Středoškolská technika 2010

Setkání a prezentace prací středoškolských studentů na ČVUT

VYBRANÉ NUMERICKÉ ALGORITMY

Ondřej Krupa

Vyšší odborná škola a Střední průmyslová škola

Mariánská 1100, Varnsdorf

Anotace:

Mým úkolem je popsat vybrané numerické algoritmy a k nim vytvořit numerické programy (algoritmy) práce a k nim ověřené programy (algoritmy). Programy jsem tvořil v MATLABu, což je název programu a zároveň zkratka, která znamená matematická laboratoř. Práce je rozdělená do pěti témat: Numerické derivace, Numerické integrace, Aplikace numerických derivací a integrací ve 2D, Postupná aproximace a Fourierův rozvoj funkce. Hlavní myšlenkou práce je její srozumitelnost, naučení a pochopení, protože mají složit hlavně studentům. Pomocí těchto prací by se měl laik naučit základní operace v MATLABu, proto jsem vše důkladně popsal.

Klíčová slova:

MATLAB, numerické algoritmy, Fourierův rozvoj, numerické derivace, numerické integrace, postupná aproximace

Obsah

Úvod

Název mé práce je Vybrané numerické algoritmy. Práce spočívá ve vytvoření algoritmů, přesněji pěti numerických algoritmů a k nim zhotovit ověřené programy. U vytváření algoritmu jsem použil zajímavý a propracovaný program MATLAB.

Práce se větví do pěti podobných témat. První z nich je Numerická derivace, kde jsem vysvětlil pojem derivace, dále ukázal jejich obecný výpočet a převedení vzorců pro výpočet do numerické podoby, se kterou umí pracovat výpočetní technika, tedy program MATLAB. Dalším cvičením jsou Numerické integrace, zde jsem objasnil pojem integrace, předvedl obecný výpočet integrálu do numerické podoby a použil tento výpočet v aplikaci MATLAB. Třetí laboratorní práce spočívá v Aplikaci numerických derivaci a numerických integraci ve 2D, kde jsem použil předešlá dvě témata a aplikovat je v rovině. Zde jsem vytvořil a popsal programy, které počítají plochu, objem tělesa, délku křivky, střední hodnotu, efektivní, moment setrvačnosti a těžiště. V další úloze Řešení nelineárních rovnic metodou postupné aproximace jsem vysvětlil, co znamená postupná aproximace, k čemu se používá a ukázal, jak ji využít k počítání nelineárních rovnic. Poslední cvičení je Fourierův rozvoj funkce, kde jsem popsal co to Fourierův rozvoj je, jak se obecně počítá a jak ho použít v MATLABu. U většiny programů jsem pro příklad zužitkoval základní matematické funkce, u kterých jde provést rychlá kontrola. Všechny algoritmy jsou zapsány univerzálně, takže jdou použít pro libovolný signál, rovnici…

Práce mají sloužit hlavně studentům, k nastudování a pochopení daného problém. Proto největší prioritu dostala srozumitelnost, snadné pochopení a nenáročnost, protože když o nějakém neurčitém tématu slyšíte poprvé, tak je velice obtížné ho pochopit. Pomocí těchto prací se laik může naučit základní operace v MATLABu.

K tématu jsem se dostal ve škole v předmětu Modelování, kde se učíme programování v MATLABu a některé numerické algoritmy. Předmět mě vyučuje můj konzultant Ing. Petr Bannert, který mi usnadnil pochopení a zpracováni daných témat. Patří mu mé poděkování.

1 Numerické derivace

Obsah

1. Generace funkce sinus.
2. Program v MATLABu, který derivuje libovolnou funkce.
3. Rozšíření o výpočet odchylky numerické metody od analytické.

1.1 Derivace v analytice

V matematice s k výpočtu derivací používají obvykle algebraické výrazy, které byly odvozeny na základě platných matematických pravidel. Ty se ovšem dají použít pouze v případě, že je známo algebraické vyjádření funkce, které se pak derivuje podle určitých pravidel. V případech, kdy známe pouze několik hodnot funkce bez znalosti jejího algebraického vyjádření, je třeba použít numerické metody derivace.

1.2 Numerické derivace

Druhy numerických derivací:

a) **Dopředná,** která používá body následující po bodu, pro který se počítá.

b) **Centrální (středová),** která používá stejný počet bodů před bodem a po bodu, pro který se počítá.

c) **Zpětná**, která používá body předchozí před bodem, pro který se počítá.

Vlastnosti numerických derivací:

a) **Centrální** derivace je nejužívanější a také nejpřesnější pro numerický odhad hodnoty derivace. Používá se u mezilehlých bodů derivované funkce (závislosti).

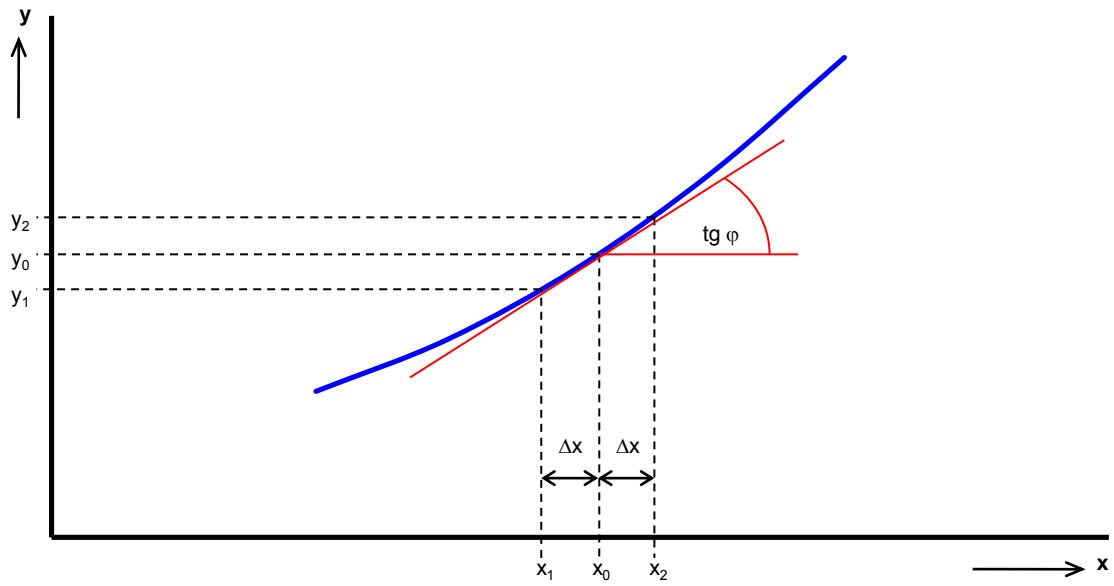
b) **Dopředná** a **zpětná** derivace jsou méně přesnější než centrální. V některých případech je však lepší použít tyto než centrální, např. v tom případě, kdy se odhaduje hodnota v krajním bodě dat nebo v případě, že se pokoušíme odhadnout hodnotu derivace v oblasti, ve které probíhají prudké změny hodnot. Při přibližování se k prudké změně je pak výhodnější použít zpětnou derivaci, při jejím překonání pak derivaci dopřednou.

Při používání numerických derivací samozřejmě vznikají chyby:

1. **Ořezávací chyba** – vzniká při odhadu hodnot derivace z několika málo diskrétních bodů. U těchto derivací závisí na vzdálenosti těchto bodů (diference).

2. **Zaokrouhlovací chyba** – vzniká v důsledku toho, že počítač pracuje s pevným počtem číslic.

Obecně platí, že při snižování ořezávací chyby se zvyšuje chyba zaokrouhlovací, proto je nutné při vypočtu derivace a návrhu diference (Δx) najít vhodný střed.



Obr. 1. 1 – Princip numerické derivace 1. řádu

Kde Δx …… je vzdálenost mezi dvěma vzorky (vzorkovací perioda)

Odvození numerických derivací 1. řádu:

Při výpočtu numerické derivace 1. řádu vyházíme z obr. 1. 1, který vystihuje princip derivace.

Dopředná numerická derivace 1. řádu v bodě x_0 je dána vztahem:

$$y'(x_0) = tg\varphi \cong \frac{y_2 - y_0}{x_2 - x_0} = \frac{y_2 - y_0}{\Delta x} \qquad (1.1)$$

Zpětná numerická derivace 1. řádu v bodě x_0 je dána vztahem:

$$y'(x_0) = tg\varphi \cong \frac{y_0 - y_1}{x_0 - x_1} = \frac{y_0 - y_1}{\Delta x} \qquad (1.2)$$

Centrální numerická derivace 1. řádu v bodě x_0 je pak dána aritmetickým průměrem součtu dopředné a zpětné pro který platí:

$$y'(x_0) = tg\varphi \cong \frac{1}{2}\left(\frac{y_2 - y_0}{x_2 - x_0} + \frac{y_0 - y_1}{x_0 - x_1}\right) = \frac{1}{2}\left(\frac{y_2 - y_0}{\Delta x} + \frac{y_0 - y_1}{\Delta x}\right) = \frac{1}{2}\left(\frac{y_2 - y_1}{\Delta x}\right) \qquad (1.3)$$

1.3 Derivace v MATLABu

Než začneme funkci sinus derivovat, musíme si nejdříve vygenerovat její průběh. To provedeme tak, že si nejdřív určíme interval, kterým bude křivka probíhat <*a*;*b*>, a počet vzorků N. Doporučuji průběh křivky nastavit na interval <0;π> a počet vzorku N=256. Potom si vypočteme vzorkovací periodu (krok), která je potřeba k získání souřadnic *x*.

Vztah pro výpočet vzorkovací periody:

$$\Delta x = \frac{b-a}{N} \qquad (1.4)$$

Po získání *x*-ových souřadnic vygenerujeme funkci. V MATLABu se generování zapíše tímto způsobem:

```
clc
clear
N=256;
a=0;
b=10;
krok=(b-a)/N;
x=a:krok:b-krok;
y=sin(x);
```

Ve výpisu z programu je jasné, že když za *y* dosadíme libovolnou funkci, program vytvoří její průběh.

Pro derivování je potřeba použít cyklus *for*, kterého vložíme podmínku *if* pro derivaci dopřednou, centrální a zpětnou. Pro výpočet derivace prvního vzorku využijeme derivace dopředné, pro získání derivace posledního vzorku zpětné derivace a pro všechny ostatní derivace centrální. Cyklus *for* je nutno využít z toho důvodu, že jinak bychom nemohli měnit vzorkovací periodu a musela by probíhat pro pevně určený počet vzorku N. Podmínku *if* zapíšeme tak, aby u prvního vzorku proběhla derivace dopředná, u posledního derivace zpětná a u všech ostatních centrální derivace.

