Numerické metody řešení rovnice $F(x) = 0$

Miroslava Jarešová

Obsah

Úvod

Při řešení mnoha úloh z matematiky, z fyziky i dalších oborů se stává, že dospějeme k rovnici, s jejímž řešením „si nevíme rady“. Může se to stát z několika důvodů: neznáme vzorec nebo úpravu příslušné rovnice anebo také proto, že takovou rovnici nelze pomocí jakékoliv úpravy řešit.

S rovnicemi, které nedokážeme přesně řešit, se v praxi setkáváme mnohem více než s těmi, kde se přesné řešení dá najít.

Rovnice, kde nedokážeme nalézt přesné řešení, jsou nejčastěji buď rovnice algebraické $F(x) = 0$, kde $F(x)$ je mnohočlen n-tého stupně, tj.

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = 0,$$

nebo transcendentní, což jsou nealgebraické rovnice (goniometrické, logaritmické atd.).

Při řešení těchto rovnic je vhodné nejprve udělat grafický odhad kořene – nejčastěji postupujeme tak, že si narýsujeme graf funkce $F(x)$ (nejlépe použít postupu pro vyšetřování průběhu funkce pomocí diferenciálního počtu), kořeny rovnice pak jsou souřadnice průsečíků s osou x.

V předložené práci je tento krok nahrazen počítačovým zpracováním – vykreslením grafu zadané funkce. Grafy je možno vykreslit pomocí programů Famulus, Pascal, Mathematica a DELPHI. Práce je zpracovaná tak, že v programech Famulus a Pascal je nutno zadat funkci již přímo do zdrojového kódu, v programu Mathematica je možno funkci vkládat po spuštění patřičného souboru vytvořeného pomocí tohoto programu, v prostředí DELPHI se podařilo vyřešit problém zadávání funkce tím způsobem, že funkce se vkládá za běhu programu do patřičného okna.

Graf vykreslujeme tak, abychom mohli provést separaci kořenů,tj. hledáme separační interval, ve kterém se nachází právě jeden z kořenů nelineární rovnice. Při separaci využíváme *Bolzanovy – Weierstrassovy věty*:

Jestliže pro $a < b$ platí $F(a) \cdot F(b) < 0$, pak existuje alespoň jedno číslo $x_0 \in (a, b)$ tak, že $F(x) = 0$.

Postačující podmínka pro separaci kořene je: jestliže pro $a, b \in R$, $a < b$ platí $F(a) \cdot F(b) < 0$ a $F'(x) \neq 0$ pro všechna $x \in (a, b)$, pak má rovnice $F(x) = 0$ v intervalu (a, b) právě jeden kořen.

Máme-li provedenu separaci kořenů, můžeme začít řešit rovnici za pomoci níže uvedených numerických metod, které si popíšeme.

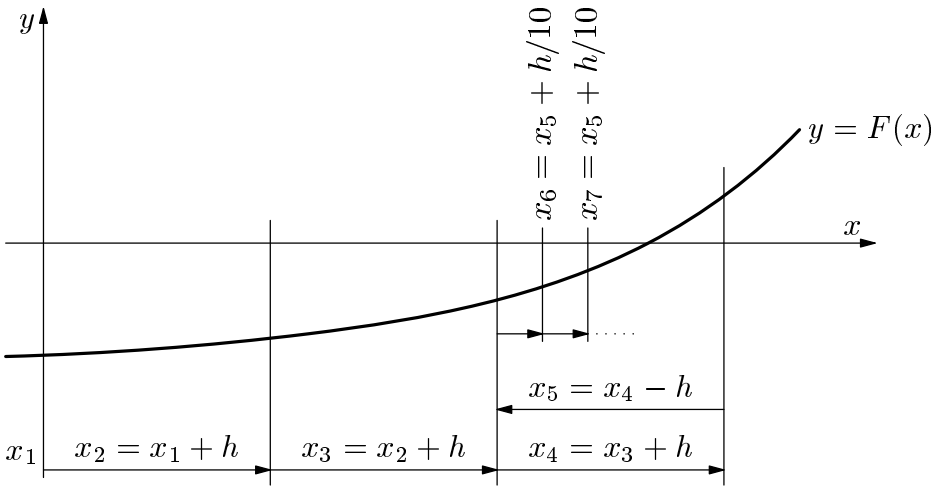
1 Popis jednotlivých metod

Nejprve si ukážeme metodu, která je velmi jednoduchá, avšak velmi zdlouhavá. K řešení dojdeme až po velmi vysokém počtu kroků. V této metodě není nutné provádět separaci kořenů.

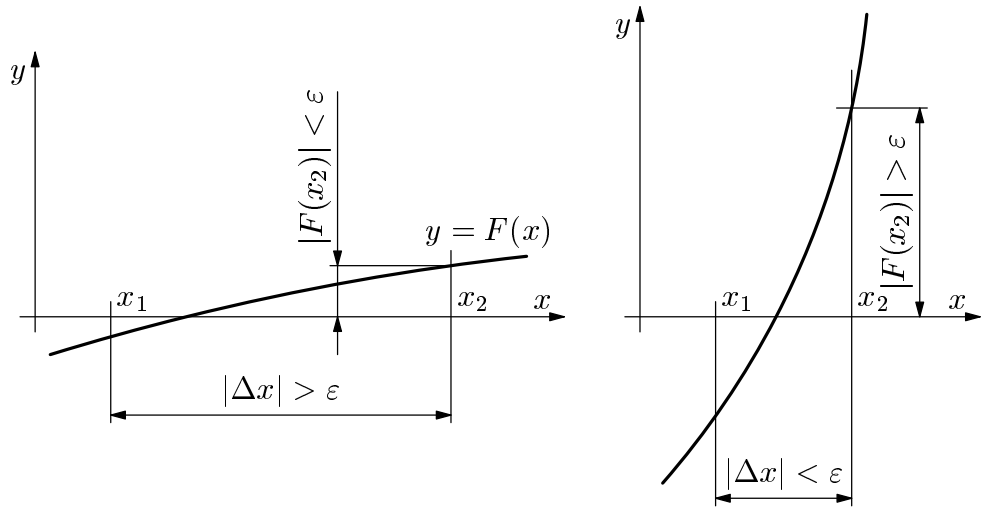
1.1 Metoda postupného přiblížení

Při určování kořene se začíná libovolně zvoleným krokem x a vhodně zvoleným krokem h. Nejdříve se pro zadané x určí hodnota $F(x)$, určí se její znaménko, zvětší x o krok h a znovu určí hodnota $F(x)$. Tento postup se opakuje tak dlouho, pokud se neurčí $F(x)$ s opačným znaménkem. V tomto případě mezi dvěma posledními souřadnicemi leží kořen rovnice. Vrátíme se o krok zpět a krok zmenšíme, např. $h' = \frac{h}{10}$. Dále se zvětší x o nový krok h' a znovu určí hodnota $F(x)$. Postup je opakován, dokud není určen kořen s dostatečnou přesností, tzn. dokud neplatí $|x_i - x_{i+1}| < \varepsilon$ a $|F(x_{i+1})| < \varepsilon$, viz obr. 1.

Je vhodné provádět oba testy, neboť ve většině případů neznáme charakter funkce. Význam testů je zřejmý z obr. 2a, b, které ukazují nedostatečnost použití jen jednoho testu.



Obr. 1 Metoda postupným přibližováním



Obr. 2a, b Testování kořene

1.2 Metoda půlení intervalu (bisekce)

Tato metoda je jednou z nejjednodušších metod pro určení kořene algebraické nebo trancendentní rovnice $F(x) = 0$ v intervalu $\langle x_0, x_1\rangle$, kde $F(x)$ je funkce spojitá v intervalu $\langle x_0, x_1\rangle$ a $F(x_0) \cdot F(x_1) < 0$. Interval $\langle x_0, x_1\rangle$ rozdělíme na dvě stejně velké části $\langle x_0, x_2\rangle$, $\langle x_2, x_1\rangle$, kde $x_2 = \frac{x_0 + x_1}{2}$. Jestliže $|x_1 - x_0| < \varepsilon$, pak x_2 je aproximace kořene, jinak $\langle x_0, x_2\rangle$ nebo $\langle x_2, x_1\rangle$ obsahuje kořen. Pro $F(x_0) \cdot F(x_2) < 0$ je kořen v oblasti $\langle x_0, x_2\rangle$, jinak je v oblasti $\langle x_2, x_1\rangle$. Tento interval je znovu rozpůlen a celý proces se opakuje. Pro test splnění přesnosti odhadu kořene je opět jako v předchozím případě vhodné testovat obě podmínky na ose x i y.

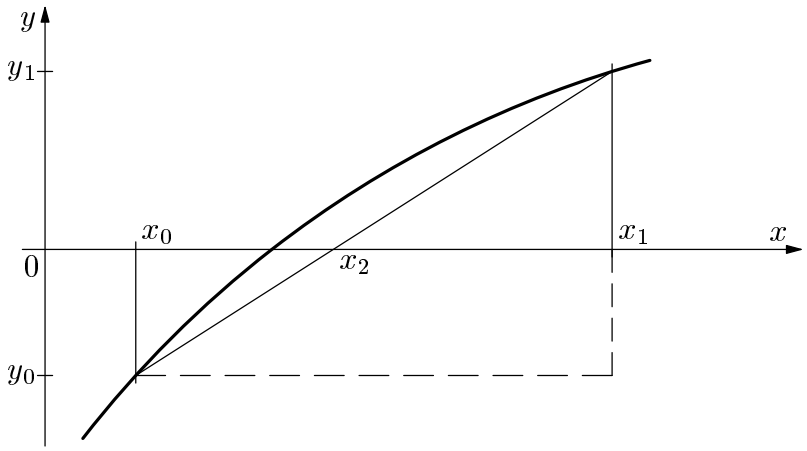
Nevýhodou metody je velký počet kroků, než je nalezen odhad kořene s předepsanou přesností. Výhodou je jednoduchá a snadná realizovatelnost na počítači a je vhodné ji použít, jsou-li počítačové informace o poloze kořene chudé. V takovém případě určíme dva body, ve kterých má funkce $F(x)$ opačné znaménko, a aplikujeme metodu. Tato metoda konverguje pro libovolnou spojitou funkci.

