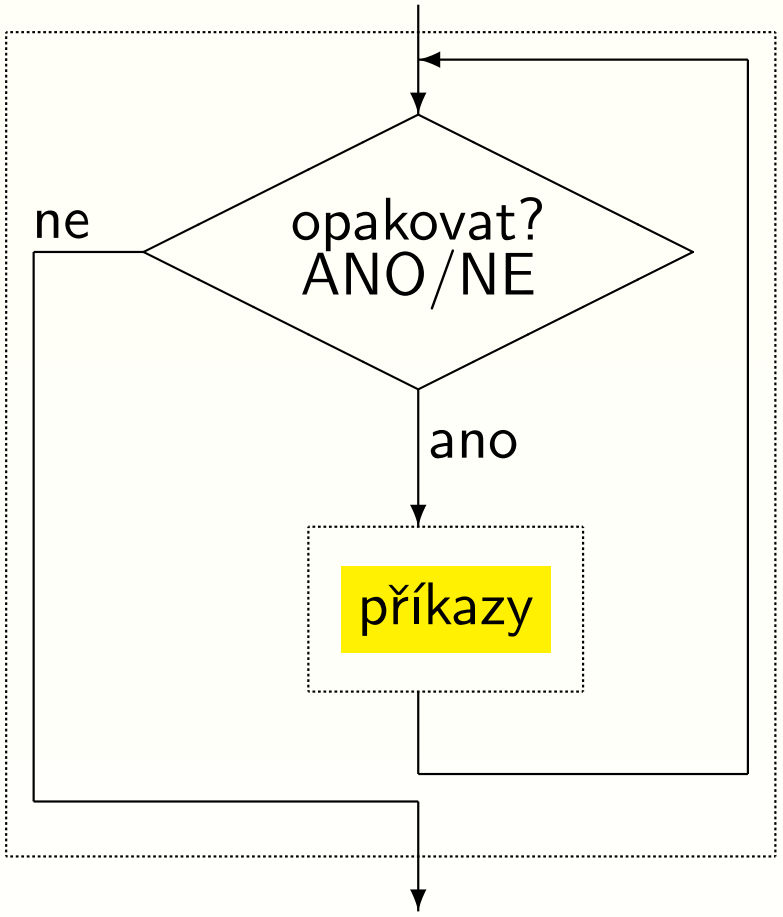
MATLAB

- operátor **:**
  `x=prvni : krok : posledni,`
  `k=a : b`
- **for** cyklus, počet opakování je znám předem
  for `k=1:n`
  to, co se opakuje
  end
- **while** cyklus, opakování závisí na platnosti podmínky
  while podmínka_trvání_cyklu
  to, co se opakuje
  end

Opakování výpočtu: dokud platí podmínka

Cyklus "while": opakování určuje podmínka

while podmínka_opakování
    to, co se opakuje
end



Vrstevnatost

Změnu hodnot proměnných provádíme přiřazovacím příkazem.
Podmíněný příkaz, přepínač a příkazy cyklu určují **pořadí příkazů**.

```
f = @(x) x.^3 - 2.*x +1
fplot(f)
a = input('Zadej dolní mez: ')
b = input('Zadej horní mez: ')
```

```
if f(a)*f(b) < 0
    while abs(b-a) > presnost
        x=(a+b)/2
        if f(a)*f(x) > 0
            a=x
        else
            b=x
        end
    end
end
```

Skoky ven

- Někdy nastane v průběhu cyklu situace, kdy je vhodné (nutné) ukončit opakování nehledě na platnost podmínky opakování.
  K tomu slouží příkaz break, který ukončí cyklus
  (skočí za slovo end, které odpovídá začátku cyklu),
  pro vložené cykly – ukončí vnitřní, vnější pokračuje.
- Někdy nastane v průběhu cyklu situace, kdy je vhodné (nutné) ukončit pouze aktuální iteraci cyklu.
  K tomu slouží příkaz continue, který ukončí aktuální iteraci a skočí na testování podmínky. Pro vložené cykly se také vztahuje pouze k vnitřnímu cyklu.
- Někdy je nutné dříve ukončit funkci. K tomu slouží příkaz return, který realizuje návrat tam, odkud byla funkce volána.
- Někdy je nutné dříve ukončit funkci nebo skript.
  Pro tento případ lze použít error(text).
- Příkaz, který se právě provádí, násilím ukončíme pomocí stisku CTRL+C. Většinou se CTRL+C používá, pokud jsme napsali nekonečný cyklus nebo jsme zapomněli na středník a necháváme vypisovat příliš velké matice.