Použití cyklu for, do kterého je vložena podmínka if, pro výpočet derivace:

```
%************************************************************
% vypocet derivace
%------------------------------------------------------------
for I=1:N,
    if I == 1
         dery(I)=(y(I+1)-y(I))/krok;           % dopředná
    elseif I == N
         dery(I)=(y(I)-y(I-1))/krok;           % zpětná
       else
         dery(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
       end
end
```

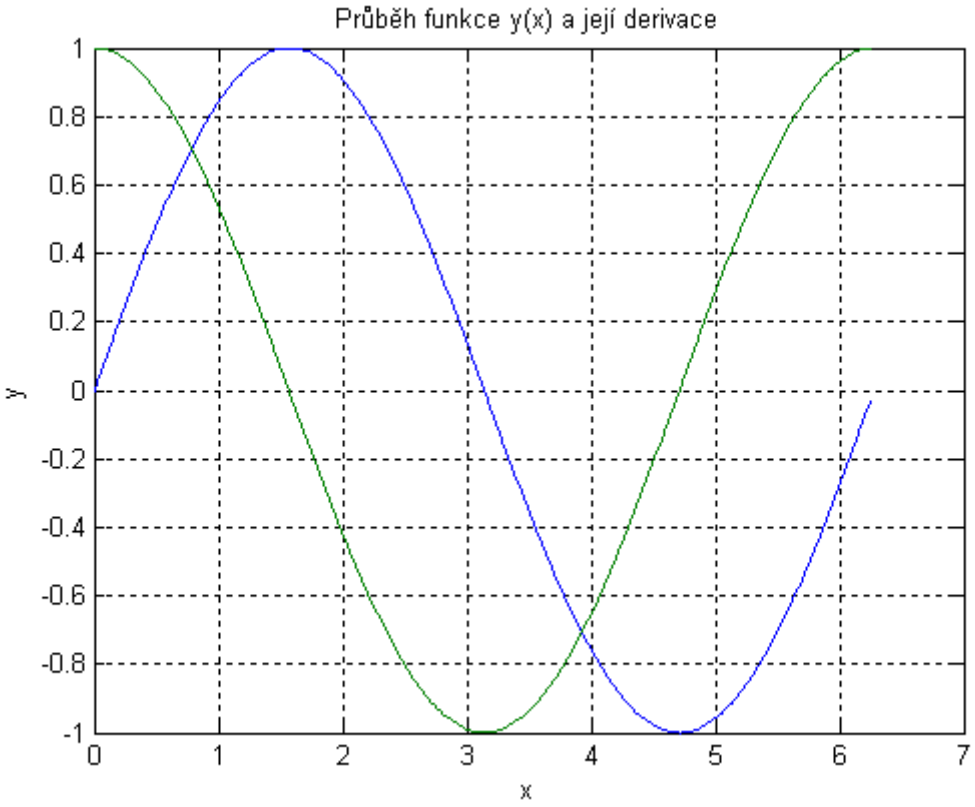
Dále si vykreslíme průběh funkce *y(x)* a její derivace do grafu. V našem případě jsme použili funkci sinus, pro kterou najdeme v tabulkách (MFCHT) její derivaci jako funkci cosinus.

$(\sin x)' = (\cos x)$	(1.5)

Díky tomu si můžeme zkontrolovat, jestli jsme při výpočtu neudělali chybu.

Výpisu průběhu funkce y(x) a průběh její derivace:

```
subplot 211
plot(x,y,x,dery)
grid;
title('Průběh funkce y(x) a její derivace');
xlabel('x');
ylabel('y');
```



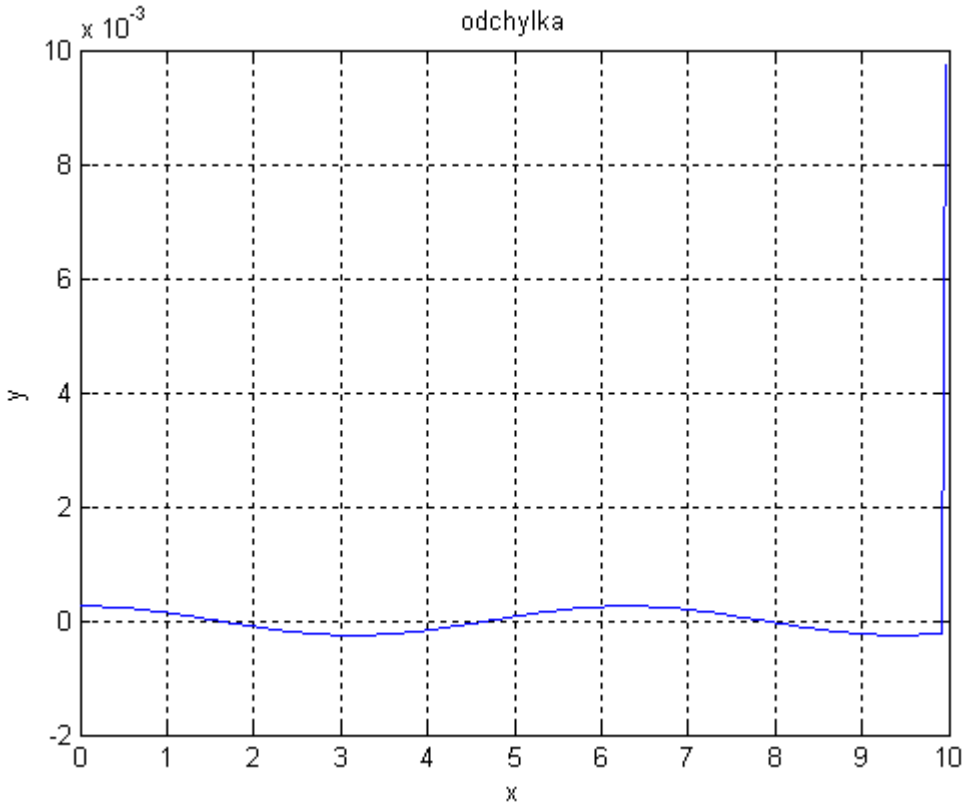
Obr. 1. 2 – Průběh funkce y(x) a její derivace

1.4 Vypočet odchylky

Výpočet odchylky provedeme jednoduchým způsobem. Už víme, že pro funkci sinus je její derivace funkce cosinus. Tak stačí vygenerovat průběh funkce cosinus, což je vlastně derivovaná funkce sinus, a od toho odečíst průběh derivované funkce sinus.

Výpočet odchylky a její výpis do grafu:

```
dera=cos(x);
Od=dera-dery;
subplot 212
plot(x,Od)
grid;
title('odchylka');
xlabel('x');
ylabel('y');
```



Obr. 1. 3 – Odchylka numerické derivace od analytické derivace

1.5 Výpis celého programu

```
clc
clear
N=256;
a=0;
b=10;
```

```
krok=(b-a)/N;
x=a:krok:b-krok;
y=sin(x);
for I=1:N,
    if I == 1
         dery(I)=(y(I+1)-y(I))/krok;           % dopředná    elseif
I == N
         dery(I)=(y(I)-y(I-1))/krok;           % zpětná
       else
         dery(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
       end
end
subplot 211
plot(x,y,x,dery)
grid;
title('Průběh funkce y(x) a její derivace');
xlabel('x');
ylabel('y');
dera=cos(x);
Od=dera-dery;
subplot 212
plot(x,Od)
grid;
title('odchylka');
xlabel('x');
ylabel('y');
```

2 Numerické integrace

Obsah

1. Naprogramování v MATLABu výpočet integrace:
   a) obdélníkovou metodou
   b) lichoběžníkovou metodou
2. Rozšíření programu o výpočet:
   a) odchylky numerické integrace od analytické pro obě metody
   b) odchylky numerické integrace od analytické pro obě metody v procentech
   c) chyby obou metod

2.1 Integrál

Integrál je pojem z matematiky, konkrétně integrálního počtu. Integrál zobecňuje pojmy jako plocha, objem, součet či suma. Integrování je opačnou operací k derivování.

Velké množství určitých integrálů nelze vyjádřit prostřednictvím elementárních funkcí, v těchto případech používáme k určení hodnoty integrálu přibližných metod, mezi něž patří tzv. **numerická integrace.** Při numerické integraci se snažíme nahradit integrál jiným druhem výpočtu, při kterém se snažíme zajistit, aby se získaná hodnota od skutečné hodnoty integrálu lišila co nejméně. Numerické metody výpočtu integrálu jsou velmi vhodné pro použití ve výpočetní technice, neboť umožňují vytvoření relativně jednoduchých algoritmů pro určování hodnot určitých integrálů. Výběr správné metody závisí na požadované přesnosti řešení, časové náročnosti a dalších parametrech.

Neurčitým integrálem obecně rozumíme výpočet průběhu funkce *f(x)*. A vypočteme ho ze vztahu:

Neurčitý integrál se používá k určení maxima a minima křivky. Určitým integrálem rozumíme

$$\int f(x)dx = F(x) + c \qquad (2.1)$$

výpočet plochy pod křivkou.

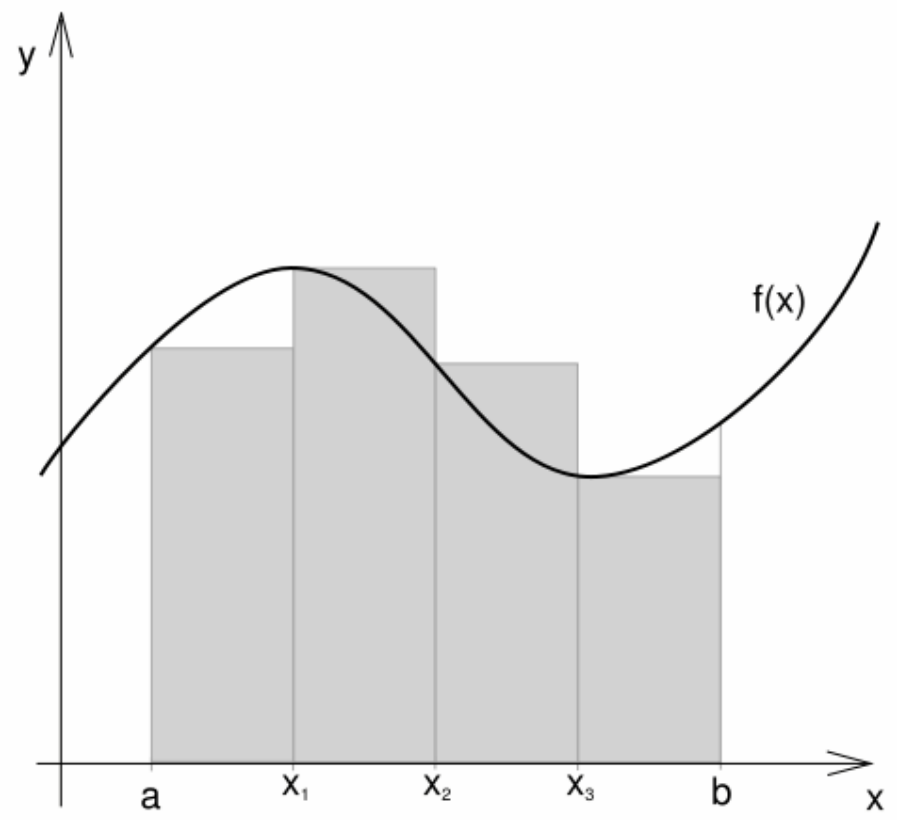
Existují tři základní metody numerické integrace.

d) obdélníková metoda
e) lichoběžníková metoda
f) Simpsonova metoda

U obdélníkové metody je vypočet plochy pod křivkou *f(x)* nahrazen výpočtem plochy pod obdélníkem, u metody lichoběžníkové je výpočet nahrazen výpočtem plochy pod lichoběžníkem a u Simpsonovy metody je vypočet zaměněn za výpočet plochy pod parabolou.

Nepřesnost u jakékoliv metody můžeme snížit zmenšením vzorkovací periody (kroku). Před začátkem výpočtu je dobré si určit, jakou přesnost potřebujeme. Určení přesnosti je hlavně důležité u programů, které počítají mnohem složitější výpočty, a zbytečně velká vzorkovací perioda velice ovlivní výpočetní čas. Přesnost (nepřesnost, toleranci) označujeme jako ε.