Algoritmus:

Určíme střed intervalu $x_2 = \frac{x_0 + x_1}{2}$. Pokud $F(x_2) = 0$, je úloha vyřešena. Jinak pokračujeme hledáním kořenu v té polovině intervalu, kde má funkce v krajních bodech opačná znaménka. Postupné zmenšování intervalu opakujeme, dokud absolutní hodnota funkce uprostřed intervalu neklesne pod předem zvolenou mez.

1.3 Metoda sečen

Jednotlivé kroky této metody spočívají v lineární interpolaci intervalu (x_0, x_1), která je znázorněna na obr. 3.



Obr. 3 Metoda sečen

Z podobnosti trojúhelníků odvodíme:

$$\frac{x_2 - x_0}{0 - y_0} = \frac{x_1 - x_0}{y_1 - y_0},$$

$$x_2 = x_0 - y_0 \frac{x_1 - x_0}{y_1 - y_0} = \frac{y_1 x_0 - y_0 x_1}{y_1 - y_0}.$$

Pokud $F(x_2) = 0$, je rovnice vyřešena, jinak pokračujeme hledáním kořenu v té části intervalu, kde má funkce v krajních bodech opačné znaménko, podobně jako v metodě půlení intervalu.

1.4 Metoda tečen (Newtonova metoda)

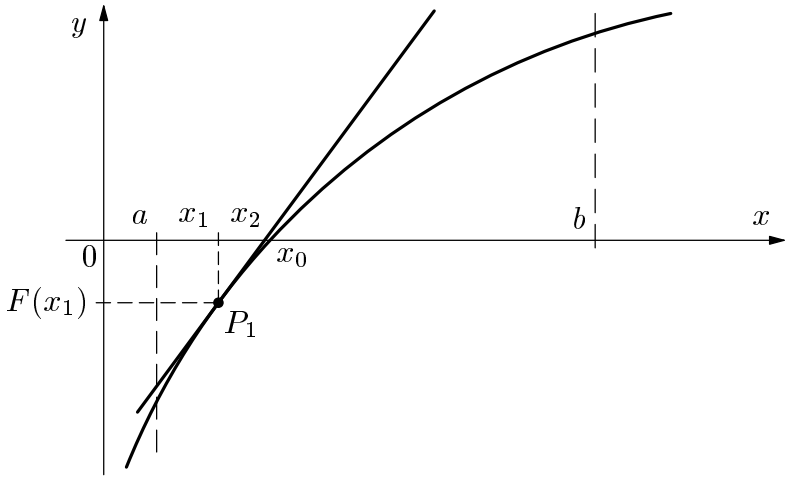
Jestliže v rovnici $F(x) = 0$ je F funkcí, která má na intervalu $\langle a, b\rangle$ spojitou druhou derivaci, $F(a) \cdot F(b) < 0$, derivace F' a F'' jsou na intervalu $\langle a, b\rangle$ různé od nuly a $x_1 \in \langle a, b\rangle$ je takový bod, že $F(x_1) \cdot F''(x_1) > 0$, pak první aproximací kořene rovnice $F(x) = 0$ je bod

$$x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}.$$

Po výpočtu aproximace x_2 kořene lze metody tečen znovu použít pro nalezení lepší aproximace kořene. Potom při každém dalším kroku se počet správných číslic kořene prakticky zdvojnásobí.

Geometrický význam metody tečen:

Křivka se nahradí tečnou v bodě $P_1 = [x_1, f(x_1)]$ (z toho plyne název metoda tečen).



Obr. 4 Metoda tečen

Nyní si ukážeme metodu, která je velmi jednoduchá, ale nemusí vždy konvergovat – metoda iterační. Podívejme se nyní na tuto metodu podrobněji.

1.5 Metoda iterační

Při řešení rovnice $F(x) = 0$ iterační metodou se řešení rovnice nahradí některou rovnocennou rovnicí tvaru

$$x = \varphi(x)$$

a posloupnost aproximací se vytváří takto: zvolí se počáteční aproximace x_0 a další aproximace se počítají pomocí rekurentního předpisu

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \ldots$$

Iterační metoda konverguje, jestliže

$$\lim_{k \to \infty} x_k = a,$$

kde a je hledaný kořen rovnice $x = \varphi(x)$ a též tedy kořen výchozí rovnice $F(x) = 0$. Funkce $\varphi = \varphi(x)$ je tzv. iterační funkce. Konvergence metody a pracnost nezbytných výpočtů závisejí na vhodné volbě iterační funkce φ a na počáteční aproximaci x_0[1].

[1]K tomu, abychom dosazováním do rovnice $x = \varphi(x)$ získávali vždy lepší přibližné hodnoty (aproximace) je třeba (a stačí), aby úhel α tečny ke křivce $y = \varphi(x)$ a osou x byl $0 < \alpha < 45°$ anebo $135° < \alpha < 180°$, tj. aby $|\operatorname{tg} \alpha| < 1$. Pokud není tato podmínka splněna, nelze tuto metodu použít. Podmínku $|\operatorname{tg} \alpha| < 1$ lze zapsat užitím derivace funkce $\varphi(x)$ také takto $|\varphi'(x)| < 1$ pro všechna $x \in (a, b)$, kde je splněna separační podmínka $F(a) \cdot F(b) < 0$. Výše uvedené vztahy platí za předpokladu, že φ je v intervalu $\langle a, b \rangle$ spojitá a zobrazuje jej do sebe, tj. $\forall x \in \langle a, b \rangle$ je též $\varphi(x) \in \langle a, b \rangle$ a derivace $\varphi'(x)$ je spojitá v (a, b). V případě $|\varphi'(x)| > 1 \ \forall x \in (a, b)$ lze tuto funkci nahradit funkcí inverzní $\varphi^{-1}(x)$. Problémy mohou nastávat, pokud je v okolí kořene $|\varphi'(x)|$ jen o málo menší než jedna. Pak by se mohlo stát, že po sobě jdoucí aproximace se málo liší a přitom jsou daleko od kořene. V tomto případě je vhodné buď zvolit jinou iterační funkci nebo jinou numerickou metodu.

Vzhledem k tomu, že iterační metoda nemusí vždy konvergovat, ukážeme si její použití pouze za pomoci programu Mathematica, kde je to nejrychlejší a nejjednodušší. Budeme sledovat konvergenci funkce v závislosti na počtu bodů.

Příklad 1

Řešte rovnici

$$F(x) = \frac{x}{2} + \sin(x) - 2 = 0.$$

Řešení

Z rovnice vyjádříme neznámou x:

$$x = 4 - 2\sin(x).$$

Zvolíme

$$\varphi(x) = 4 - 2\sin(x),$$

potom

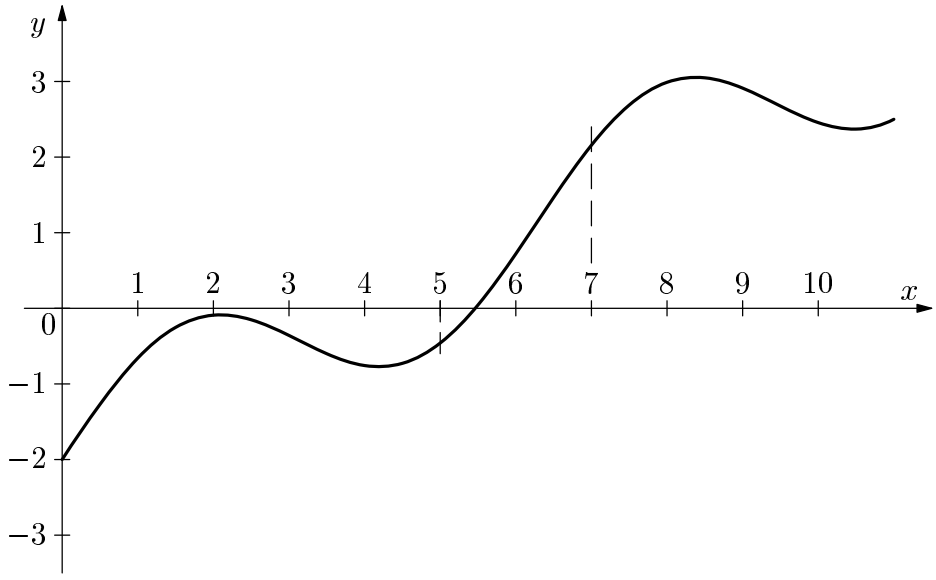
$$\varphi'(x) = -\cos(x).$$

Zvolíme separační interval $x \in (5; 7)$ (viz obr. 1), potom

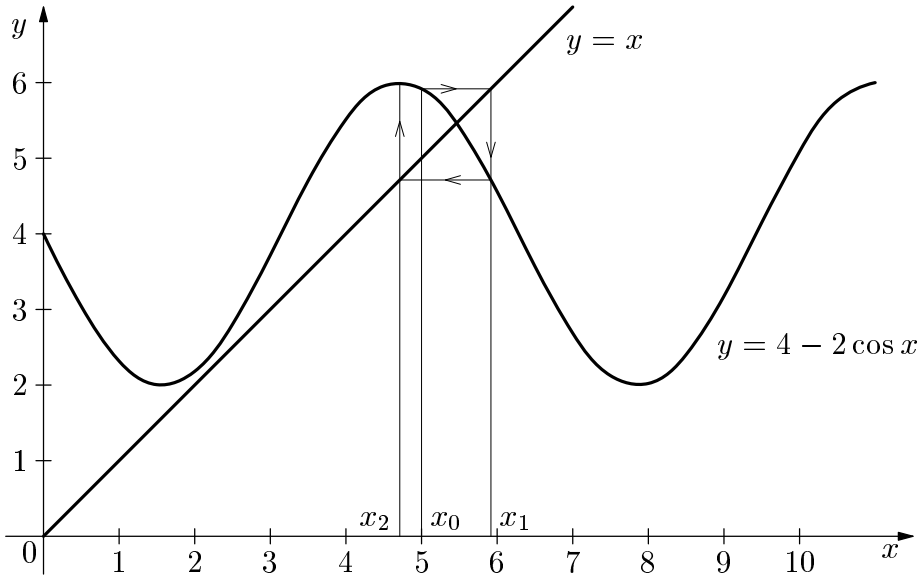
$$-2\cos 5 = -0{,}567,$$

$$-2\cos 7 = -1{,}507.$$

Není splněna podmínka $|\varphi'(x)| < 1$ pro $x \in (5; 7)$, iterační metodu nelze v tomto případě použít (viz obr. 6).