Výpis výsledků

středník Některé příkazy vypisují své výsledky (např. `x = 215`).
Pokud nepotřebujeme tyto výsledky vidět, lze potlačit výpis výsledků příkazu napsáním středníku na konec příkazu (`x = 215;`).
Vypisování výsledků výpočtu by mělo být přehlednější než to, co vznikne pouhým nenapsáním středníku.

disp Nejméně užitečná je funkce disp, která se chová přibližně stejně, jako vynechání středníku.

fprintf Elegantní úpravu vytvoříme pomocí funkce fprintf (více - `help fprintf`).

printf Obrázky (grafy) lze uložit v různých grafických formátech pomocí funkce print (více - `help print`).

publish Skript (text programu) můžeme uložit v různých formátech pomocí funkce publish (více - `help publish`).

save Obsah jednotlivých proměnných lze uložit do souboru pomocí funkce save (více - `help save`)

Úloha první

Jsou zadány 3 funkce $y = f_1(x),\ y = f_2(x),\ y = f_3(x)$.

1. Do jednoho obrázku zobrazte grafy zadaných křivek.
   Můžete použít například funkci
   `fplot(funkce,[xmin,xmax,ymin,ymax])`, kde pomocí
   `[xmin,xmax,ymin,ymax]` lze zadat rozsah zobrazovaných hodnot.
   V některých případech je to vhodné použít, aby bylo hledanou část roviny lépe vidět.
2. Označte ty průsečíky zadaných křivek, které jsou vrcholy "křivočarého trojúhelníka", tvořeného grafy $f_1(x), f_2(x), f_3(x)$.
   Použijte označení: x_{12} pro průsečík f_1 a f_2, x_{13} pro průsečík f_1 a f_3, x_{23} pro průsečík f_2 a f_3.
3. Z grafů určete intervaly, ve kterých budete hledat průsečíky.
4. Určete průsečíky křivek **metodou půlení intervalu** s přesností $\varepsilon = 0.0001$.
   Průsečík křivek hledejte jako řešení rovnic
   $$f_1(x) - f_2(x) = 0,\ f_1(x) - f_3(x) = 0,\quad f_2(x) - f_3(x) = 0$$
5. Pokuste se určit průsečíky pomocí **solve()** a uveďte, zda se výpočet zdařil.

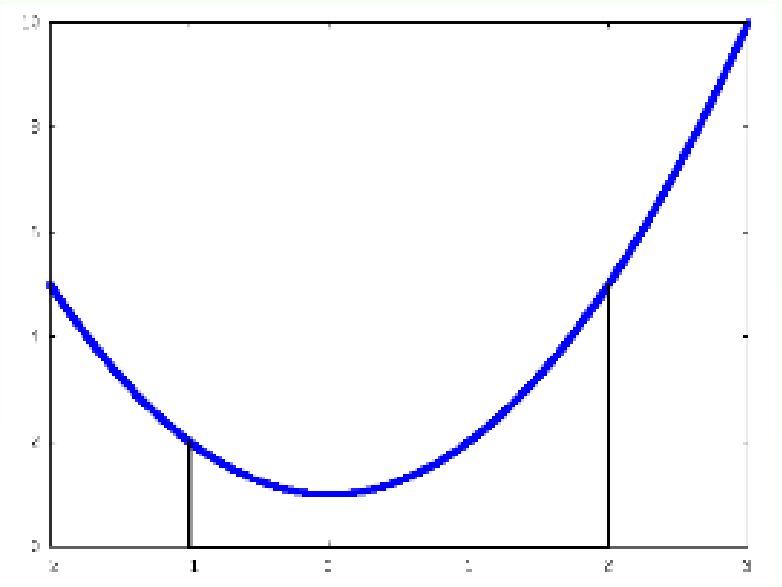
V Ý P O Č E T U R Č I T É H O I N T E G R Á L U