2.2 Obdélníková metoda



Obr. 2. 4 – Obdélníková metoda

kde a …… je začátek intervalu, od kterého počítáme

b …… je konec intervalu, do kterého počítáme

Přičemž platí:

$$a = x_0 < x_1 < x_2 < ... < x_N = b \qquad (2.2)$$

Vztah pro výpočet chyby obdélníkové metody:

$$R_N = |I - I_r| \le \frac{D_0 (b-a)^2}{N^2} \le \varepsilon \qquad (2.3)$$

Kde D_0 …… je maximální hodnota derivované funkce *f(x)*

R_N ……. je chyba výpočtu numerické integrace

I …… je integrál získaný analytickou metodou

I_r …… je integrál získaný numerickou metodou

Vztah pro získání D_0:

$$D_0 = \max_{a \le x \le b} |f'(x)| \qquad (2.4)$$

Počet kroků (vzorkovací periodu) určíme ze vztahu:

$$N \ge \sqrt{\frac{D_0 (b-a)^2}{\varepsilon}} \qquad (2.5)$$

Po převodu vztahu (2.1) do numerického tvaru pro obdélníkovou metodu, se kterým můžeme počítat v MATLABu, dostáváme:

$$F(x) + c = \sum_{N=1}^{i} f(x_N) \cdot \Delta x \qquad (2.6)$$

Určitý integrál v analytice vypočteme tímto vztahem:

$$\int_a^b f(x)dx = \int_a^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{N-2}}^{x_{N-1}} f(x)dx + \int_{x_{N-1}}^{b} f(x)dx \qquad (2.7)$$

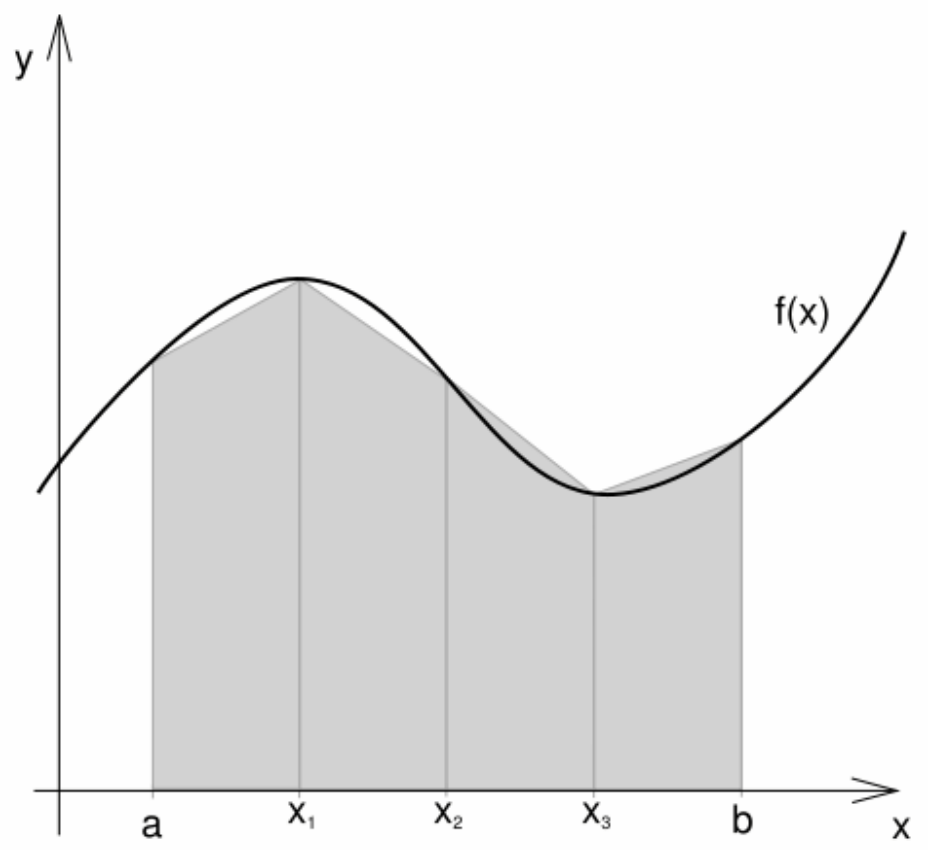
Po úpravě dostáváme:

$$\int_a^b f(x)dx \cong f(x_0)\Delta x + f(x_1)\Delta x + \ldots + f(x_{N-2})\Delta x + f(x_{N-1})\Delta x \qquad (2.8)$$

Po převodu z analytického tvaru do numerického:

$$I = \frac{b-a}{N} \sum_{i=1}^{N} f(x_{i-1}) \qquad (2.9)$$

2.3 Lichoběžníková metoda



Obr. 2. 5 – Lichoběžníková metoda

Vztah pro výpočet chyby lichoběžníkové metody:

$$R_N = |I - I_r| \leq \frac{D_l (b-a)^3}{12N^3} \leq \varepsilon \qquad (2.10)$$

Vztah pro získání D_l:

$$D_l = \max_{a \leq x \leq b} |f''(x)| \qquad (2.11)$$

Počet kroků (vzorkovací periodu) určíme ze vztahu:

$$N \geq \sqrt[3]{\frac{D_l (b-a)^3}{12\varepsilon}} \qquad (2.12)$$

Po převodu vztahu (2. x) do numerického tvaru pro lichoběžníkovou metodu, se kterým můžeme počítat v MATLABu, dostáváme:

$$F(x)+c=\sum\left[f(x_{N-1})+f(x_N)\right]\frac{\Delta x}{2} \qquad (2.13)$$

Určitý integrál v analytice vypočteme tímto vztahem:

$$\int_a^b f(x)dx=\int_a^{x_1} f(x)dx+\int_{x_1}^{x_2} f(x)dx+\ldots+\int_{x_{N-2}}^{x_{N-1}} f(x)dx+\int_{x_{N-1}}^{b} f(x)dx \qquad (2.14)$$

Po úpravě dostáváme:

$$\int_a^b f(x)dx\cong\frac{f(x_0)+f(x_1)}{2\Delta x}+\frac{f(x_1)+f(x_2)}{2\Delta x}+\ldots+\frac{f(x_{N-2})+f(x_{N-1})}{2\Delta x}+\frac{f(x_{N-1})+f(x_N)}{2\Delta x} \qquad (2.15)$$

Po převodu z analytického tvaru do numerického:

$$I=\frac{b-a}{N}\left[f(x_0)+2f(x_1)+2f(x_2)+\ldots+f(x_{N-1})+f(x_N)\right] \qquad (2.16)$$

2.4 Výpočet neurčitého integrálu

Začneme generováním průběhu funkce cosinus. Postupujeme stejným způsobem jako v úloze 1. Jediný rozdíl bude v tom, že místo funkce sinus potřebujeme funkci cosinus. Funkci cosinus požadujeme, protože integrování je opačným způsobem k derivování. Tedy z funkce cosinus dostaneme funkci sinus.

Generování funkce cosinus:

```
clc
clear
N=256;
a=0;
b=2*pi;
krok=(b-a)/N;
x=a:krok:b-krok;
y=cos(x);
```

Teď přistoupíme k výpočtu neurčitého integrálu obdélníkovou metodou. Nejdříve musíme provést výpočet pro první vzorek, pro který nelze počítat s předchozím. Výpočet vzorku od dvou do N umístíme do cyklu *for*.

Příklad výpočtu neurčitého integrálu obdélníkovou metodou:

```
inty_obd(1)=krok*y(1);
for I=2:N,
    inty_obd(I)=inty_obd(I-1)+krok*y(I);
end
```

Při výpočtu neurčitého integrálu lichoběžníkovou metodou postupuje obdobně. Jediný rozdíl bude ve vzorci a při výpočtu prvního vzorku, kde za něj dosadíme nulu.

Příklad výpočtu neurčitého integrálu lichoběžníkovou metodou:

```
inty_lich(1)=0;
for I=2:N,
    inty_lich(I)=inty_lich(I-1)+krok*(y(I)+y(I-1))/2;
end
```

Jako další bod programu je vykreslení křivek do grafu

```
subplot 311                              %graf 1. nahoře
plot(x,inty_obd,x,inty_lich)
grid;
```

Jestli se nám v grafu objeví průběh dvou sinusovek, je to důkaz správného výpočtu.

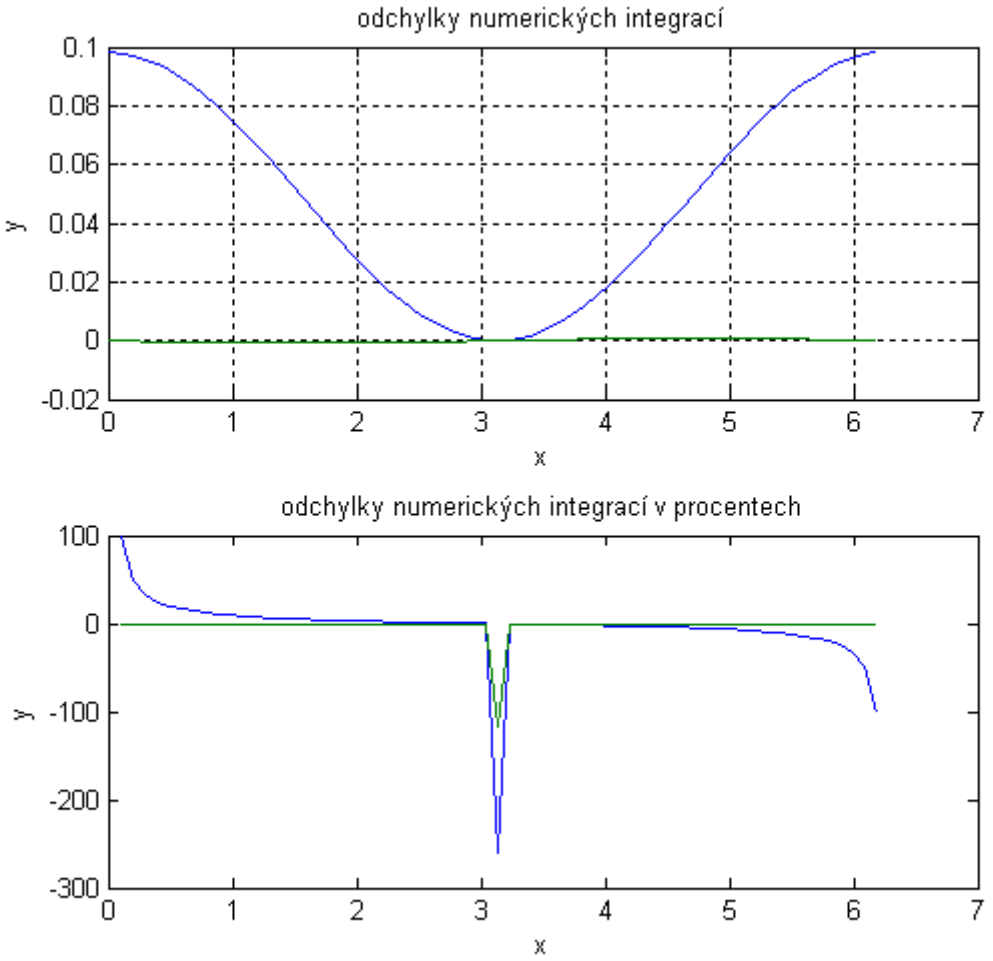
2.5 Výpočet odchylek integrací

Pro určení odchylky je potřeba analyticky integrovaná funkce cosinu. Víme, že pro funkci cosinus je její integrace funkce sinus. V tom případě stačí vygenerovat funkci sinus. Jako další krok programu je odečtení vektorů proběhu funkcí numerických integrací od vektoru funkce sinus. Tím získáme odchylku. Pro získání odchylek procentuálních je potřeba podíl vektorů analytické integrace od numerické vydělit vektorem analytické integrace a vynásobit stem. Aby nám odchylky k něčemu byly, je potřeba je vykreslit do grafů.