Obr. 5 Graf funkce $F(x) = \frac{x}{2} + \sin(x) - 2$



Obr. 6 Metoda iterační – iterační funkce

V případě obr. 6 je vidět, že použití iterační metody nikdy nepovede k nalezení hledaného kořene rovnice (dostáváme se mimo interval $(5; 7)$, kde by měl být kořen).

Příklad 2

Je dána rovnice $x^3 - 4x + 2 = 0$. Ukažte, že iterační metoda je použitelná v případě volby iterační funkce $\varphi(x) = \dfrac{x^3 + 2}{4}$ dané rovnice pouze při hledání jednoho kořene.

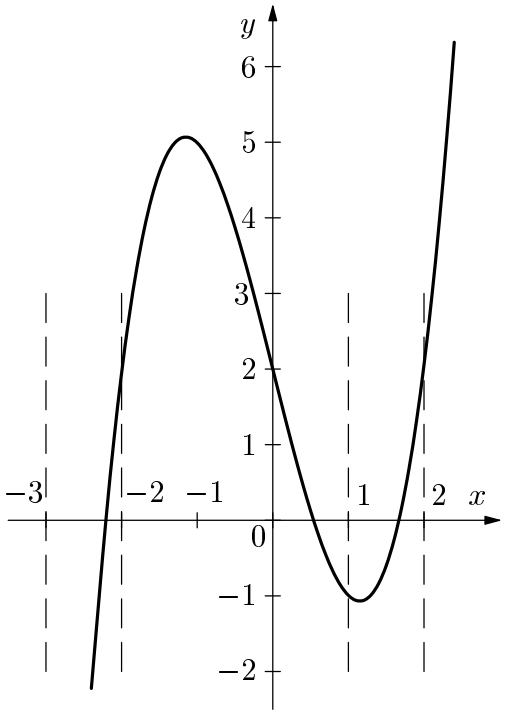
Řešení

Určíme separační intervaly (viz obr. 7): $\varphi(x) = \dfrac{x^3 + 2}{4}$, $\varphi'(x) = \dfrac{3x^2}{4}$.

a) $x \in (0; 1)$: $\quad 0 < \varphi'(x) < \dfrac{3}{4}$ $\quad$ metodu lze použít,

b) $x \in (-3; -2)$: $\quad 3 < \varphi'(x) < \dfrac{27}{4}$ $\quad$ metodu nelze použít,

c) $x \in (1; 2)$: $\quad \dfrac{3}{4} < \varphi'(x) < 3$ $\quad$ metodu nelze použít.

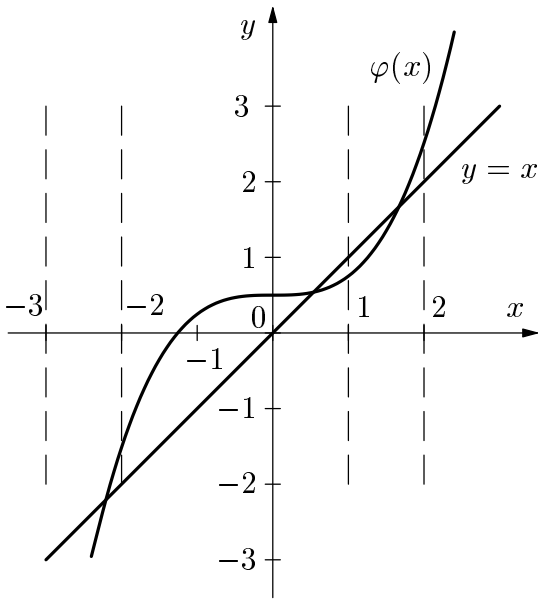


Obr. 7 Metoda iterační – graf funkce $F(x) = x^3 - 4x + 2$

Metoda konverguje pouze v případě kořene z intervalu $(0; 1)$, v ostatních dvou případech nelze kořeny hledat pomocí iterační metody, protože tato metoda v těchto případech diverguje. Tento závěr je možno potvrdit i pomocí grafického řešení – postup by byl obdobný jako v příkladu 1.

Příklad 2 ukazuje, že ani vhodně zvolená funkce $\varphi(x)$ není zárukou konvergence metody (stačí několik různých intervalů a vše je jinak), v tomto případě metoda konverguje pouze v intervalu $(0; 1)$, v ostatních případech diverguje. To se dá předem zjistit pomocí podmínky

$$|\varphi'(x)| = |\operatorname{tg}\alpha| < 1.$$



Obr. 8 Metoda iterační – iterační funkce

Řešení pomocí programu Mathematica:

```
Iteracni[(x^3+2)/4,{x,0},10]
1.krok - koren= 0.5
2.krok - koren= 0.53125
3.krok - koren= 0.537483
4.krok - koren= 0.538818
5.krok - koren= 0.539108
6.krok - koren= 0.539171
7.krok - koren= 0.539185
8.krok - koren= 0.539188
9.krok - koren= 0.539189
10.krok - koren= 0.539189

Iteracni[(x^3+2)/4,{x,0},4]
1.krok - koren= -6.25
2.krok - koren= -60.5352
3.krok - koren= -55457.3
```

Poznámka:

Pokud bychom v příkladu 2 zvolili jinou iterační funkci, např.

$$\varphi(x) = \sqrt[3]{4x - 2},$$

potom $\varphi'(x) = \dfrac{4}{3}\dfrac{1}{\sqrt[3]{(4x-2)^2}}$.

Tato derivace $\varphi'(x)$ není spojitá v bodě $x_0 = \frac{1}{2}$. Nelze proto volit separační interval, který by obsahoval $x = \frac{1}{2}$ (i když se jeden z kořenů nachází v okolí tohoto bodu, jak víme z předchozího postupu).

Pomocí programu Mathematica bychom zjistili, že ani pro tuto iterační funkci metoda nekonverguje – tentokrát ani v jednom případě. Po několika iteracích se dostáváme velmi rychle mimo separační interval.

Vhodná volba iterační funkce se ukazuje v celé řadě případů jako problematická (viz poznámka 1 pod čarou str. 6). Z tohoto důvodu se ukazuje jako vhodnější používat pro počítačové zpracování jiné – „spolehlivější“, před tím popsané metody.

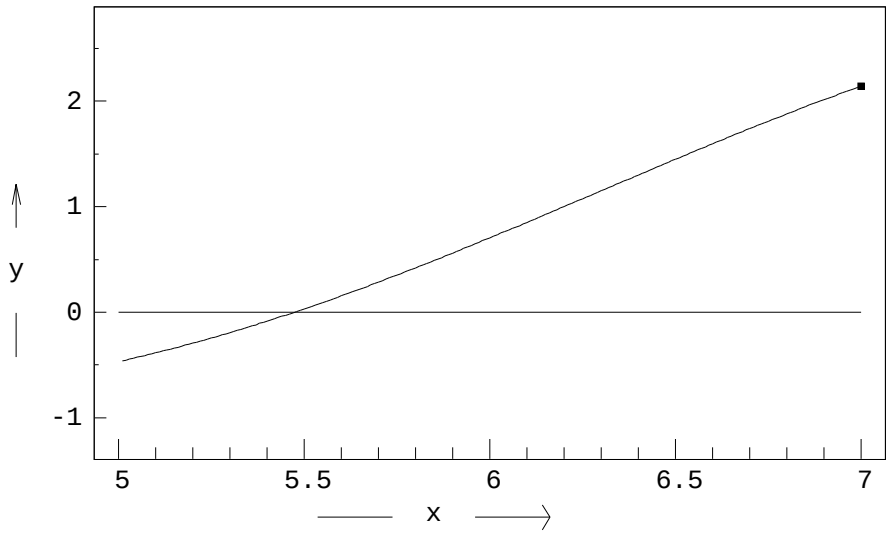
2 Zpracování metod užitím programu Famulus

Před použitím níže uvedených metod je nutné provést separaci kořenů. K tomu použijeme níže vytvořený program ve Famulovi, pomocí kterého vykreslíme průběh funkce $F(x)$. Vykreslení grafu funkce si ukážeme při řešení rovnice

$$F(x) = \frac{x}{2} + \sin x - 2 = 0.$$

Model ve Famulovi bude mít tvar:

```
Vykresleni grafu funkce k metodam reseni rovnice f(x) = 0

- - - - - proměnné, konstanty, procedury a funkce  - - - - - - -
FUNCTION f(x)=x/2+sin(x)-2
- - - - - - - - - - -  počáteční hodnoty   - - - - - - - - - - -
xmin=5 ! pocatecni hodnota intervalu
xmax=7 ! koncova hodnota intervalu
y1=f(xmin) ! pocatecni funkcni hodnota
y2=f(xmax) ! koncova funkcni hodnota
dx=0.01 ! krok
x=xmin ! pocatecni funkcni hodnota
- - - - - - - - - - - - - - -  model   - - - - - - - - - - - - - - -
y=f(x)
 IF x>xmax THEN STOP END
x=x+dx
```



Obr. 9 Graf funkce $F(x) = \frac{x}{2} + \sin x - 2$

Po provedení separace, tj. určení intervalu $(a; b)$, kde platí $F(a) \cdot F(b) < 0$ a zároveň $\forall x \in (a; b)\ F'(x) \neq 0$, můžeme řešit rovnici $F(x) = 0$ pomocí níže popsaných metod.