Výpočet určitého integrálu: plocha pod křivkou

$$\int_a^b f(x)dx \approx S$$

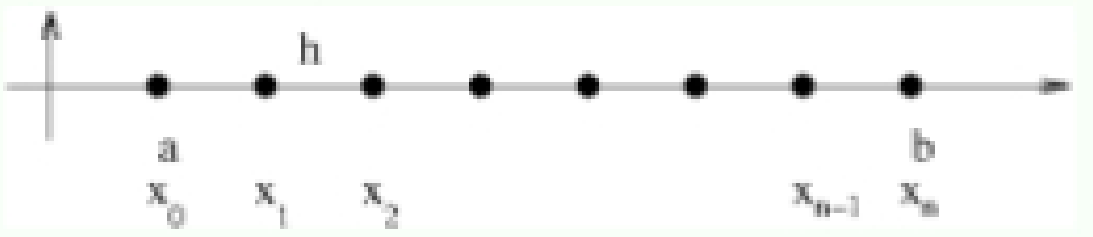
Funkce $f(x)$ je na intervalu $< a, b >$ spojitá a
$\forall x \in \langle a,\ b\rangle : f(x) \geq 0$.
Princip numerického výpočtu:

- interval $< a, b >$ rozdělíme na intervaly délky h.
- přesnou plochu nahradíme plochou obdélníka, lichoběžníka ,...
- plochy vypočteme a sečteme



Dělení intervalu

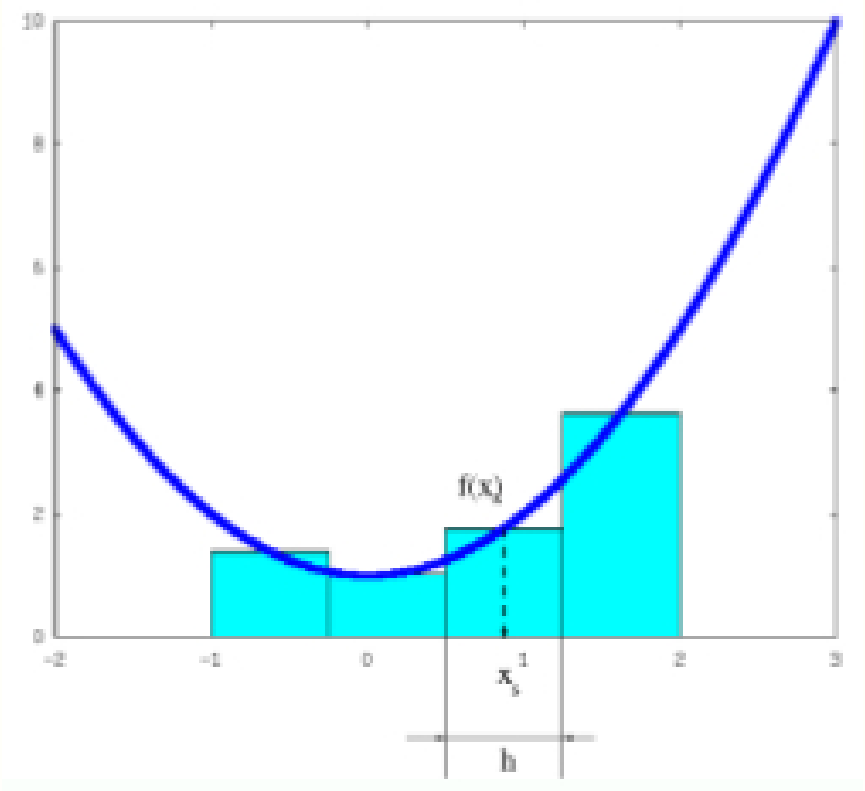
Integrujeme na intervalu $< a,\ b >$.
$x_0 = a,\ x_N = b$
Volíme **krok h** - vzdálenost mezi sousedními body x_i a x_{i+1}
nebo **počet bodů (N)**, na které chceme interval rozdělit