Analytická integrace funkce cosinus a výpočet odchylek s jejich vykreslením:

```
inty_anal=sin(x);
Oo1=inty_obd-inty_anal;                  %odchylka obdelnikove
Ol1=inty_lich-inty_anal;                 %odchylka lichobeznikove
subplot 312                              %graf 2. Uprostřed
plot(x,Oo1,x,Ol1)
grid;
Oo2=(inty_obd-inty_anal)./inty_anal*100;  %odchylka obdel. v %
Ol2=(inty_lich-inty_anal)./inty_anal*100; %odchylka lich. v %
subplot 313                              %graf 3. Uprostřed
plot(x,Oo2,x,Ol2)
```

Zde si všimněte použití tečky u výpočtu odchylky v procentech. Tečka v MATLABu znamená, že se daná operace musí provést pro každý vzorek. V našem případě se musí každý vzorek podělit integraci analytickou.



Obr. 2. 6 – Odchylky numerických integrací

Z grafu je vidět, že použití odchylek v procentech je nepřesné a v některých místech nám průběh ukazuje veliké výkyvy, které nastávají, když dělíme nulou, a to je v místech, kde funkce prochází právě nulou. Z toho důvodu je použití odchylek v procentech nepřesné, je lepší použití normálních odchylek.

2.6 Výpočet chyby obou metod

U určení chyby obdélníkové metody vycházíme ze vztahu (2.3). Pro výpočet je potřeba derivovat výchozí neintegrovanou funkci cosinus. Poté najdeme maximum této derivace, zde si musíme zadat, že se maximum může nacházet i v záporných hodnotách, proto je potřeba použít absolutní hodnotu. Teď nám stačí zapsat vztah pro výpočet chyby integrace obdélníkovou metodou do MATLABu.

Příklad výpočtu chyby obdélníkové metody a první derivace:

```
for I=1:N,
    if I == 1
         der1(I)=(y(I+1)-y(I))/krok;            % dopředná
       elseif I == N
```

```
        der1(I)=(y(I)-y(I-1))/krok;           % zpětná
      else
        der1(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
      end
end
Do=max(abs(der1));                  %vypocet maxima prvni derivace
Chyba_obd=100*(Do*(b-a)^2)/N^       %vypocet chyby u obdelnikove v %
```

Chyba je počítána v procentech, takže je vynásobena stem.

Pro výpočet chyby u lichoběžníkové metody se držíme vztahu (2.10). Zde je pro výpočet potřeba provést derivaci dvakrát. Tedy protočíme funkci cosinus cyklem derivace dvakrát. Jako další najdeme maximum druhé derivace, to uděláme stejným způsobem jako u výpočtu chyby obdélníkové metody a dále použijeme vzorec pro výpočet chyby lichoběžníkové metody.

Příklad výpočtu chyby obdélníkové metody a druhé derivace:

```
for I=1:N,
    if I == 1
        der2(I)=(y(I+1)-y(I))/krok;           % dopředná
       elseif I == N
        der2(I)=(y(I)-y(I-1))/krok;           % zpětná
      else
        der2(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
      end
end
Dl=max(abs(der2));                      %výpocet maxima druhe derivace
Chyba_lich=100*(Dl*(b-a)^3)/(12*N^3)  %vypocet chyby u lich v %
```

Zde je také chyba počítaná v procentech tedy násobena stem.

2.7 Výpis celého programu

```
Clc
clear
a=0;
b=2*pi;
N=64;
krok=(b-a)/N;
x=a:krok:b-krok;
y=cos(x);
%*****************************************************************
% Vypocet neurciteho integralu - obdelnikova metoda
%-----------------------------------------------------------------
inty_obd(1)=krok*y(1);
for I=2:N,
    inty_obd(I)=inty_obd(I-1)+krok*y(I);
end
%*****************************************************************
% Vypocet neurciteho integralu - lichobeznikova metoda
```

```
%-----------------------------------------------------------------------
inty_lich(1)=0;
for I=2:N,
    inty_lich(I)=inty_lich(I-1)+krok*(y(I)+y(I-1))/2;
end
subplot 311                                 %graf 1. Nahoře
plot(x,inty_obd,x,inty_lich)
grid;
inty_anal=sin(x);
Oo1=inty_obd-inty_anal;                     %odchylka obdelnikove
Ol1=inty_lich-inty_anal;                    %odchylka lichobeznikove
subplot 312                                 %graf 2. Uprostřed
plot(x,Oo1,x,Ol1)
title('odchylky numerických integrací');
xlabel('x');
ylabel('y');
grid;
Oo2=(inty_obd-inty_anal)./inty_anal*100;  %odchylka obdelnikove v %
Ol2=(inty_lich-inty_anal)./inty_anal*100; %odchylka lichobeznikove v
%

subplot 313                                 %graf 3. Uprostřed
plot(x,Oo2,x,Ol2)
title('odchylky numerických integrací v procentech');
xlabel('x');
ylabel('y');
%***********************************************************************
% Vypocet prvni derivace
%-----------------------------------------------------------------------
for I=1:N,
    if I == 1
         der1(I)=(y(I+1)-y(I))/krok;           % dopředná
       elseif I == N
         der1(I)=(y(I)-y(I-1))/krok;           % zpětná
       else
         der1(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
       end
end
Do=max(abs(der1));                      %vypocet maxima prvni
derivace
Chyba_obd=100*(Do*(b-a)^2)/N^2          %vypocet chyby u obdelnikove
v %
%***********************************************************************
% Vypocet druhé derivace
%-----------------------------------------------------------------------
for I=1:N,
    if I == 1
         der2(I)=(y(I+1)-y(I))/krok;           % dopředná
```

```
        elseif I == N
          der2(I)=(y(I)-y(I-1))/krok;           % zpětná
        else
          der2(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
        end
end
Dl=max(abs(der2));                    %výpocet maxima druhe derivace
Chyba_lich=100*(Dl*(b-a)^3)/(12*N^3)  %vypocet chyby u lich. metody
v %
```

3 Aplikace numerických derivací a integrací ve 2D

Obsah

1. Generování funkce sinus a její derivace
2. Rozšíření programu o výpočet:
   a) plochu pod rovinnou křivkou
   b) délku oblouku rovinné křivky
   c) střední hodnotu
   d) efektivní hodnotu
   e) objem tělesa, které vznikne při rotaci křivky kolem osy x => objem rotačního tělesa
   f) moment setrvačnosti rotačního tělesa
   g) těžiště rotačního tělesa

2.2 Generace rovinné křivky a její derivace

Než začneme s výpočty, musíme si nejdříve vygenerovat rovinnou křivku, v našem případě průběh funkce sinus. To provedeme tak, že si nejdřív určíme interval, kterým bude křivka probíhat <a;b>, a počet vzorků N. Doporučuji průběh křivky nastavit na interval <0;π> a počet vzorku N=256. Potom si určíme frekvenci (krok), která je potřeba k získání souřadnic *x*. Po získání *x*-ových souřadnic vygenerujeme funkci. V MATLABu se generování zapíše tímto způsobem:

```
a=0;
b=pi;
N=256;
krok=(b-a)/N;
x=a:krok:b-krok;
y=sin(x);
```

Dále je potřeba provést derivaci rovinné křivky, kterou bude potřeba pro výpočet délky křivky. Pro funkci sinus je její derivace funkce cosinus, to najdeme v MFCHT, což nám pomůže pro kontrolu při výpisu do grafu.

Celý vypočet derivace a výpis křivek do grafu v MATLABu:

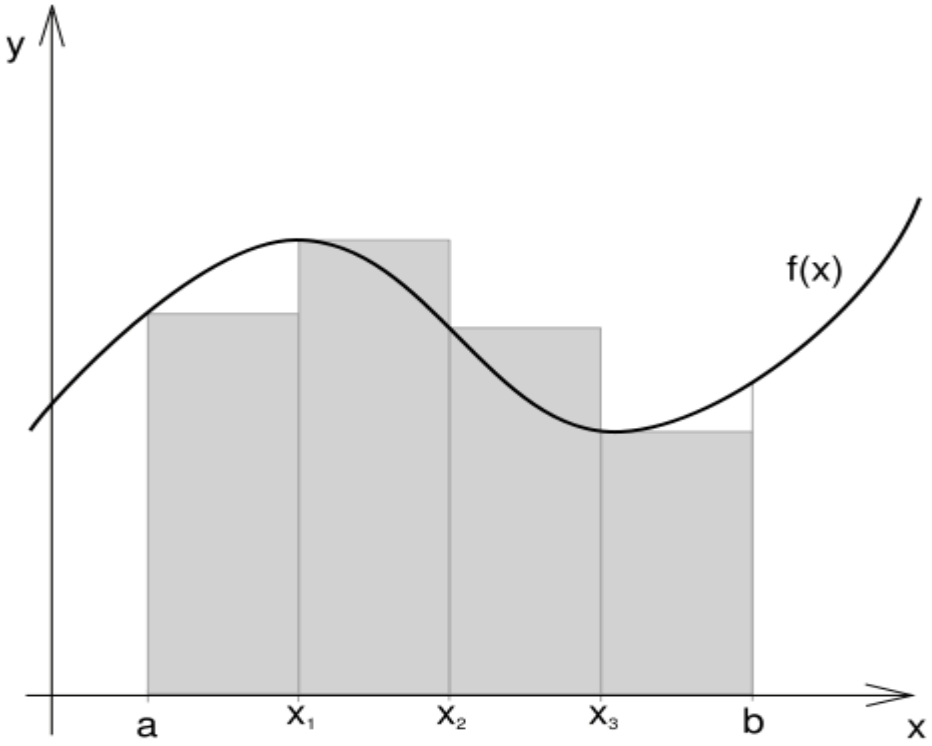
```
for I=1:N,
    if I == 1
         dery(I)=(y(I+1)-y(I))/krok;          % dopredna derivace
    elseif I == N
         dery(I)=(y(I)-y(I-1))/krok;          % zpetna derivace
    else
         dery(I)=0.5*(y(I+1)-y(I-1))/krok;    % centralni derivace
    end
end
```

```
plot(x,y,x,dery)
grid;
```

Nyní můžeme přistoupit k samotným výpočtům.

3.2 Plocha pod rovinnou křivkou

Úkolem je zjistit obsah plochy mezi osou *x* a funkcí f(x). Z numerických integrací víme, že můžeme použít tři základní metody pro výpočet. Pro získání plochy použijeme obdélníkovou metodu, kde je výpočet plochy pod funkcí f(x) nahrazen výpočtem plochy pod obdélníkem.



Obr. 3. 7 – Obdélníková metoda

Analytické vyjádření výpočtu plochy pod funkcí f(x) obdélníkovou metodou:

$$S = \int_a^b f(x)dx \qquad (3.1)$$

Převodem analytického tvaru do numerického dostáváme:

$$S = \Delta x \sum_{i=1}^{N} f(x_{i-1}) \qquad (3.2)$$

kde $f(x_i)$ …… je okamžitá hodnota (funkční hodnota) i-tého vzorku.