2.1 Metoda postupného přiblížení

Model ve Famulovi má tvar:

```
                    Metoda postupneho priblizeni.

Hledany koren aproximuje promenna x na konci cyklu.
- - - - - - - proměnné, konstanty, procedury a funkce  - - - - - - -
FUNCTION f(x)=x/2+sin(x)-2
- - - - - - - - - - - - -  počáteční hodnoty   - - - - - - - - - - - -
xmin=5 ! pocatecni hodnota intervalu
k=0 ! pocet kroku
y=f(xmin) ! pocatecni funkcni hodnota
h=0.1 ! pocatecni hodnota kroku
x=xmin;
- - - - - - - - - - - - - - - - -  model   - - - - - - - - - - - - - - - -
k=k+1
 IF y=0 THEN STOP END;
 IF f(xmin)*y>0 THEN x=x+h; y=f(x);
   IF abs(y)<1e-7 AND h<1e-7 THEN STOP END
                              ELSE x=x-h;y=f(x);h=h/10;END
```

k	x	y
1	5.100000	-0.375815
2	5.200000	-0.283455
3	5.300000	-0.182267
4	5.400000	-0.072764
5	5.500000	0.044460
6	5.400000	-0.072764
7	5.410000	-0.061379
8	5.420000	-0.049917
9	5.430000	-0.038379
10	5.440000	-0.026765
11	5.450000	-0.015077
12	5.460000	-0.003315
13	5.470000	0.008520
14	5.460000	-0.003315
15	5.461000	-0.002135
16	5.462000	-0.000954
17	5.463000	0.000228
18	5.462000	-0.000954
19	5.462100	-0.000836
20	5.462200	-0.000718
21	5.462300	-0.000600
22	5.462400	-0.000481
23	5.462500	-0.000363
24	5.462600	-0.000245
25	5.462700	-0.000127
26	5.462800	-0.000009
27	5.462900	0.000110
28	5.462800	-0.000009
29	5.462810	0.000003
30	5.462800	-0.000009
31	5.462801	-0.000007
32	5.462802	-0.000006
33	5.462803	-0.000005
34	5.462804	-0.000004
35	5.462805	-0.000003
36	5.462806	-0.000002
37	5.462807	-0.000000
38	5.462808	0.000001
39	5.462807	-0.000000
40	5.462807	-0.000000
41	5.462807	-0.000000

Tab. 1 Průběh hledání polohy kořene v závislosti na počtu iterací – metodou postupného přiblížení

2.2 Metoda půlení intervalu (bisekce)

Model ve Famulovi má tvar:

```
                    Metoda puleni intervalu (bisekce)

Hledani nuloveho bodu funkce  y = x/2+sin(x)-2 v intervalu (5,7)
- - - - - - - proměnné, konstanty, procedury a funkce  - - - - - - -
FUNCTION f(x) = x/2+sin(x) - 2
- - - - - - - - - - - - -  počáteční hodnoty   - - - - - - - - - - - -
xmin=5  ! zacatek intervalu
xmax=7  ! konec intervalu
y0=f(xmin)
y1=f(xmax)
IF y0*y1>0 THEN
  WRITELN "V koncovych bodech nema funkce ruzna znamenka !"
  STOP
END
k=0          ! ... pocet iteraci
- - - - - - - - - - - - - - - - -  model   - - - - - - - - - - - - - - - -
 x2 = (xmin+xmax)/2
 y2 = f(x2)
 IF y0*y2>0
   THEN  xmin=x2; y0=y2
   ELSE  xmax=x2; y1=y2
 END
 IF abs(xmax-xmin)<1e-7 AND abs(y2)<1e-7 THEN STOP END
k=k+1
```

k	x2	y2
1	6.000000	0.720585
2	5.500000	0.044460
3	5.250000	-0.233934
4	5.375000	-0.100889
5	5.437500	-0.029676
6	5.468750	0.007037
7	5.453125	-0.011410
8	5.460938	-0.002209
9	5.464844	0.002408
10	5.462891	0.000098
11	5.461914	-0.001055
12	5.462402	-0.000479
13	5.462646	-0.000190
14	5.462769	-0.000046
15	5.462830	0.000026
16	5.462799	-0.000010
17	5.462814	0.000008
18	5.462807	-0.000001
19	5.462811	0.000004
20	5.462809	0.000002
21	5.462808	0.000000
22	5.462807	-0.000000
23	5.462807	0.000000
24	5.462807	-0.000000

Tab. 2 Průběh hledání polohy kořene v závislosti na počtu iterací – metodou půlení intervalu

2.3 Metoda sečen

Model ve Famulovi má tvar:

```
                              Metoda secen

Reseni rovnice sin(x)+x/2-2=0 v intervalu <5,7>
- - - - - - - proměnné, konstanty, procedury a funkce  - - - - - - -
FUNCTION f(x)=sin(x)+x/2-2
- - - - - - - - - - - - -  počáteční hodnoty   - - - - - - - - - - - -
xmin=5  ! zacatek intervalu
xmax=7  ! konec intervalu
y0=f(xmin)
y1=f(xmax)
IF y0*y1>0 THEN
  WRITELN "V koncovych bodech ma funkce stejna znamenka."
  STOP
END
k=0          !  pocet iteraci
- - - - - - - - - - - - - - - - -  model   - - - - - - - - - - - - - - -
 x2=xmin-(xmax-xmin)/(y1-y0)*y0
 y2=f(x2)
 xmin=xmax ;y0=y1
 xmax=x2 ;y1=y2
 IF abs(xmax-xmin)<1e-7 AND abs(y2)<1e-7 THEN STOP END
k=k+1
```

k	x2	y2
1	5.350871	-0.127565
2	5.442956	-0.023318
3	5.463554	0.000882
4	5.462803	-0.000005
5	5.462807	-0.000000

Tab. 3 Průběh hledání polohy kořene v závislosti na počtu iterací – metodou sečen

2.4 Metoda tečen

Model ve Famulovi má tvar:

```
                              Metoda tecen

Reseni rovnice  y = sin(x)+x/2-2=0 v intervalu <5,7>
- - - - - - - proměnné, konstanty, procedury a funkce  - - - - - - -
FUNCTION f(x) = sin(x)+x/2-2
FUNCTION f1(x) = cos(x)+1/2     ! derivace funkce f(x)
- - - - - - - - - - - - -  počáteční hodnoty   - - - - - - - - - - - -
x1=6          ! libovolny bod intervalu <5,7>
k=0           ! pocet iteraci
- - - - - - - - - - - - - - - - -  model   - - - - - - - - - - - - - - -
x0=x1
y1=f(x1)
y1d=f1(x1)
x1=x1-y1/y1d
IF abs(x1-x0)<1e-7 AND abs(y1)<1e-7 THEN STOP END
k=k+1
```

k	x1	y1
1	5.506507	0.720585
2	5.463367	0.052339
3	5.462807	0.000662
4	5.462807	0.000000

Tab. 4 Průběh hledání polohy kořene v závislosti na počtu iterací – metodou tečen

Poznámka:

Metoda postupným přibližováním je zde uvedena jen jako ilustrační, vzhledem k vysokému počtu kroků vedoucích k řešení, z tohoto důvodu nebude již dále uváděna.

Metoda iterační zde není uvedena z důvodů „nespolehlivé“ konvergence – viz závěr 1. kapitoly, příklady 1 a 2.

3 Zpracování metod pomocí programu Pascal

3.1 Grafické provedení separace

Pomocí níže uvedeného programu je provedena separace. Program potom vyřeší danou rovnici metodou půlení intervalu.

Před spuštěním programu je třeba na začátku programu za deklarační částí zadat funkci $f(x)$ – viz řádek „zadani funkce“.