$$x_k = a + kh, \quad b = a + Nh$$

$$h = (b - a)/N$$

Obdélníková metoda

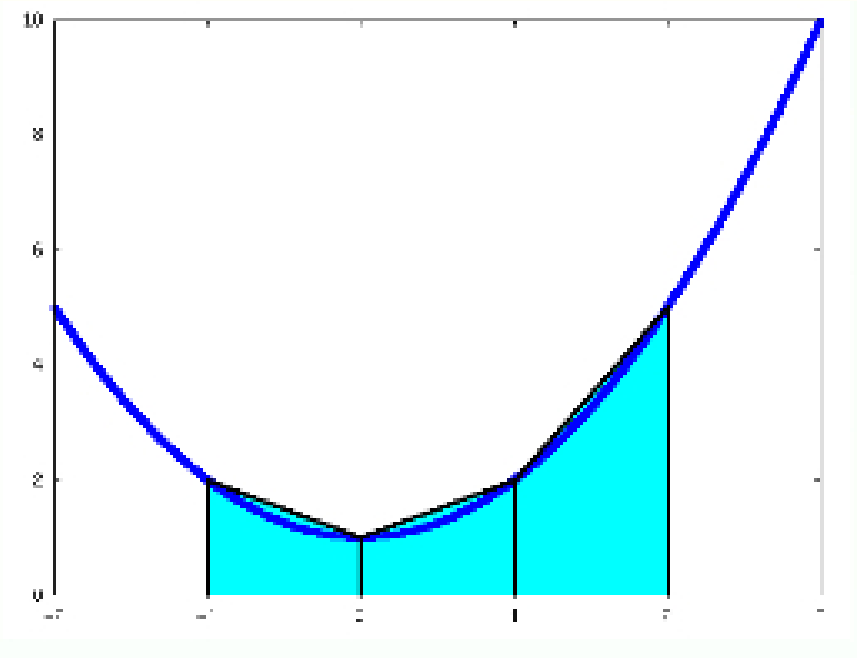


$$\int_a^b f(x)dx \approx S$$

$$S = \sum_{\text{všechny obdélníky}} S_k$$

$$S_k = h \cdot f(x_s)$$

Lichoběžníková metoda

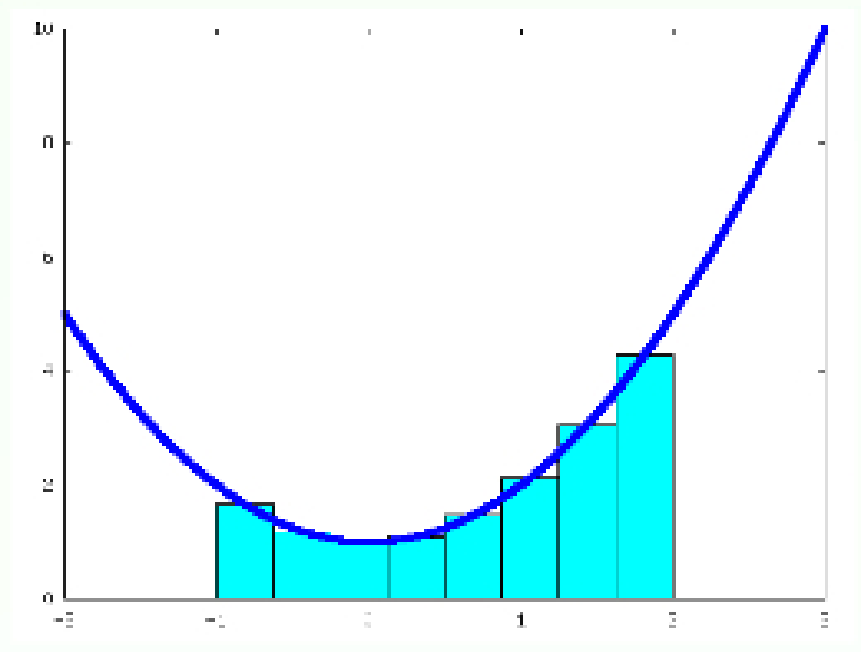


$$\int_a^b f(x)dx \approx S$$

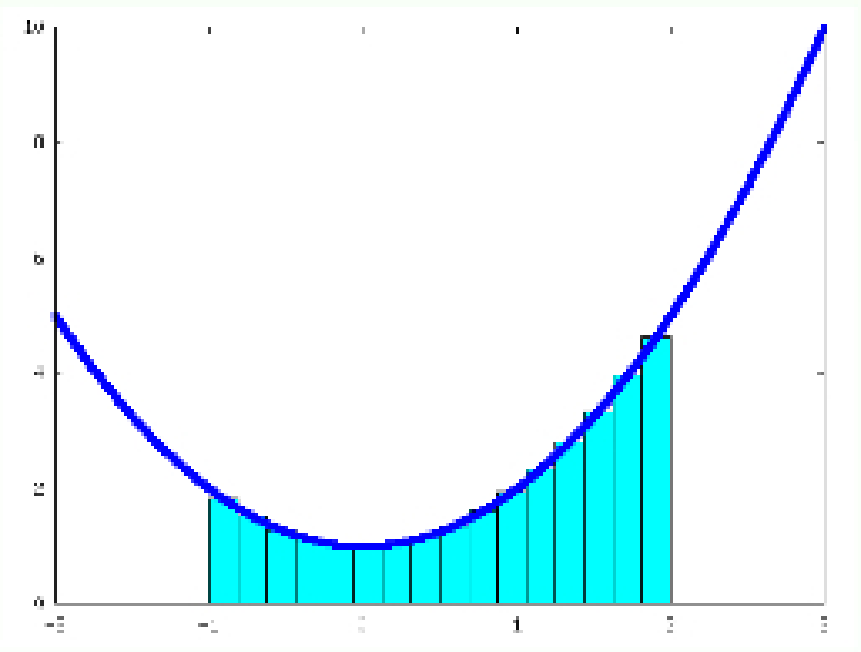
$$S = \sum_{\text{všechny lichoběžníky}} S_k$$

$$S_k = h \cdot \frac{f(x_i) + f(x_{i+1})}{2}$$

Dělení intervalu



N=8



N=16

Metody

- obdélníková metoda