Příklad výpočtu plochy v MATLABu:

```
obsah=krok*sum(y)
```

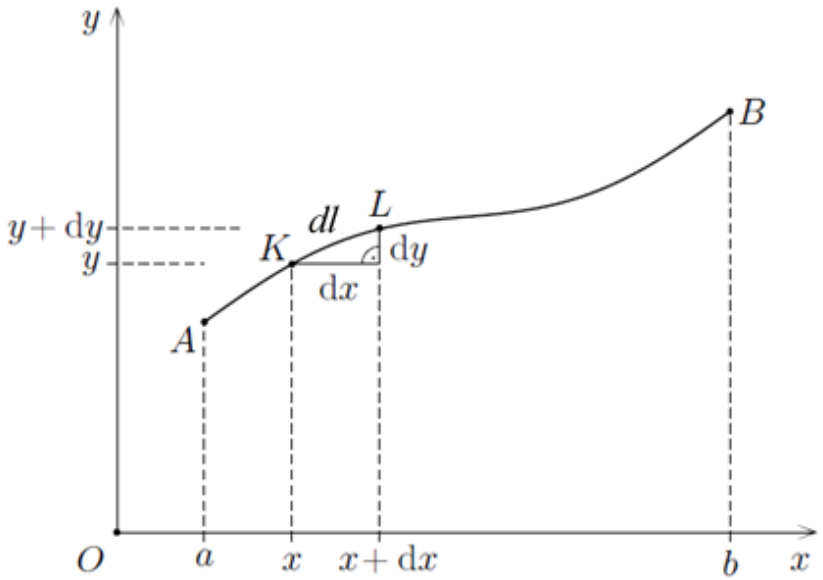
Dále nesmíme zapomenout, že když má funkce průběh pod osou x, tak by se hodnoty odčítaly nebo by vycházely záporné hodnoty, proto musíme použít absolutní hodnotu ke každému vzorku.

Příklad výpočtu plochy v MATLABu s absolutní hodnotou:

```
obsah=krok*sum(abs(y))
```

3.3 Délka oblouku rovinné křivky

Základním předpokladem pro výpočet délky oblouku rovinné křivky je to, že funkce f(x) je definovaná a diferencovatelná na intervalu <a,b>. Při výpočtu vycházíme z následujícího obrázku obr. 3.1.



Obr. 3. 8 – Délka dl elementu křivky

Při výpočtu postupujeme tak, že uvažujeme velmi malý element *dl* mezi body *KL* dané křivky, jehož průmětem do osy *x* je interval <x, x+dx>. Délka průmětu do osy *y* je dána vztahem:

$$dy = f'(x)dx \qquad (3.5)$$

Z Pythagorovy věty (oblouk KL můžeme aproximovat úsečkou) použité na pravoúhlý trojúhelník KLM pak pro délku *dl* úsečky KL dostáváme:

$$dl = \sqrt{dx^2 + dy^2} = \sqrt{dx^2 \cdot \left(1 + \frac{dy^2}{dx^2}\right)} = \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx = \sqrt{1 + [f'(x)]^2}\, dx \qquad (3.3)$$

Pro celkovou délku *l* uvažované křivky pak platí:

$$l = \int_a^b \sqrt{1+\left[f'(x)\right]^2}\,dx \qquad (3.4)$$

Po převodu analytického vztahu (3.4) na numerický dostáváme:

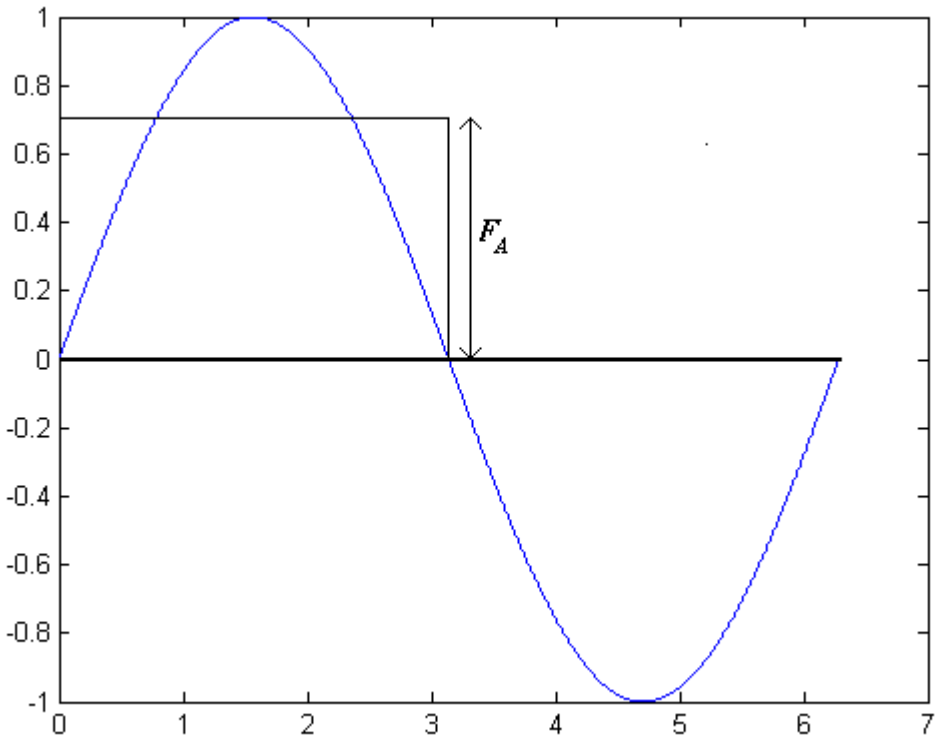
$$l = \sum_{i=1}^{N} \sqrt{1+f^2(x_i)} \cdot \Delta x = \Delta x \cdot \sum_{i=1}^{N} \sqrt{1+\left[f'(x_i)\right]^2} \qquad (3.5)$$

Příklad zápisu vztahu (3.6) v MATLABu:

```
Delka_l=krok*sum((1+dery.^2).^0.5)    % Delka oblouku rovinne krivky
```

3.4 Střední hodnota

Střední hodnotou rozumíme výšku obdélníka se stejnou plochou jako plocha pod křivkou za jednu periodu.



Obr. 3. 9 – Výška střední hodnoty

Analytické vyjádření střední hodnoty obecného signálu *f(t)* je dána následujícím vztahem:

$$F_A = \frac{1}{T}\int_0^T f(x)dx \qquad (3.6)$$

Po převodu analytického tvaru do numerického dostáváme přibližně:

$$F_A \cong \frac{1}{N \cdot \Delta x} \cdot \sum_{i=1}^{N} f(x_i) \cdot \Delta x \qquad (3.7)$$

Po úpravě získáme:

$$F_A \cong \frac{1}{N} \cdot \sum_{i=1}^{N} f(x_i) \qquad (3.8)$$

Příklad zapsání vztahu (3.8) do MATLABu:

```
stredni=(1/N)*sum(y)
```

3.5 Efektivní hodnota

Analytické vyjádření efektivní hodnoty obecného signálu *f(x)* je dána následujícím vztahem:

$$F_{ef} = \sqrt{\frac{1}{T}\int_0^T f^2(x)dx} \qquad (3.9)$$

Po úpravě do numerického tvaru platí:

$$F_{ef} \cong \sqrt{\frac{1}{N \cdot \Delta x} \cdot \sum_{i=1}^{N} f^2(x_i) \cdot \Delta x} \qquad (3.10)$$

Po zjednodušení:

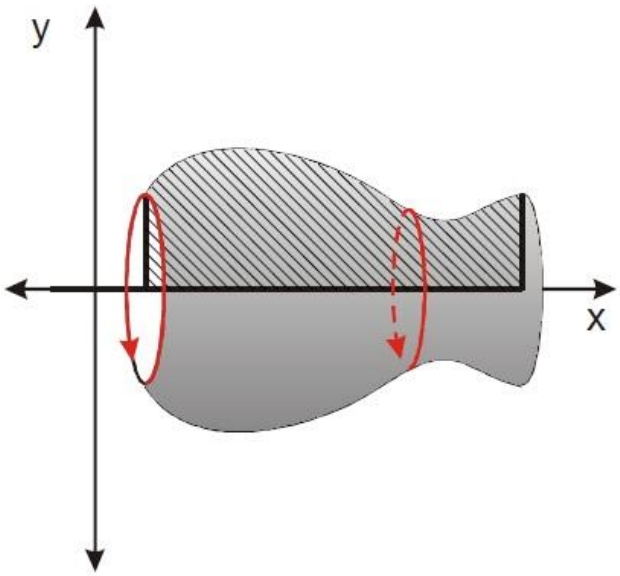
$$F_{ef} \cong \sqrt{\frac{1}{N}} \cdot \sqrt{\sum_{i=1}^{N} f^2(x_i)} \qquad (3.11)$$

Vztah (3.12) mužeme do MATLABu zapsat takto:

```
efektivni=(1/N)^0.5*(sum(y.^2)).^0.5
```

3.6 Objem rotačního tělesa

Začneme představou. Máme libovolnou rovinnou křivku, když se otočí kolem osy x o 360º vždy ve stejné vzdálenosti od osy, vznikne těleso, u kterého máme vypočítat objem.

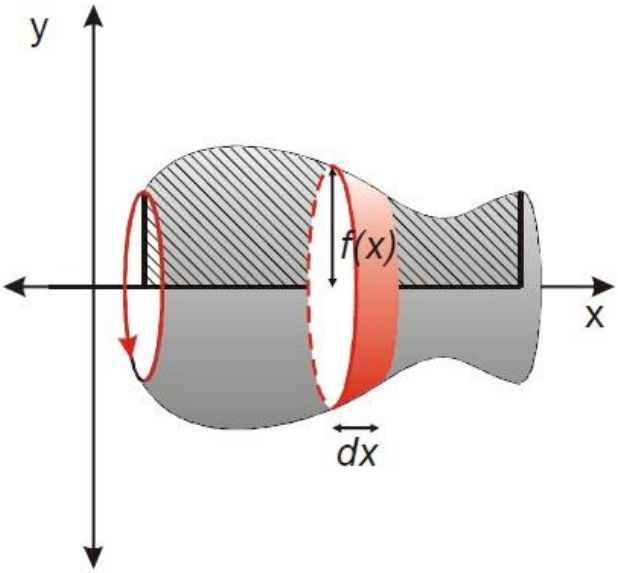


Obr. 3. 10 – Těleso vzniklé rotací křivky

Obecně objem vypočítáme ze vztahu:

$V = v \cdot \pi \cdot r^2$	(3.12)

Abychom mohli určit objem jakéhokoliv tělesa vzniklého rotací libovolné funkce, musíme ho rozdělit na jednotlivé válečky, kde výška je *dx* a poloměr je vzdálenost funkce *f(x)* od osy *x* v určitém vzorku.



Obr. 3. 11 – Těleso vzniklé rotací křivky s ukázkou jednotlivého vzorku

Když za výšku tělesa dosadíme krok *dx* a za poloměr vzdálenost bodu funkce f(x) od osy x v určitém vzorku, dostaneme vztah pro výpočet válce v určitém vzorku:

$$dV = dx \cdot \pi \cdot f^2(x) \tag{3.13}$$

A pro celou křivku:

$$V = \int_a^b \pi \cdot f^2(x) \cdot dx \tag{3.14}$$

Převodem analytického tvaru do numerického získáme:

$$V = \Delta x \cdot \pi \cdot \sum_{i=1}^{N} f^2(x_i) \tag{3.15}$$

Vztah (3.16) pro výpočet v MATLABu:

```
objem=pi*krok*sum(y.^2)
```

3.7 Moment setrvačnosti rotačního tělesa

Moment setrvačnosti je fyzikální veličina. Určuje velikost setrvačnosti tělesa při otáčivém pohybu, která závisí na rozložení hmoty v tělese vzhledem k ose otáčení.