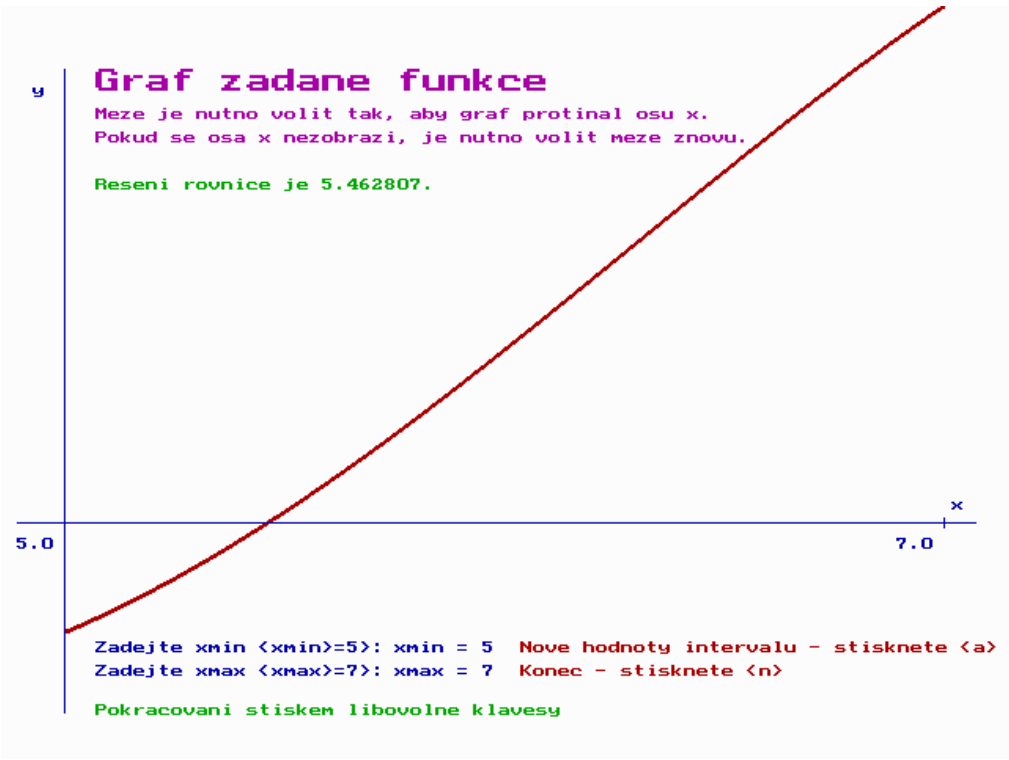
Výpis zdrojového kódu programu:

```
Program GrafFunkce;
uses Crt,Graph;
var xp,yp: array[1..600] of real;
    i,m,n,Graphmode,Graphdriver,h,k:integer;
    xminz, xmaxz,xmin,xmax,ymin,ymax,x,y,eps,xres:real;
    xmins,xmaxs,xress:String[10];
    Zn: char;
function f(x:real):real;
begin
  f:=x/2+sin(x)-2;  {zadani funkce}
end;

{Nacitani vstupnich hodnot v grafickem modu}
Procedure VstupReal(Popis:string;var cislo:real);
 var pom:string[30];
     znak:char;
     chyba,i,j,px,py:integer;
 begin
  j:=Length(Popis);px:=GetX;py:=GetY;
  repeat
     pom:='';chyba:=0;
     for i:=1 to 30 do pom:=pom+#219;
     setcolor(white);outtextxy(pX,pY,pom);pom:='';
     setcolor(blue);outtextxy(pX,pY,Popis);
     MoveTo(pX+8*j,pY);
     repeat znak:=readkey;
            if znak<>#13 then pom:=pom+znak;
            outtextxy(GetX,GetY,pom)
     until znak=#13;
     if pom<>'' then  val(pom,cislo,chyba)
  until Chyba=0;
  if pom='' then cislo:=1e32;
end{VstupReal};

procedure Bisekce(xmini,xmaxi,epsi:real;
                  var x0,y0:real; var k1:integer);
 var y1,y2:real;
```

```
 begin
  k1:=0;
  if f(xmini)*f(xmaxi)>0 then begin
   OutTextXY
    (60,110,'Neni splnena podminka f(Xmin) * f(Xmax) < 0,
             zvolte jiny interval.');
   exit;
                              end
                 else
 repeat
  y1:=f(xmini);
  y2:=f(xmaxi);
  x0:=(xmini+xmaxi)/2;
  y0:=f(x0);
  if y1*y0>0 then begin xmini:=x0; y1:=y0; end
             else begin xmaxi:=x0; y2:=y0; end;
  k1:=k1+1;
 until (abs(xmaxi-xmini)<epsi) and (abs(y0)<epsi);
end;

begin
  ClrScr;
  Zn:=' ';
 repeat
  Graphdriver:=Detect;InitGraph(Graphdriver,Graphmode,' ');
  SetBkColor(white); {barva pozadi}
  m:=GetMaxX-80;n:=GetMaxY-80;
  SetTextStyle(0,0,2);
  SetColor(Magenta);      {barva nadpisu}
  OutTextXY(60,40,'Graf zadane funkce ');
  SetTextStyle(0,0,1);
  OutTextXY
   (60,65,'Meze je nutno volit tak, aby graf protinal osu x.');
  OutTextXY
   (60,80,'Pokud se osa x nezobrazi, je nutno volit meze znovu.');
  SetTextStyle(0,0,1);
  SetColor(Blue);
  OutTextXY(60,405,'Zadejte xmin <xmin>=5>: ');
  MoveTo(250,405);VstupReal('xmin = ',xmin);
  OutTextXY(60,420,'Zadejte xmax <xmax>=7>: ');
  MoveTo(250,420);VstupReal('xmax = ',xmax);
  xminz:=xmin;xmaxz:=xmax;
  SetColor(Red);  {barva grafu}
  for i:=1 to m do
   begin
     x:=xmin+(xmax-xmin)*i/m;
     y:=f(x); xp[i]:=x;yp[i]:=y;
  end;
```

```
  ymin:=yp[1];
  ymax:=yp[1];
  for i:=2 to m do
    if yp[i] > ymax then ymax:=yp[i];
  for i:=2 to m do
    if yp[i] < ymin then ymin:=yp[i];
  SetLineStyle(0,0,3);  {tloustka cary grafu}
  MoveTo(40+1,trunc(n*(f(xmin+(xmax-xmin)/m)-ymax)/(ymin-ymax)));
  for i:=1 to m do begin   {vykresleni grafu}
   x:=xmin+(xmax-xmin)*i/m; y:=f(x);
   lineTo(40+i,trunc(n *(y-ymax)/(ymin-ymax)));
                    end;
  SetLineStyle(0,0,1);
  SetColor(Blue);   {barva os}
  h:=trunc(-n*ymax/(ymin-ymax));
  Line(10,h,m+60,h);  {osa x}
  Line(40,40,40,450); {osa y}
  Line(40+m,h-3,40+m,h+3);
  Str(xmax:2:1,xmaxs);
  Str(xmin:2:1,xmins);
  SetColor(Blue);
  OutTextXY(10,h+10,xmins);
  OutTextXY(m+10,h+10,xmaxs);
  OutTextXY(m+45,h-15,'x');
  OutTextXY(20,50,'y');
  SetColor(green);
  eps:=1e-7;
  Bisekce(xminz,xmaxz,eps,xres,y,k); {reseni}
  Str(xres:8:6,xress);
  OutTextXY(60,445,'Pokracovani stiskem libovolne klavesy');
  repeat   until keypressed;
  if f(xmin)*f(xmax)<0 then
           OutTextXY(60,110,'Reseni rovnice je '+ xress +'.');
  SetColor(red);
  Outtextxy(330,405,'Nove hodnoty intervalu - stisknete <a>');
  Outtextxy(330,420,'Konec - stisknete <n> ');
  Repeat Zn:=ReadKey  until Zn in ['N','n','A','a'];
 until
  (Zn='n') or (Zn='N');
 CloseGraph;
end.
```

Výpis běhu programu:



Obr. 10 Graf funkce $F(x) = \frac{x}{2} + \sin x - 2$ v Pascalu

3.2 Řešení rovnice $F(x) = 0$ s volbou jednotlivých metod

Před spuštěním níže uvedeného programu je nutno provést separaci – pomocí programu **GrafFunkce**, který je uveden výše.

Rovnice $F(x) = 0$ je řešena metodou půlení intervalu, metodou sečen a tečen.

Před spuštěním programu je třeba do řádku „zadani funkce“ zadat funkci a do řádku „derivace funkce“ zadat její derivaci v případě použití metody tečen. Pro porovnání metod je u každé metody uveden také počet iterací potřebných k získání kořene s požadovanou přesností.

Výpis zdrojového kódu programu:

```
Program Rovnice;
uses Crt;
var xmin,xmax,x,y,eps:real;
    k,u:integer;
    Zn:char;
```

```
function f(x:real):real;
begin
  f:=x/2+sin(x)-2;  {zadani funkce}
end;

function fd(x:real):real;
begin
  fd:=cos(x)+1/2;  {derivace funkce}
end;

procedure Bisekce(xmini,xmaxi,epsi:real;
                  var x0,y0:real; var k1:integer);
 var y1,y2:real;
 begin
  k1:=0;
 repeat
  y1:=f(xmini);
  y2:=f(xmaxi);
  x0:=(xmini+xmaxi)/2;
  y0:=f(x0);
  if y1*y0>0 then begin xmini:=x0; y1:=y0; end
             else begin xmaxi:=x0; y2:=y0; end;
  k1:=k1+1;
 until (abs(xmaxi-xmini)<epsi) and (abs(y0)<epsi);
end;

procedure Secny(xmini,xmaxi,epsi:real;
                var x0,y0:real; var k1:integer);
 var y1,y2:real;
 begin
  k1:=0;
 repeat
  y1:=f(xmini);
  y2:=f(xmaxi);
  x0:=xmini-(xmaxi-xmini)/(y2-y1)*y1;
  y0:=f(x0);
  xmini:=xmaxi; y1:=y2;
  xmaxi:=x0;y2:=f(x0);
  k1:=k1+1;
 until (abs(xmaxi-xmini)<epsi) and (abs(y0)<epsi);
end;

procedure Tecny(xmini,xmaxi,epsi:real;
          var x0,y1:real; var k1:integer);
 var h,y1d,x3:real;
 begin
  k1:=0;
  x0:=(xmini+xmaxi)/2;
```

```
 repeat
  xmini:=x0;
  y1:=f(xmini);
  y1d:=fd(xmini);
  x0:=x0-y1/y1d;
  k1:=k1+1;
 until (abs(x0-xmini)<epsi) and (abs(y1)<epsi);
end;

begin
 ClrScr;
 writeln('              Reseni rovnice f(x) = 0');
 write('     Zadejte dolni mez intervalu xmin = (5) ');
 readln(xmin);
 write('     Zadejte horni mez intervalu xmax = (7) ');
 readln(xmax);
 write('     Zadejte pozadovanou presnost eps = (1e-7) ');
 readln(eps);
 writeln('     Vyberte metodu vypoctu: ');
 writeln('     1 - Bisekce     2 - Secny    3 - Tecny');
 Zn:=' ';
repeat
 writeln('---------');
 write('  Cislo metody = ');readln(u);
  case u of
   1: Bisekce(xmin,xmax,eps,x,y,k);
   2: Secny(xmin,xmax,eps,x,y,k);
   3: Tecny(xmin,xmax,eps,x,y,k);
  end;
 writeln('Reseni rovnice je ',x:8:6);
 writeln('Pocet iteraci je ',k-1);
 writeln('Chcete pouzit jinou metodu? a/n ');
 Repeat Zn:=ReadKey  until Zn in ['N','n','A','a'];
 until
  (Zn='n') or (Zn='N');
end.
```

Výpis běhu programu:

```
                    Reseni rovnice f(x) = 0
    Zadejte dolni mez intervalu xmin = (5) 5
    Zadejte horni mez intervalu xmax = (7) 7
    Zadejte pozadovanou presnost eps = (1e-7) 1e-7
    Vyberte metodu vypoctu:
    1 - Bisekce     2 - Secny    3 - Tecny
---------
   Cislo metody = 1
```

```
Reseni rovnice je 5.462807
Pocet iteraci je 24
Chcete pouzit jinou metodu? a/n

---------
   Cislo metody = 2
Reseni rovnice je 5.462807
Pocet iteraci je 5
Chcete pouzit jinou metodu? a/n

---------
   Cislo metody = 3
Reseni rovnice je 5.462807
Pocet iteraci je 4
Chcete pouzit jinou metodu? a/n
```