$$\int_a^b f(x)\,dx \simeq h\cdot(f(a)+f(a+h)+\cdots+f(a+ih)\cdots+f(b))$$

$$h\sum_{i=0}^{n} f(a+ih)$$

- lichoběžníková metoda

$$\int_a^b f(x)\,dx \simeq h\cdot(\frac{1}{2}f(a)+f(a+h)+\cdots+f(a+ih)\cdots+f(b-h)+\frac{1}{2}f(b))$$

$$h\left(\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n-1} f(a+ih)\right)$$

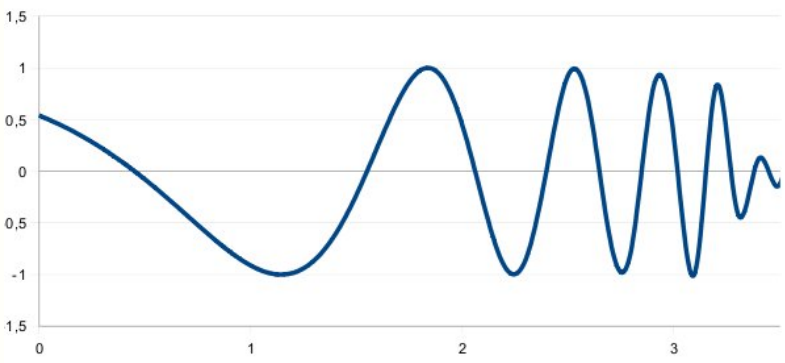
- Simpsonova metoda

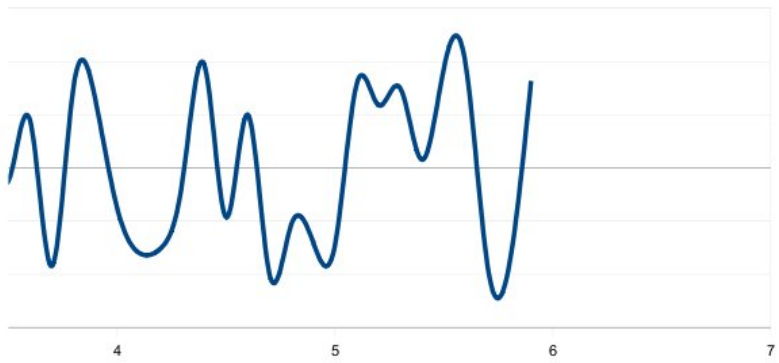
$$\int_a^b f(x)\,dx \simeq \frac{h}{3}\cdot(f(a)+4f(a+h)+2f(a+2h)+\cdots+4f(b-h)+f(b))$$

Kolik vnitřních bodů?