Moment setrvačnosti obecného tělesa:

$$J = \int r^2 dm = \int J \, dJ \tag{3.16}$$

Moment setrvačnosti elementárního válečku:

$$dJ = \frac{1}{2} \rho \cdot dV \cdot r^2 \;\Rightarrow\; dJ = \frac{1}{2} \rho \cdot dV \cdot f^2(x) \tag{3.17}$$

Element objemu:

$$dV = \pi \cdot r^2 \cdot dx = \pi \cdot f^2(x) \cdot dx \tag{3.18}$$

Po dosazení do rovnice dJ (3. 17) za dV (3.18) dostáváme:

$$dJ = \frac{1}{2} \rho \cdot \pi \cdot f^4(x) \cdot dx \tag{3.19}$$

Pro moment setrvačnosti celého rotačního tělesa v analytice pak platí:

$$J = \int_a^b \frac{1}{2}\rho \cdot \pi \cdot f^4(x) \cdot dx = \frac{1}{2}\rho \cdot \pi \cdot \int_a^b f^4(x) \cdot dx \qquad (3.20)$$

Po převodu do numerického tvaru dostáváme:

$$J \cong \frac{1}{2}\rho \cdot \pi \cdot \sum_{i=1}^{N} f^4(x_i) \cdot \Delta x = \frac{1}{2}\rho \cdot \pi \cdot \Delta x \cdot \sum_{i=1}^{N} f^4(x_i) \qquad (3.21)$$

Při známé hmotnosti lze pak moment setrvačnosti rotačního tělesa určit podle vztahu:

$$J = \frac{1}{2} m \cdot \sum_{i=1}^{N} f^2(x_i) \qquad (3.22)$$

Samozřejmě pro získání hmotnosti si musíme určit hustotu, bez které hmotnost nemůžeme vypočítat, a tak i celý moment setrvačnosti.

Příklad výpočtu momentu setrvačnosti v MATLABu:

```
hustota=10
msetrvacnosti=0.5*hustota*pi*krok*sum(y.^4)
```

3.8 Těžiště rotačního tělesa

Těžiště je hmotný bod (hmotný střed), na který působení gravitační síly vyvolá stejný účinek jako na celé těleso.

Obecný vztah pro výpočet těžiště:

$$x_T = \frac{\int x\, dm}{m} \qquad (3.23)$$

kde $\mathbf{x_T}$ ..... je vzdálenost od dolní meze intervalu funkce f(x)

$$dm = \rho \cdot dV = \rho \cdot \pi \cdot f^2(x) \cdot dx \qquad (3.24)$$

$$m = \int_a^b \rho \cdot \pi \cdot f^2(x) \cdot dx \qquad (3.25)$$

Po dosazení vztahů (3.21) a (3.25) do (3.23) dostáváme:

$$x_T = \frac{\int_a^b x \cdot \rho \cdot \pi \cdot f^2(x) \cdot dx}{\int_a^b \rho \cdot \pi \cdot f^2(x) \cdot dx} = \frac{\rho \cdot \pi \cdot \int_a^b x \cdot f^2(x) \cdot dx}{\rho \cdot \pi \cdot \int_a^b f^2(x) \cdot dx} = \frac{\int_a^b x \cdot f^2(x) \cdot dx}{\int_a^b f^2(x) \cdot dx} \quad (3.26)$$

Po úpravě vztahu (3.26) do numerické podoby:

$$x_T = \frac{\sum_{i=1}^{N} x_i \cdot f^2(x_i) \cdot \Delta x}{\sum_{i=1}^{N} f^2(x_i) \cdot \Delta x} = \frac{\Delta x \cdot \sum_{i=1}^{N} x_i \cdot f^2(x_i)}{\Delta x \cdot \sum_{i=1}^{N} f^2(x_i)} = \frac{\sum_{i=1}^{N} x_i \cdot f^2(x_i)}{\sum_{i=1}^{N} f^2(x_i)} \quad (3.27)$$

Vztah (3.27) pro výpočet v MATLABu:

```
teziste=sum(x.*y.^2)/sum(y.^2)
```

3.9 Výpis celého programu

```
clc
clear
a=0;
b=2*pi;
N=256;
krok=(b-a)/N;
x=a:krok:b-krok;
y=sin(x);
%*****************************************************************
% Vypocet derivace
%-----------------------------------------------------------------
for I=1:N,
    if I == 1
         dery(I)=(y(I+1)-y(I))/krok;           % dopředná
    elseif I == N
         dery(I)=(y(I)-y(I-1))/krok;           % zpětná
       else
         dery(I)=0.5*(y(I+1)-y(I-1))/krok;     % centrální
       end
end
plot(x,y,x,0)
grid;
%*****************************************************************
% Uziti integralniho poctu
%-----------------------------------------------------------------
hustota=10;
obsah=krok*sum(abs(y))
delka=krok*sum((1+dery.^2).^0.5)
objem=pi*krok*sum(y.^2)
stredni=(1/N)*sum(y)
```

```
efektivni=(1/N)^0.5*(sum(y.^2)).^0.5
msetrvacnosti=0.5*hustota*pi*krok*sum(y.^4)
teziste=sum(x.*y.^2)/sum(y.^2)
```

4 Řešení nelineárních rovnic metodou postupné aproximace

Obsah

1. Program pro řešení nelineárních rovnic metodou postupní aproximace. Program je napsán pro rovnici $8x^3 - 6x - 1 = 0$

2.3 Postupná aproximace

Postupná aproximace neboli postupné přiblížení se používá hlavně u A/D (analogově číslicový) převodníků, které fungují na tomto principu. My metodu postupné aproximace použijeme k řešení nelineárních rovnic, které nelze vypočítat analyticky. Tedy si postupně budeme přibližovat k výsledku. Přesnost není stoprocentní jako v analytice, ale nepřesnost je minimální závislá na počtu cyklů.

Příklad 1:

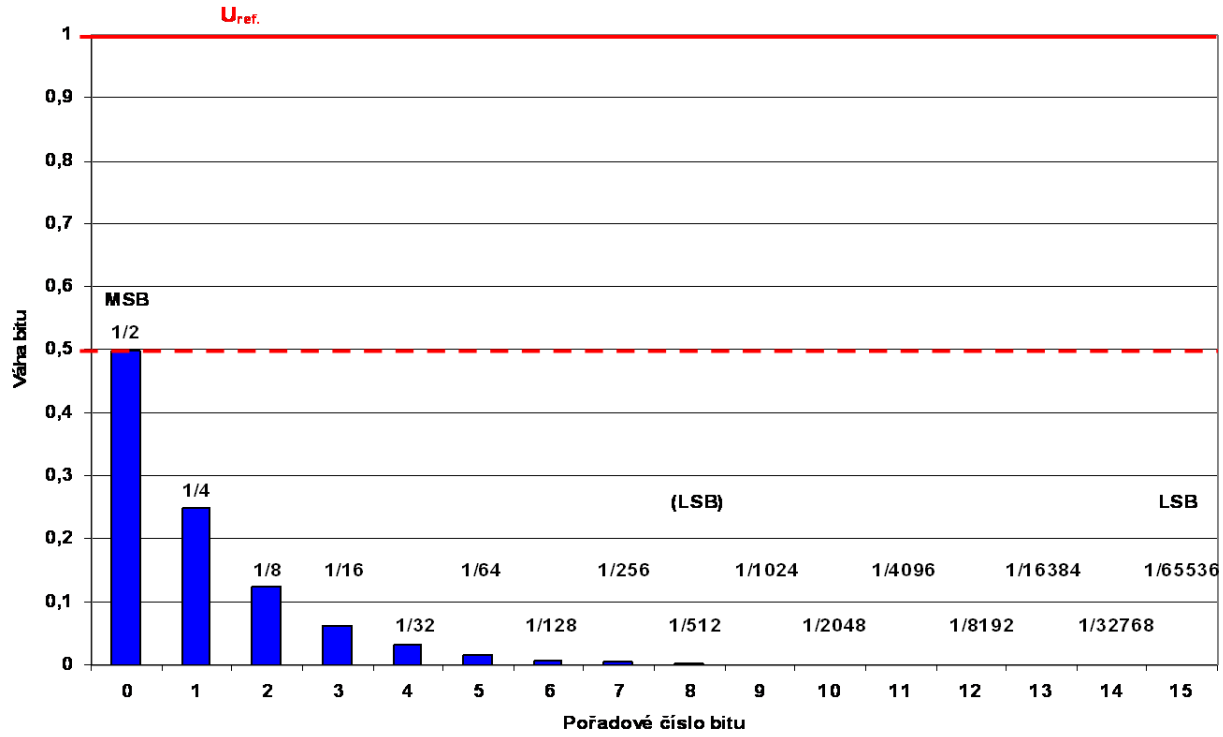
$3x - 4 = \pi$

$x = \frac{\pi + 4}{3}$ Lineární rovnice, analytické řešení

Příklad 2:

$8x^3 - 6x - 1 = 0$ Nelineární kubická rovnice nemá analytické řešení => numerické řešení pomocí postupné aproximace.

4.2 Princip postupné aproximace



Obr. 4. 12 – Princip postupné aproximace

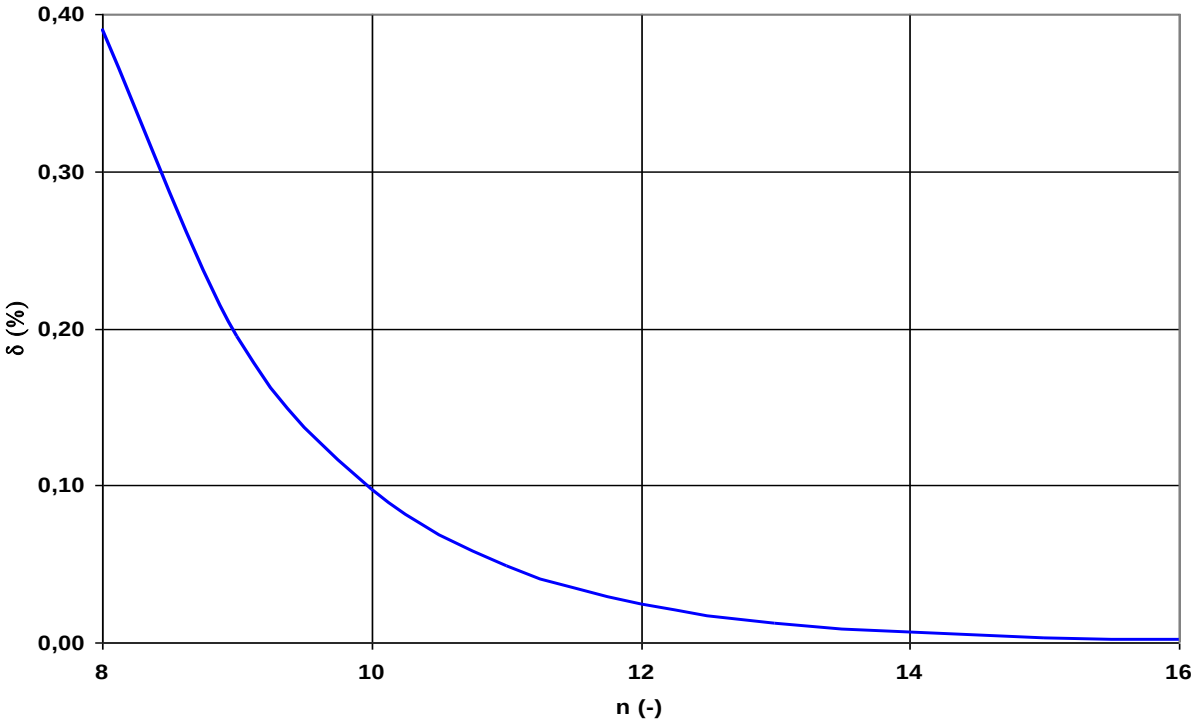
Kde $U_{ref.}$ …… je reference (velikost intervalu, na kterém hledáme neznámou)