4 Zpracování metod užitím programu Mathematica

Zde jsou uvedeny zdrojové kódy v programu Mathematica. S výstupním požadavkem požadované přesnosti kořene jsou zde zpracovány metody půlením intervalu, sečen a tečen (Newtonova). Programy dále vypisují počty iterací potřebné k dosažení požadované přesnosti.

```
Bisekce[f_,{x_,a_,b_},eps_]:=
    Module[{x2=N[a]-N[b], k=0, xmin=N[a], xmax=N[b]},
          If[(f/.x->xmin) (f/.x->xmax)>0,
                   Print[" Zadany interval nesplnuje podminku
                   f(a).f(b)<0"],
           While[(f/.x->xmin) (f/.x->xmax)<0  && Abs[xmax-xmin]>eps
                  && Abs[f/.x->x2]>eps,
                    x2=(xmin+xmax)/2;
                    k=k+1;
                    If[(f/.x->xmin) (f/.x->x2)<0, xmax=x2, xmin=x2]
                ];
                 Print["Koren je v bode ", x2];
                 Print["Pocet iteraci je ", k]
            ]
          ]

Secna[f_,{x_,a_,b_},eps_]:=
    Module[{x2=N[b]-N[a], k=0, xmin=N[a], xmax=N[b]},
          If[(f/.x->xmin)*(f/.x->xmax)>0,
                   Print[" Zadany interval nesplnuje podminku
                   f(a).f(b)<0"],
          While[(f/.x->xmin)*(f/.x->xmax)<0  && Abs[xmax-xmin]>eps
                  && Abs[f/.x->x2]>eps,
                   x2=xmin-(xmax - xmin)/
                      ((f/.x->xmax) - (f/.x->xmin))*(f/.x->xmin);
                   k=k+1;
                   If[(f/.x->xmin)*(f/.x->x2)<0, xmax=x2,xmin=x2]
                ];
                 Print["Koren je v bode ", x2];
                 Print["Pocet iteraci je ", k]
             ]
          ]


Newton[f_,{x_,a_,b_},eps_]:=
    Module[{k=0, x0=0, x1=(N[a]+N[b])/2,fd=D[f,x]},
           While[Abs[x1-x0]>eps && Abs[f/.x->x1]>eps,
                    x0=x1;
```

```
                x1=x1-(f/.x->x1)/(fd/.x->x1);
                k=k+1;
            ];
            Print["Koren je v bode ", x1];
            Print["Pocet iteraci je ", k]
        ]
```

Dále jsou zde uvedeny metody půlení intervalu a iterační, které mají jako výstupní požadavek zadání počtu kroků, v průběhu výpočtu jsou vypisovány jednotlivé iterace.

```
PuleniInt[f_,{x_,a_,b_},n_]:=
    Module[{x2, k=0, xmin=N[a], xmax=N[b]},
          If[(f/.x->xmin) (f/.x->xmax)>0,
                   Print[" Zadany interval nesplnuje podminku
                   f(a).f(b)<0"],
           While[(f/.x->xmin) (f/.x->xmax)<0  && k<n,
    Print[" Interval, ve kterem je koren:  <",xmin," , ", xmax,">"];
                    x2=(xmin+xmax)/2;
                    k=k+1;
                    If[(f/.x->xmin) (f/.x->x2)<0, xmax=x2, xmin=x2]
                ];
              If[(f/.x->xmin) == 0, Print["Koren je v bode ",xmin],
                If[(f/.x->xmax) == 0, Print["Koren je v bode ",xmax],
                                        Print["Koren je v bode ",x2]
                  ]
                ]
            ]
          ]
```

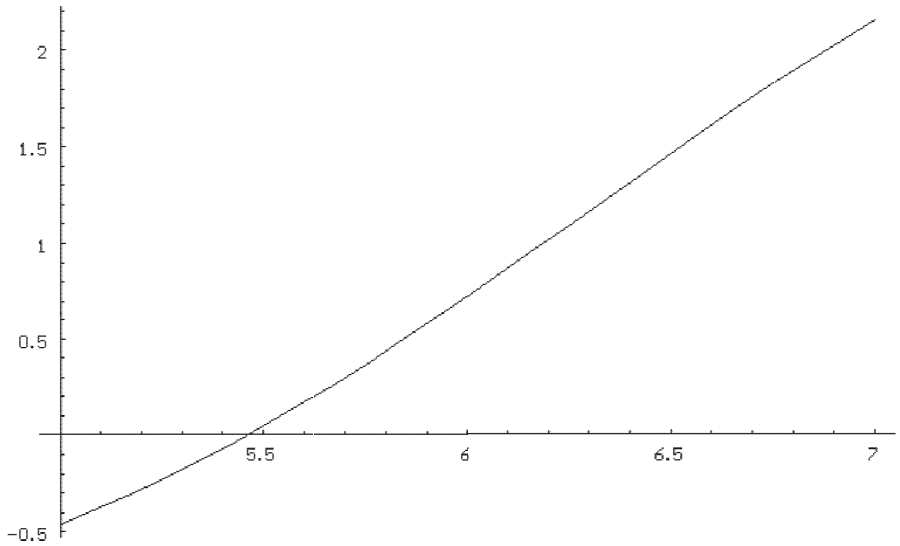
```
Iteracni[fi_,{x_,x0_},n_]:=
    Module[{k=0, xx=N[x0]},
           While[k<n,
                    k=k+1;
                    xx=fi/.x->xx; Print[k,".krok - koren= ",N[xx]];
                ]
          ]
```

Před spuštěním výše uvedených programů je třeba provést separaci, a to tak, že si nejprve vykresíme graf zadané funkce. Na ukázku zde opět hledáme řešení rovnice $F(x) = \frac{x}{2} + \sin x + 2 = 0$. Nakreslením grafu funkce $F(x)$ opět můžeme určit vhodný interval $(5; 7)$.

Výpis běhu programu:

```
Mathematica 2.2 for DOS 387
Copyright 1988-93 Wolfram Research, Inc.

In[1]:= Plot[x/2+Sin[x]-2,{x,5,7}]
Out[1]= -Graphics-

In[2]:= << MathRov.m

In[3]:= Bisekce[x/2+Sin[x]-2,{x,5,7},0.0000001]
Koren je v bode 5.46281
Pocet iteraci je 24

In[4]:= Newton[x/2+Sin[x]-2,{x,5,7},0.0000001]
Koren je v bode 5.46281
Pocet iteraci je 4
```

Graf získaný pomocí příkazu Plot[x/2+Sin[x]-2,x,5,7]:



Obr. 11 Graf funkce $F(x) = \frac{x}{2} + \sin x - 2$ pomocí programu Mathematica

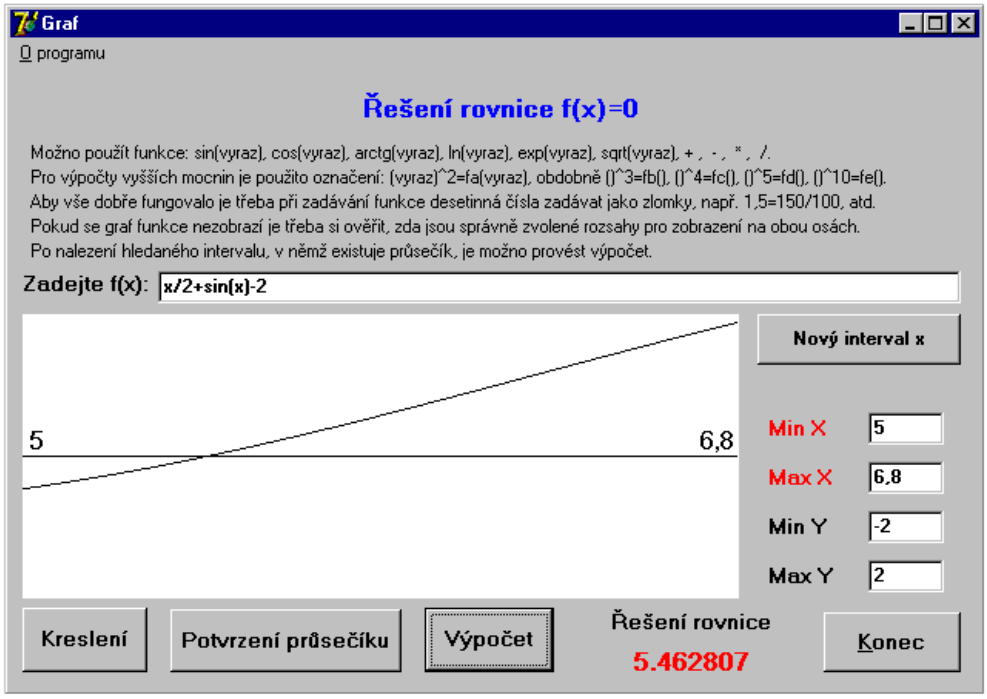
5 Řešení rovnice $F(x) = 0$ v prostředí DELPHI

Program vytvořený v prostředí DELPHI v sobě zahrnuje vše najednou – separaci i řešení rovnice. K jeho spuštění není třeba mít na počítači nainstalováno DELPHI, vytvořený exe soubor ke své činnosti potřebuje pouze prostředí Windows.

Po spuštění programu je třeba nejprve zadat funkci a zvolit interval pro x. Meze pro y jsou předefinovány, ale je možno je měnit – při změně mezí pro y je ale nutné zachovat jejich symetrii vzhledem k ose x. [2] Po této volbě je nutno stisknout tlačítko kreslení.

Pokud nalezneme interval, kde existuje průsečík, je třeba tento průsečík potvrdit, v opačném případě je nutno zadat jiný interval stiskem tlačítka **Nový interval x**.

Program je ošetřen i pro případ potvrzení neexistujícího průsečíku – nedovolí provést výpočet. Po stisku tlačítka **Výpočet** se vypíše řešení rovnice – zde je použita metoda půlení intervalu, která se v tomto případě jeví jako nejspolehlivější. V případě existence většího počtu řešení dané rovnice je možno postupnou volbou separačních intervalů zjistit všechna řešení rovnice (přesnost je nastavena na 6 desetinných míst).



Obr. 11 Řešení rovnice $\frac{x}{2} + \sin x - 2$ v prostředí DELPHI

[2] Jiný přístup je zvolen při kreslení osy x v Pascalu, každý z těchto způsobů má své výhody i nevýhody. V obou případech je třeba brát v úvahu tu skutečnost, že kreslení grafů slouží v tomto případě především ke správnému provedení separace.