Jedna z možností:

- volím n=10
- počítám s_n a s_2n
- dokud $|s_{2n} - s_n| > \varepsilon$ "zpřesňuji", tj. zvětšujeme n





Algoritmus

SOUČET n hodnot...

- Máme **dáno** : konkrétní funkci f(x), interval $\langle a, b \rangle$, **metodu**.
  1. Volíme $n = 10$. (například)
  2. Určíme $h = \frac{b-a}{n}$
  3. Sečteme (podle metody) funkční hodnoty v bodech $x_0 = a, x_1 = x_0 + h, \ldots x_n = b$ a S_1
  4. Sečteme (podle metody) funkční hodnoty v bodech $x_0 = a + \frac{h}{2}, x_1 = x_0 + h, \ldots x_k = b - \frac{h}{2}$ a S_2
  5. $I_n = S1 \cdot h$, $I_{2n} = (S_1 + S_2) \cdot \frac{h}{2}$
  6. Jestli $|I_{2n} - I_n| < \varepsilon$ **končíme** výsledek je I_{2n} **jinak**
  7. Zvětšíme n a zmenšíme h: $n = n + n$ $h = \frac{h}{2}$.
  8. $S1 = S_1 + S_2$ (podle metody)
- **opakujeme kroky 4-8**

Výpočty v Matlabu

- Pokus o přesný výpočet (nutný balíček pro symbolické výpočty)

```
syms x
int(exp(x)*sin(x))
int(exp(x)*sin(x),0,1)
```

- Numerický výpočet:

```
trapz() %lichoběžníková metoda
quad()  %Simpsonova metoda
```

Úloha první: pokračování

1. Podle grafu sestrojeného v první úloze zapište, jak se určí výsledná plocha pomocí součtu/rozdílu určitých integrálů z $f_1(x), f_2(x), f_3(x)$ (v mezích určených v předchozí úloze jako průsečíky zadaných křivek), tj.

$$P = P_1 \pm P_2 \pm P_3, \quad \text{kde}$$

$$P_1 = \int_{x_{\dots}}^{x_{\dots}} f_1 \; dx, \; P_2 = \int_{x_{\dots}}^{x_{\dots}} f_2 \; dx, \; P_3 = \int_{x_{\dots}}^{x_{\dots}} f_3 \; dx.$$

2. Jednotlivé integrály určete obdélníkovou[1] metodou a výsledek porovnejte s výsledkem matlabovské funkce `integral`.
3. Určete výslednou plochu "křivočarého trojúhelníku", vytvořeného grafy třech (v minulé úloze zadaných) křivek.

[1] opraveno 11.10.

M–files

Program v MATLABu – posloupnost příkazů, které mohou být uloženy v souborech. Takovým souborům s instrukcemi říkáme **m–files**.

- **Program** je obyčejný textový soubor, který lze vyvtvořit v **textovém** editoru, můžeme použít editor MATLABu.
- **Příkazy** (instrukce) se zapisují stejně jako v příkazovém řádku, oddělují se znaky **; , ...** nebo **odřádkováním**
- **Konec programu** nemá žádný speciální znak, jednoduše poslední příkaz. Předčasné ukončení – příkaz **return**
- Komentář (poznámky) : od znaku % do konce řádku
- Komentáře na začátku souboru jsou dostupné z **help**u pomocí **help jméno_programu**

Typy **m**–souborů

Scénáře (skripty)

- Pracují s daty v pracovním prostředí (workspace).
- Nemají vstupní argumenty, nemohou vracet výsledky.
- Použití : odladění a orientační výpočty.