Postupná aproximace spočívá ve hledání neznámé (konstanty) *y(x)*, která leží na intervalu <a;b>. Interval na obrázku je vzdálenost mezi osou *x* a $U_{ref.}$. Na ose x vidíme pořadové číslo bitu, kterým je mocněna dvojka, která nám dělí referenci.

$$LSB = \frac{U_{ref.}}{2^n}$$ n …… pořadové číslo bitu (počet cyklů)

$$U_{ref.} = b - a$$

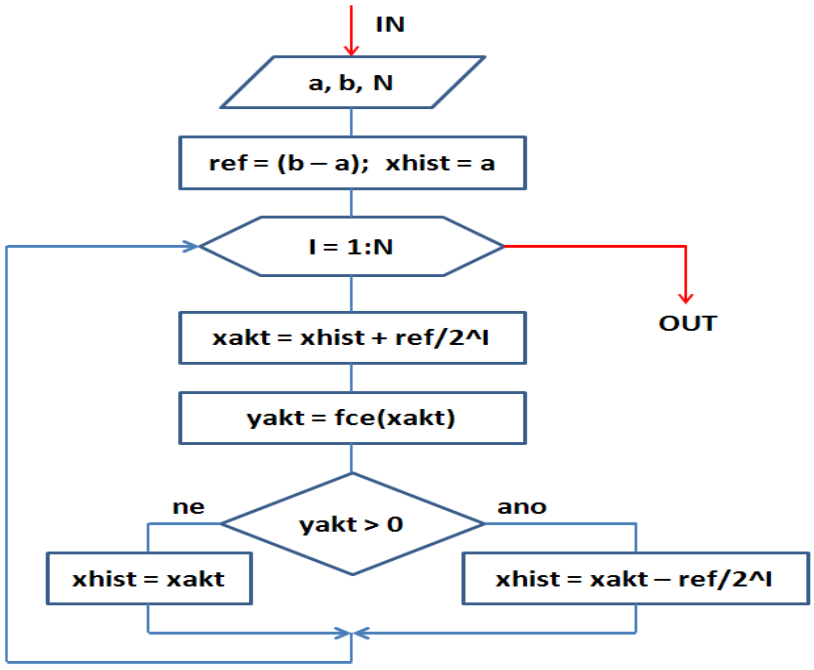
Každý krok algoritmu přičítáme LSB s následujícím pořadím bitu. Když se přičtení nevejde do neznámé (přeteče), tak poslední hodnotu odečteme. V dalším kroku vezmeme LSB s pořadím bitu +1 a přičteme k hodnotě $U_{ref.}$, která se do neznámé vešla. Tento postup opakujeme podle počtu cyklů.



Obr. 4. 13 – Závislost chyby na aproximační metody na počtu kroků algoritmu

Z obrázku vidíme, že pro 16 kroků je skoro 100% přesnost a chyba je zanedbatelná. V další části budeme počítat s počtem kroků 16.

4.3 Princip programu



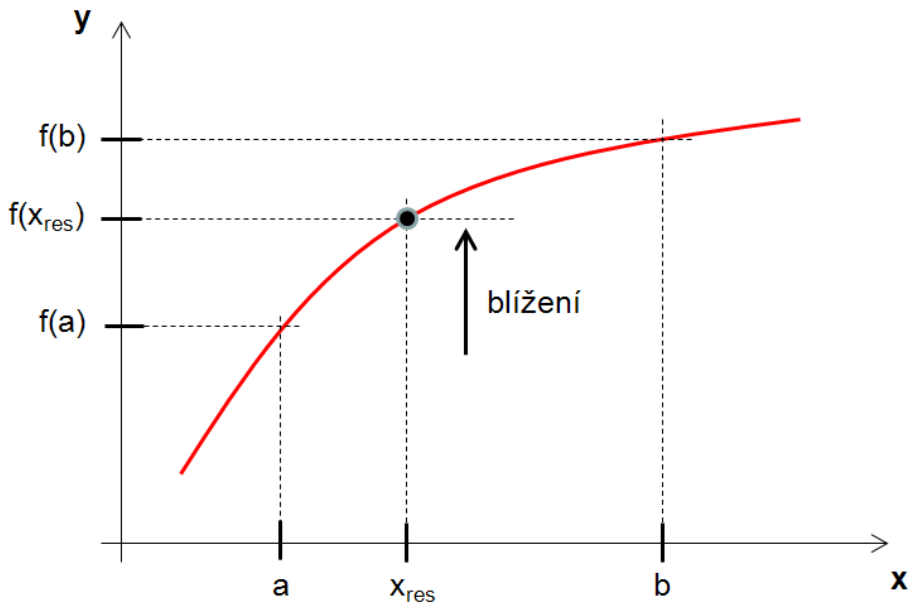
Obr. 4. 14 – Vývojový diagram pro rostoucí funkci

Na vývojovém diagramu vidíme princip postupné aproximace, který platí **jen pro rostoucí funkce**. Diagram je definován pro rovnici. Na začátku programu definujeme začátek intervalu *a*, konec intervalu *b* a počet cyklů N. Interval si určíme tak, aby na něm ležel průsečík rovnice s osou *x*. Dále si vygenerujeme *x* souřadnice, které jsou důležité pro průběh rovnic. Jako další přiřadíme Xhist začátek intervalu. Potom do programu umístíme cyklus for, ve kterém bude podmínka if, která rozhodne jestli Yres nepřekročilo průsečík a podle toho zapíše za Xhist. Cyklus for bude probíhat od jedné do N. Po N-krát proběhnutí cyklu se na výstupu vypíše řešení uložené v Xakt.

Program pro rostoucí rovnice s příkladem uvedenou rovnici:

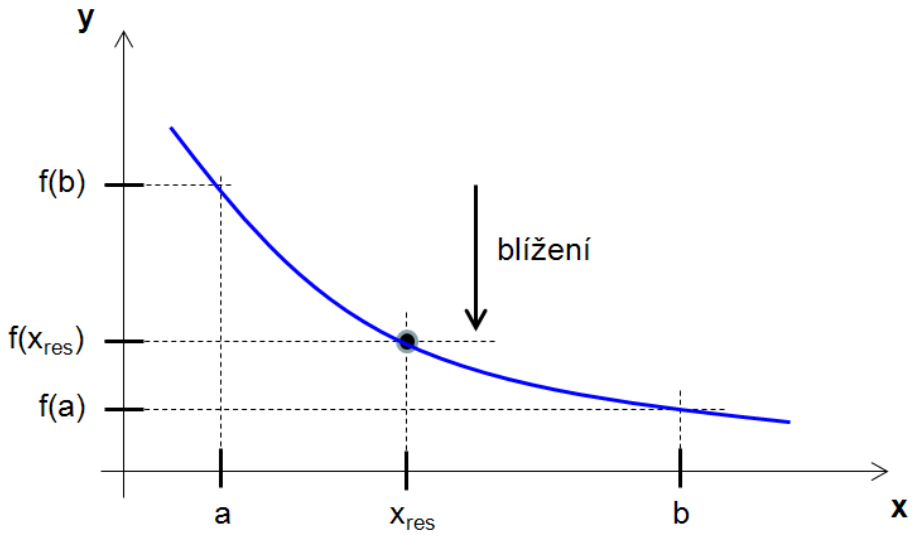
```
clc
clear
a=-8;
b=-6;
ref=b-a
x=a:Xref/100:b;
y=8*x.^3-6*x-1;
plot(x,y);
grid;
N=16;
Xhist=a;
for I=1:N,
    Xakt=Xhist+Xref/(2^I);
    Yres=8*Xakt^3-6*Xakt-1;
    if Yres>0,
        Xhist=Xakt-Xref/(2^I);
    else
        Xhist=Xakt;
    end
end
reseni=Xakt
```

Než začneme algoritmus používat pro jinou rovnici, musíme ji upravit do tvaru, kde se levá strana bude rovnat nule. Potom je potřeba v programu přepsat rovnici, kde generujeme průběh rovnice *y(x)* a v cyklu pro Yres. Na následujícím obrázku vidíme, že pro rostoucí funkci se blížíme ze zdola. To znamená, že podmínka *if* testuje, jestli Yres není větší než nula.



Obr. 4. 15 – Aproximace pro rostoucí funkci

Pro klesající funkce se blížíme ze shora, tedy testujeme, jestli hodnota Y_res neklesla pod nulu.



Obr. 4. 16 – Aproximace pro klesající funkci

Nyní, když známe předpoklady pro rostoucí i klesající rovnici, můžeme program upravit pro libovolnou rovnici. Musíme v programu vypočíst konstantu *k*, podle které určíme, jestli je funkce klesající nebo rostoucí.

Vztah pro výpočet konstanty:

$$k = \frac{f(b) - f(a)}{b - a}$$	(1.4)

Do cyklu vložíme ještě jednu podmínku if, která bude testovat, jestli konstanta je menší nebo větší než nula. Když je konstanta větší než nula, podmínka pošle průběh programu do části pro rostoucí rovnici, a když je menší než nula, tak naopak.

Program pro libovolné rovnice s příkladem první rovnice:

```
clc
clear
a=0.5;                  %zacatek intervalu
b=1.5;                  %konec intervalu
ref=b-a                 %vypocet reference
x=a:Xref/100:b;         %generování x vektoru
y=8*x.^3-6*x-1;         %generovani prubehu rovnice
plot(x,y)
grid
ya=8*a^3-6*a-1;
yb=8*b^3-6*b-1;
k=(yb-ya)/(b-a)
N=16;                   %pocet kroku cyklu
Xhist=a;
for I=1:N,
    Xakt=Xhist+Xref/(2^I);
    Yres=8*Xakt^3-6*Xakt-1;
    if k>=0
        if Yres>0,
            Xhist=Xakt-Xref/(2^I);
        else
            Xhist=Xakt;
        end
    else k<0
        if Yres<0,
            Xhist=Xakt-Xref/(2^I);
        else
            Xhist=Xakt;
        end
    end
end
reseni=Xakt
```

5 Fourierův rozvoj funkce

Obsah

1. Program, který dokáže popsat libovolný periodický signál pomocí Fourierova rozvoje a zobrazí jednotlivé členy Fourierova rozvoje do kmitočtového spektra

5.1 Definice Fourierova rozvoje

Fourierův rozvoj (Fourierova řada) byl pojmenován po francouzském matematikovi a fyzikovi Josephu Fourierovi. Je určen k zápisu jakéhokoliv periodického signálu pomocí goniometrických funkcí sinus a cosinus. Tedy každá periodická funkce lze pomocí Fourierova rozvoje popsat nekonečnou řadou, jejíž jednotlivé složky mají charakter goniometrických funkcí. Jednotlivé složky nazýváme k-té harmonické.