Výpis zdrojového kódu programu v DELPHI

```
unit fce;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls, Funk, Menus;

type
  TGraf = class(TForm)
    EFunkce: TEdit;
    Kresleni: TButton;
    EXmin: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    EXmax: TEdit;
    EYmin: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    EYmax: TEdit;
    NInterval: TButton;
    Label5: TLabel;
    Label7: TLabel;
    Vypocet: TButton;
    Label9: TLabel;
    Konec: TButton;
    Label6: TLabel;
    Label8: TLabel;
    Reseni: TLabel;
    Label10: TLabel;
    Prusec: TButton;
    Label11: TLabel;
    Obrazek: TImage;
    Label12: TLabel;
    Menu: TMainMenu;
    Oprogramu1: TMenuItem;
    procedure KresleniClick(Sender: TObject);
    procedure NIntervalClick(Sender: TObject);
    procedure VypocetClick(Sender: TObject);
    procedure KonecClick(Sender: TObject);
    procedure PrusecClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure Oprogramu1Click(Sender: TObject);
```

```
   private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Graf: TGraf;
  Xmin1, Xmax1:extended;
implementation

uses Unit1;

{$R *.DFM}

type
   typ = extended;
var
   x:     typ;
   Funkce: string;
   i:    integer;

procedure TGraf.FormCloseQuery(Sender: TObject;
                               var CanClose: Boolean);
begin
 CanClose:=MessageDlg('Opravdu chcete skončit?',
           mtConfirmation,[mbYes,mbNo],0)=mrYes;
end;

procedure TGraf.KonecClick(Sender: TObject);
begin
  Close;
end;

procedure Chyba( s: string );
begin
   ShowMessage( s );
   Abort
end;

procedure Dalsi;
begin
   repeat
      Inc(i)
   until Funkce[i]<>' '
end;
```

```
function Cislo: typ;
var x: typ;
begin
   Result := 0;
   while Funkce[i] in ['0'..'9'] do
   begin
      Result := 10*Result + ord( Funkce[i] )-ord( '0' );
      Dalsi
   end;

   if Funkce[i]<>'.' then exit;

   { za desetinnou tečkou: }
   x := 0.1;

   while Funkce[i] in ['0'..'9'] do
   begin
      Result := Result + x*( ord( Funkce[i] )-ord( '0' ) );
      x := x/10;
      Dalsi
   end
end;

function Text: string;
begin
   Result := '';
   while Funkce[i] in ['a'..'z'] do
   begin
      Result := Result + Funkce[i];
      Dalsi
   end
end;

function Term: typ;     forward;
function Faktor: typ;   forward;

function Vyraz: typ;
var T: typ;
    zn: char;
begin
   T := Term;

   while Funkce[i] in ['+','-'] do
   begin
      zn := Funkce[i];
      Dalsi;
      case zn of
      '+': T := T+Term;
```

```
      '-': T := T-Term;
      end { case }
   end; { while }

   Vyraz := T
end;

function Term: typ;
const  MAX = 9999;
var FF: typ;
    zn: char;
begin
   FF := Faktor;

   while Funkce[i] in ['*','/'] do
   begin
      zn := Funkce[i];
      Dalsi;
      case zn of
      '*': FF := FF*Faktor;
      '/': try
             FF := FF/Faktor;
           except
             if FF > 0 then FF := MAX
                       else
             if FF < 0 then FF := -MAX
                       else FF := 0
           end
      end { case }
   end; { while }

   Term := FF
end;

function Faktor: typ;
var  Nazev: string;

begin
   case Funkce[i] of
   '0'..'9': Faktor := Cislo;
   'a'..'z': { promenna nebo funkce }
             begin
                Nazev := Text;
                if Nazev = 'x' then
                begin
                   Faktor := x;
                end
                              else
```

```
                { funkce: }
                begin
                   if Funkce[i]<>'(' then Chyba('Chybí "("')
                                     else Dalsi;
                   if Nazev = 'sin' then Faktor := sin(Vyraz)
                                    else
                   if Nazev = 'cos' then Faktor := cos(Vyraz)
                                    else
                   if Nazev = 'arctg' then
               try Faktor := arctan(Vyraz) except Faktor := 0 end
                                     else
                   if Nazev = 'ln' then
               try Faktor := ln(Vyraz) except Faktor := 0 end
                                    else
                   if Nazev = 'exp' then
               try Faktor := exp(Vyraz) except Faktor := 0 end
                                    else
                   if Nazev = 'fa' then
               try Faktor := fa(vyraz) except Faktor := 0 end
                                    else
                   if Nazev = 'fb' then
               try Faktor := fb(Vyraz) except Faktor := 0 end
                                    else
                   if Nazev = 'fc' then
               try Faktor := fc(Vyraz) except Faktor := 0 end
                                    else
                    if Nazev = 'fd' then
               try Faktor := fd(Vyraz) except Faktor := 0 end
                                    else
                    if Nazev = 'fe' then
               try Faktor := fe(Vyraz) except Faktor := 0 end
                                    else
                   if Nazev = 'sqrt' then
               try Faktor := sqrt(Vyraz) except Faktor := 0 end
                     else Chyba( 'Neznama funkce "'+Nazev+'"' );
                   if Funkce[i]<>')' then Chyba('Chybí")"')
                                     else Dalsi
                end
             end;
   '(':      begin
                Dalsi;
                Faktor := Vyraz;
                if Funkce[i]<>')' then Chyba( 'Chybí ")"' )
                                  else Dalsi
             end
   end { case }
end;
```

```
function f(ax: typ):typ;
begin
   x := ax;
   i := 1;
   f := Vyraz
end;

procedure TGraf.KresleniClick(Sender: TObject);
var  i,j, imax,jmax: integer;
     x,y:extended;
     Xmin,Xmax, Ymin,Ymax: extended;
 begin
   try
      Xmin := StrToFloat( EXmin.Text );
      Xmax := StrToFloat( EXmax.Text );
      Ymin := StrToFloat( EYmin.Text );
      Ymax := StrToFloat( EYmax.Text );
      except
      Chyba( 'Spatne zadane rozsahy' );
     end;
   Obrazek.Visible:=true;
   Vypocet.Visible:=false;
   Funkce := LowerCase( EFunkce.Text )+'#';
   imax := Obrazek.Width-1;
   jmax := Obrazek.Height-1;
   { vymazat: }
   Obrazek.Canvas.Pen.Style := psClear;
   Obrazek.Canvas.Rectangle(0,0,imax+1,jmax+1 );
   Obrazek.Canvas.Pen.Style := psSolid;
   Obrazek.Canvas.MoveTo(0,(jmax+1) div 2); (*osa x*)
   for i:=0 to imax do
   Obrazek.Canvas.LineTo(i,(jmax+1) div 2);
   Obrazek.Canvas.Font.Size:=12;
   Obrazek.Canvas.TextOut(5,(jmax-40) div 2,EXmin.Text);
   Obrazek.Canvas.TextOut(i-25,(jmax-40) div 2,EXmax.Text);
   for i:=0 to imax do              (*kresleni grafu*)
    begin
      x := Xmin + i*(Xmax-Xmin)/imax;
      y := f(x);
      j := round(jmax*(1-(y-Ymin)/(Ymax-Ymin)));
      if i=0 then Obrazek.Canvas.MoveTo(i,j)
             else Obrazek.Canvas.LineTo(i,j)
    end
end;

procedure TGraf.NIntervalClick(Sender: TObject);
begin
 EXmin.SetFocus;
```

```
 EXmin.text:='';
 EXmax.text:='';
 Obrazek.Visible:=false;
 Reseni.Visible:=false;
 Vypocet.Visible:=false;

end;

procedure TGraf.VypocetClick(Sender: TObject);
var  x0, y0, y1, y2,h: extended;
     Xmin,Xmax, Ymin,Ymax: extended;
     x0s:string;
begin
   try
      Xmin := StrToFloat( EXmin.Text );
      Xmax := StrToFloat( EXmax.Text );
      Ymin := StrToFloat( EYmin.Text );
      Ymax := StrToFloat( EYmax.Text );
   except
      Chyba( 'Špatně zadané rozsahy' );
   end;
  Funkce := LowerCase( EFunkce.Text )+'#';
           (*metoda bisekce*)
 repeat
  y1:=f(Xmin);
  y2:=f(Xmax);
  x0:=(Xmax+Xmin)/2;
  y0:=f(x0);
  if y1*y0>0 then begin Xmin:=x0;y1:=y0; end
             else begin Xmax:=x0;y2:=y0; end;
 until (Abs(Xmax-Xmin)<1e-7) and (Abs(y0)<1e-7);
 Reseni.visible:=true;
 Str(x0:8:6,x0s);
 Reseni.caption:=x0s;
end;


procedure TGraf.PrusecClick(Sender: TObject);
var Xmax1,Xmin1:typ;
begin
 Xmin1 := StrToFloat( EXmin.Text );
 Xmax1 := StrToFloat( EXmax.Text );
 Funkce := LowerCase( EFunkce.Text )+'#';
 If f(Xmin1)*f(Xmax1)>0 then begin label7.Visible:=false;
   ShowMessage
   ('Není splněna podmínka f(Xmin) * f(Xmax) < 0.
   Zadejte nový interval nebo ukončete práci.');
                          exit;
```

```
                              end
                         else
               begin
 label7.Visible:=true;
 Vypocet.visible:=true;
               end;
end;

procedure TGraf.Oprogramu1Click(Sender: TObject);
begin
 Form1.ShowModal;
end;
end.
```