Funkce

- Pracují s vlastními – **lokálními** proměnnými nebo s globálními daty
- Mohou mít vstupní argumenty a mohou vracet výsledky
- Rozšiřují možnosti jazyka MATLAB

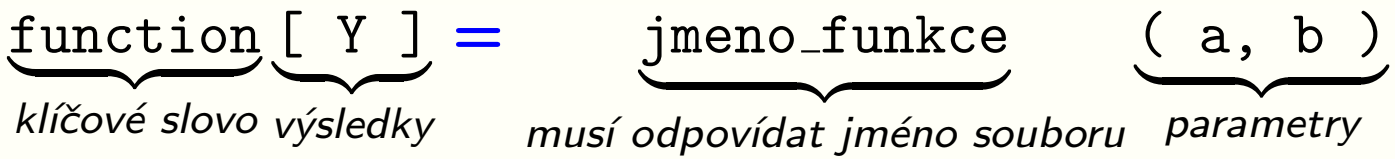
Funkce

- Hlavička funkce
  1. `function [y1,y2,...,ym] = fA(x1,x2,...,xn)`
  2. `function y = fB(x1,x2,...,xn)`
  3. `function fC(x1,x2,...,xn)`
  4. `function [y1,y2,...,ym] = fD`
- Tělo funkce

  Příkazy, posledním musí být **end**
- Volání funkce
  1. `[y1,y2,...,ym] = fA(x1,x2,...,xn)`
  2. `y = fB(x1,x2,...,xn)`
  3. `fC(x1,x2,...,xn)`
  4. `[ y1,y2,...,ym] = fD(x1,x2,...,xn)`
- Globální proměnné – `global a1 a2 ...` musí být uvedeno ve **všech** funkcích, které tyto proměnné používají
- Lokální funkce – podfunkce

  Soubor – funkce může obsahovat **několik** funkcí. **První** (hlavní) je dostupná zvně. Ostatní – lokální nebo podfunkce – mohou být volány pouze z hlavní funkce nebo z některé z lokálních funkcí v tomto souboru.

Hlavička funkce



Funkce musí mít jméno.

Může nemít výstupní hodnoty nebo parametry.

1. `function [y1,y2,y3,y4] = fA(x1,x2,x3)`
   Jméno funkce je fA. Odpovídající skript musí být v souboru `fA.m`
   Voláme se **třemi parametry**. Význam parametrů je (zpravidla) popsán v dokumentaci.
   Výsledky: funkce vrací **4** hodnoty, mohou být různých typů.
   při volání `Q=fA(x,y,z)`, bude `Q` obsahovat **první** výsledek, odpovídající `y1`.
   Pokud nepotřebujeme některé návratové hodnoty, lze nepotřebné výstupní hodnoty označit `~` , zapsat `[a,b,~,c] = fA(x,y,z)`.

   Pokud potřebujeme všechny výsledné hodnoty, voláme `[a,b,c,d]=fA(x,y,z)`.

Příklad

v souboru fa.m

```
function [y1,y2,y3,y4]=fa(x1,x2,x3)
y1=x1+x2;
y2=x1*x2;
y3=x2+x3;
y4=x2*x3;
end
```

odjinud voláme:

```
>>fa(1,2,3)
ans = 3
>>q=fa(1,2,3)
q = 3
>>[a,~,~,c]=fa(1,2,3)
a=3
c=6
>>[q1,q2,q3,q4]=fa(1,2,3)
q1=3
q2=2
q3=5
q4=4
```

Hlavička : výsledky

- function y = fB(x1,x2,x3)
  Funkce vrací jeden výsledek, může to být vektor, matice, ...
  Jednotlivé složky nelze vynechat.

Příklad

v souboru fb.m

```
function y=fb(x1,x2,x3)
y(1)=x1+x2;
y(2)=x1*x2;
y(3)=x2+x3;
y(4)=x2*x3;
end
```

odjinud voláme:

```
>>fb(1,2,3)
ans =
     3     2     5     6
>>q=fb(1,2,3)
q =
3     2     5     6
```

Hlavička: proměnný počet parametrů

Pro proměnný počet vstupních parametrů:
`doc varargin`

Pro proměnný počet výsledných parametrů:
`doc varargout`

Příklad

hledání průsečíků jako funkce:

```
function x = prusecik(f, a, b, epsilon)



end
```

Příklad

výpočet určitého integrálu jako funkce:

```
function x = plocha(f, a, b, epsilon)



end
```