Trigonometrická řada je speciálním případem obecněji pojatého Fourierova rozvoje:

$$f(t) \approx a_0 + \sum_{k=1}^{\infty} \left[a_k \cdot \cos(k\omega t) + b_k \cdot \sin(k\omega t) \right] \qquad (5.1)$$

Kde a_0 …… stejnosměrná (DC) složka;

k …… pořadové číslo k-té harmonické (celé);

a_k …… generátor amplitudy k-té harm. cos fce;

b_k …… generátor amplitudy k-té harm. sin fce;

Jednotlivé složky v analytice určíme ze vztahů:

$$a_0 = \frac{1}{T} \int_{\alpha}^{\alpha+T} f(t) dt \cong \frac{1}{N} \cdot \sum_{k=1}^{N} y_i \qquad (5.2)$$

$$a_k = \frac{2}{T} \int_{\alpha}^{\alpha+T} f(t) \cdot \cos(k\omega t) dt \cong \frac{2}{N} \sum_{k=1}^{N} y_i \cdot \cos(k\omega t) \qquad (5.3)$$

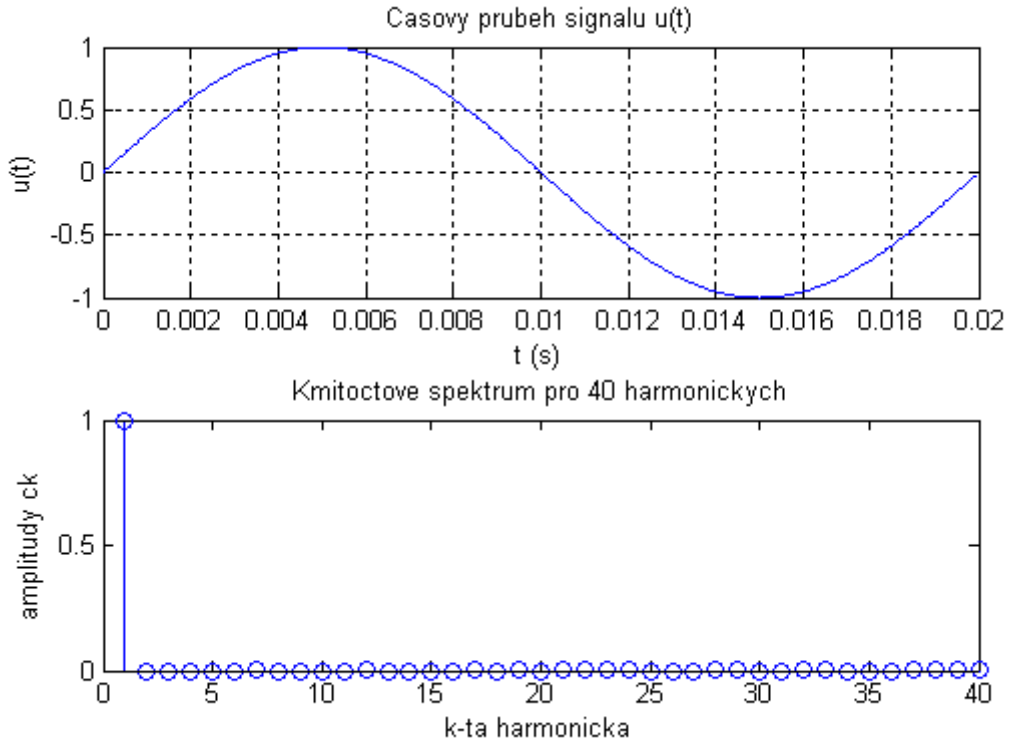
$$b_k = \frac{2}{T} \int_{\alpha}^{\alpha+T} f(t) \cdot \sin(k\omega t) dt \cong \frac{2}{N} \sum_{k=1}^{N} y_i \cdot \sin(k\omega t) \qquad (5.4)$$

Výslednou výšku vypočteme ze vztahu:

$$c_k = \sqrt{a_k^{\,2} \cdot b_k^{\,2}} \qquad (5.4)$$

5.2 Kmitočtové spektrum

Kmitočtové spektrum je vlastně ekvalizér, do kterého budeme zobrazovat jednotlivé k-té harmonické funkcí sinus a cosinus. Počet k-tých harmonických závisí na počtu kroku cyklu, který si ukážeme v další části.



Obr. 5. 1 – Průběh funkce sinus a její zobrazení do kmitočtového spektra

Na obrázku vidíme průběh funkce sinus, která je rozložena pomocí Fourierova rozvoje do jednotlivých k-tých harmonických. K-té harmonické vidíme zobrazené v kmitočtovém spektru. Zde je počet kroku cyklu 40, jak vidíme z obrázku.

5.3 Fourierův rozvoj v MATLABu

Jako první věc musíme nastavit základní parametry: frekvenci, počet kroků, ze kterých vypočteme periodu a krok a ještě určit ω (omega), která nám ovlivňuje průběh funkce *y(t).*

Nastavení základních parametrů:

```
f=50;                   %frekvence
N=256;                  %pocet kroku
T=1/f                   %perioda
krok=T/N;               %krok
omega=2*pi*f;
```

Z těchto parametrů si vygenerujeme časový vektor *t*, pomocí kterého vytvoříme průběh funkce *y(t).*

Generace časového vektoru a průběhu funkce:

```
t=0:krok:T-krok;     %generování casoveho vektoru
y=sin(omega*t);      %prubeh funkce y(t)
```

Průběh funkce y(t) je dobré si vykreslit do grafu, abychom jasně viděli, pro kterou funkci je vytvořeno kmitočtové spektrum.

Výstup funkce do grafu s popisem grafu:

```
subplot 211
plot(t,y)
title('Casovy prubeh signalu u(t)')
xlabel('t(s)')
ylabel('u(t)')
grid
```

Další krok spočívá v určení počtu kroku, tedy počtu harmonických vykreslených do kmitočtového spektra, a výpočtu stejnosměrné složky a_0, která nám říká, jak je výsledný signál posunut po ose *y*. Stejnoměrnou složku vypočteme ze vztahu 5.2.

Nastavení počtu harmonických a výpočet stejnosměrné složky:

```
PH=40;                %pocet harmonických
a0=(1/N)*sum(y)       %vypocet stejnosmerne slozky
```

Nyní nás čeká cyklus *for*, ve kterém vypočteme generátor amplitudy k-té harmonické funkce cosinus a sinus, tedy složky a_k a b_k, a výsledný signál c_k. Cyklus *for* bude probíhat od jedné do PH (počtu harmonických).

Cylkus for s výpočtem složek a výsledného signálu:

```
for k=1:PH,
    a(k)=(2/N)*sum(y.*cos(k*omega*t));
    b(k)=(2/N)*sum(y.*sin(k*omega*t));
    c(k)=(a(k)^2+b(k)^2)^0.5;
end
```

Nakonec jednotlivé složky výsledného signálu vykreslíme do grafu, k tomu si musíme vygenerovat vektor *kta.* Vektor vytvoříme od jedné po kroku jedna do PH, aby nám zobrazoval výšku výsledného signálu v jednotlivých krocích. Pro grad použijeme funkci *stem,* která nám vypíše výšku výsledného signálu v jednotlivých krocích.

Vektor kta a výstup do grafu:

```
subplot 212;
stem(kta,c)
title('Kmitoctove spektrum pro 40 harmonickych')
```

```
xlabel('k-ta harmonicka')
ylabel('amplitudy ck')
```

Nyní je program připraven pomocí Fourierova rozvoje rozložit libovolný periodický signál.

5.4 Výpis celého programu

```
clc
clear
f=50;               %frekvence
N=256;              %pocet kroku
T=1/f               %perioda
krok=T/N;           %krok
omega=2*pi*f;
t=0:krok:T-krok;    %generování casoveho vektoru
y=sin(omega*t);     %prubeh funkce y(t)
subplot 211

plot(t,y)
title('Casovy prubeh signalu u(t)')
xlabel('t (s)')
ylabel('u(t)')
grid
PH=40;              %pocet harmonických
a0=(1/N)*sum(y)     %vypocet stejnosmerne slozky
for k=1:PH,
    a(k)=(2/N)*sum(y.*cos(k*omega*t));
    b(k)=(2/N)*sum(y.*sin(k*omega*t));
    c(k)=(a(k)^2+b(k)^2)^0.5;
end
kta=1:1:PH;
subplot 212;
stem(kta,c)
title('Kmitoctove spektrum pro 40 harmonickych')
xlabel('k-ta harmonicka')
ylabel('amplitudy ck')
```

Závěr

V této práci jsem popsal numerické algoritmy, a k nim vytvořil programy. Programy jsem psal v aplikaci MATLAB. Algoritmy jsem rozdělil do pěti témat. Jako první jsou numerické derivace a integrace, které jsem následně použil v rovině. Tyto tři spolu úzce souvisí. Následuje další část, která už je odlišná od prvních třech, nazval jsem jí řešení nelineárních rovnic pomocí postupné aproximace. V poslední části Fourierův rozvoj funkce je z kraje vysvětleno o co se jedná a pak je popsán způsob jakým rozložit libovolnou funkci pomocí Fourierova rozvoje. Původně jsem chtěl ještě popsat použití numerických derivaci a integrací ve 3D, ale došel k závěru, že aplikaci bych musel velmi zúžit a bylo by to zbytečné.

U numerických derivací jsem převedl jejich analytický výpočet na numerický. Popsal jsem jak vytvořit průběh funkce v MATLABu a jak ji následně derivovat pomoci cyklu. Na závěr jsem vypočetl odchylku numerických derivací od analytických.

Část numerických integrací se zaměřila na výpočet neurčité integrálu. Zde jsem popsal jak vypočítat neurčitý integrál v MATLABu pomocí dvou metod. Tyto metody jsem nejdříve popsal a následně použil pomocí cyklů. Dále jsem zjistil odchylku těchto numerických výpočtů od výpočtů v analytice a to obecně i procentuálně. Odchylky jsem vypočetl pro obě metody. Na konci jsem ještě pomocí vzorců určil chybu těchto dvou metod.

V aplikacích numerických derivací a integrací v rovině jsem znovu ukázal generování funkce a její derivaci v MATLABu. Pak jsem vypočetl základní vlastnosti funkce, jako je její délka, obsah pod funkcí, střední a efektivní hodnotu, objem tělesa, moment setrvačnosti a těžiště. Zde by se práce dala rozšířit o výpočet dalšího libovolného parametru funkce.

U části zabývající se řešením nelineárních rovnic pomocí postupné aproximace jsem nastínil princip její a následné použití pro řešení rovnic, které nelze řešit analyticky. Tato část bylo z mého pohledu nejobtížnější, protože popsat její princip bylo složité. Nelineární rovnice jdou řešit i jiným způsobem, o to by se práce dala zesílit. Dále jsem chtěl program rozšířit o výpočet chyby u určité rovnice, ale k tomu jsem se nedostal.

Poslední kapitola se zabývá rozložením libovolného signálu pomocí Fourierova rozvoje. V této části jsem definoval Fourierův rozvoj a pomocí trigonometrické řady, kterou převedl do numerické podoby, jsem popsal funkci sinus. Samozřejmě pomocí programu lze rozložit libovolnou funkci. Toto téma bych označil za uzavřené.

Všechny programy jsou univerzální, tzv. jdou použít pro libovolnou funkci. Program MATLAB je velice podobný programovacímu jazyku C, lze všechny algoritmy do tohoto programovacího jazyka implementovat. Nemám v plánu v práci pokračovat, protože není dostatek času, ale nepopírám, že se k ní někdy vrátím.

Použitá literatura:

1. Bartsch, H.-K.: Matematické vzorce. ACADEMIA. Praha, 2006.

2. Fišer, K., Moravec, Z., Novotný, D.: Matematika pro fyziky. 2. vydání. UJEP. Ústí nad Labem, 2009.