Výpis zdrojového kódu programové jednotky **unit1.pas**

```
Unit Funk;
  interface
    type
     typ=extended;
    var X:typ;
        N:integer;
    function fa(Y:typ):typ;
    function fb(Y:typ):typ;
    function fc(Y:typ):typ;
    function fd(Y:typ):typ;
    function fe(Y:typ):typ;
   implementation

    function fa(Y:typ):typ;  (*vyraz na druhou*)
      var V: typ;
         I: integer;
       begin
        V:=1;
        for I:=1 to 2 do V:=V*Y;
        fa:=V;
       end;

    function fb(Y:typ):typ;  (*vyraz na 3*)
      var V: typ;
         I: integer;
       begin
        V:=1;
        for I:=1 to 3 do V:=V*Y;
        fb:=V;
       end;
```

```
     function fc(Y:typ):typ;  (*vyraz na 4*)
      var V: typ;
         I: integer;
       begin
        V:=1;
        for I:=1 to 4 do V:=V*Y;
        fc:=V;
       end;

    function fd(Y:typ):typ;  (*vyraz na 5*)
      var V: typ;
         I: integer;
       begin
        V:=1;
        for I:=1 to 5 do V:=V*Y;
        fd:=V;
       end;

    function fe(Y:typ):typ;  (*vyraz na 10*)
      var V: typ;
         I: integer;
       begin
        V:=1;
        for I:=1 to 10 do V:=V*Y;
        fe:=V;
       end;

end.
```

Výpis zdrojového kódu programové jednotky **unit1.pas**

```
unit unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);

  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
 close;
end;

end.
```

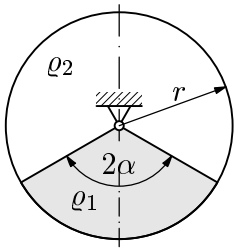
6 Ukázky fyzikálních úloh vedoucích k řešení transcendentních rovnic

Příklad 1 – kyvadlo

Kyvadlo je tvořeno válcovým kotoučem o poloměru $r = 0{,}100$ m, složeným ze dvou výsečí o hustotách $\varrho_1 = 7\,800\ \mathrm{kg\cdot m^{-3}}$, $\varrho_2 = 2\,700\ \mathrm{kg\cdot m^{-3}}$ ($\varrho_1 > \varrho_2$). Středový úhel výseče o hustotě ϱ_1 je 2α (obr. 12). Kyvadlo rozkmitáme s malou amplitudou úhlové výchylky.

a) Určete obecně periodu kmitů $T = T(r, \varrho_1, \varrho_2, \alpha)$.

b) Pro dané hodnoty r, ϱ_1 a ϱ_2 určete středový úhel 2α, při kterém je perioda kmitů minimální, a vypočtěte tuto periodu $T_{\min}$. Transcendentní rovnici, kterou dostanete, řešte pomocí některé z numerických metod.



Obr. 12

Těžiště kruhové výseče o poloměru r se středovým úhlem 2α leží ve vzdálenosti $\dfrac{2r\sin\alpha}{3\alpha}$ od středu kružnice.

Řešení

a) K určení periody harmonických kmitů musíme vypočítat moment setrvačnosti kotouče J a direkční moment D (k tomu je třeba určit těžiště kruhové výseče).

Moment setrvačnosti válce je $J = \dfrac{1}{2}mr^2 = \dfrac{1}{2}\varrho h\pi r^4$, kde h je výška válce. Výseče mají momenty setrvačnosti:

$$J_1 = \frac{1}{2}\varrho_1 h\pi r^4\frac{2\alpha}{2\pi} = \frac{1}{2}\varrho_1 h r^4\alpha, \qquad J_2 = \frac{1}{2}\varrho_2 h\pi r^4\frac{2\pi-2\alpha}{2\pi} = \frac{1}{2}\varrho_2 h r^4(\pi-\alpha).$$

Moment setrvačnosti kotouče $J = J_1 + J_2 = \dfrac{1}{2}hr^4[\varrho_1\alpha + \varrho_2(\pi-\alpha)]$. vztah pro výpočet polohy těžiště kruhové výseče

$$y_{\mathrm{T}} = -\frac{2}{3}\frac{r\sin\alpha}{\alpha},$$

je možno odvodit pomocí integrálního počtu nebo ho lze nalézt v tabulkách.

První výseč: $y_{\mathrm{T}1} = -\dfrac{2}{3}\dfrac{\sin\alpha}{\alpha}r$. Druhá výseč: $y_{\mathrm{T}2} = \dfrac{2}{3}\dfrac{\sin\alpha}{\pi-\alpha}r$.

Direkční moment: $D = m_1 g|y_{\mathrm{T}1}| - m_2 g y_{\mathrm{T}2}$,

$$D = \frac{\pi r^2 2\alpha}{2\pi}h\varrho_1 g\frac{2\sin\alpha}{3\alpha}r - \frac{\pi r^2(2\pi-2\alpha)}{2\pi}h\varrho_2 g\frac{2\sin\alpha}{3(\pi-\alpha)}r$$

$$D = \frac{2}{3}r^3 hg(\varrho_1-\varrho_2)\sin\alpha.$$

Doba kmitu

$$T = 2\pi\sqrt{\frac{J}{D}} = 2\pi\sqrt{\frac{3r}{4g}\frac{\varrho_1\alpha + \varrho_2(\pi-\alpha)}{(\varrho_1-\varrho_2)\sin\alpha}}$$

$$T = 2\pi\sqrt{\frac{3r}{4g}\left(\frac{\alpha}{\sin\alpha} + \frac{\pi}{\left(\frac{\varrho_1}{\varrho_2}-1\right)\sin\alpha}\right)}.$$

Označme $\xi = \frac{\varrho_1}{\varrho_2} - 1$, pak $T = 2\pi\sqrt{\frac{3r}{4g}\left(\frac{\alpha}{\sin\alpha} + \frac{\pi}{\xi\sin\alpha}\right)}$.

b) Aby bylo T minimální, musí být minimální výraz pod odmocninou, tj. budeme hledat minimum funkce $f(\alpha) = \frac{\alpha}{\sin\alpha} + \frac{\pi}{\xi\sin\alpha}$:

$$\frac{\mathrm{d}f(\alpha)}{\mathrm{d}\alpha} = \frac{\mathrm{d}}{\mathrm{d}\alpha}\left(\frac{\alpha}{\sin\alpha} + \frac{\pi}{\xi\sin\alpha}\right) = \frac{1}{\sin^2\alpha}\left(\sin\alpha - \alpha\cos\alpha - \frac{\pi}{\xi}\cos\alpha\right) = 0,$$

$$\frac{\alpha}{\sin\alpha} + \frac{\pi}{\xi\sin\alpha} = 0, \qquad \mathrm{tg}\,\alpha - \alpha - \frac{\pi}{\xi} = 0,$$

což je transcendentní rovnice, kterou vyřešíme pouze pro dané hodnoty:

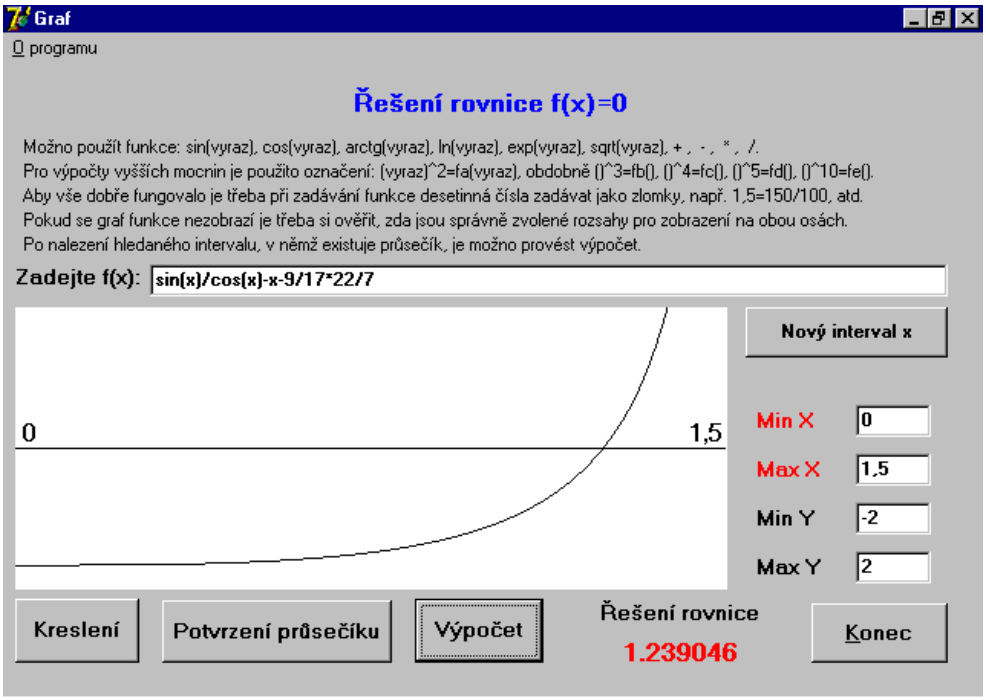
$$\xi = \frac{\varrho_1}{\varrho_2} - 1 = \frac{17}{9}, \qquad \mathrm{tg}\,\alpha - \alpha - \frac{9\pi}{17} = 0.$$

Tuto rovnici vyřešíme užitím programu v DELPHI: $\alpha = 1{,}24$ rad $= 71°$.
Potom je $\alpha_1 = 2\alpha = 142°$, $\alpha_2 = 360° - \alpha_1 = 218°$.
O tom, zda se jedná o lokální minimum se přesvědčíme pomocí druhé derivace funkce $f(\alpha)$:

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\left[\frac{1}{\sin^2\alpha}\left(\sin\alpha - \alpha\cos\alpha - \frac{\pi}{\xi}\right)\right] =$$

$$= \frac{1}{\sin\alpha}\left(\alpha + \frac{\pi}{\xi}\right) - \frac{2\cos\alpha}{\sin^3\alpha}\left(\sin\alpha - \alpha\cos\alpha - \frac{\pi}{\xi}\cos\alpha\right).$$

Pro $\alpha = 1{,}24$ rad je $\frac{\mathrm{d}^2 f(\alpha)}{\mathrm{d}\alpha^2} = 4{,}63 > 0$, nastává lokální minimum.
$T_{\min} \doteq 0{,}95$ s.



Obr. 13 Řešení transcendentní rovnice $\operatorname{tg}\alpha - \alpha - \dfrac{9\pi}{17} = 0$ v prostředí DELPHI

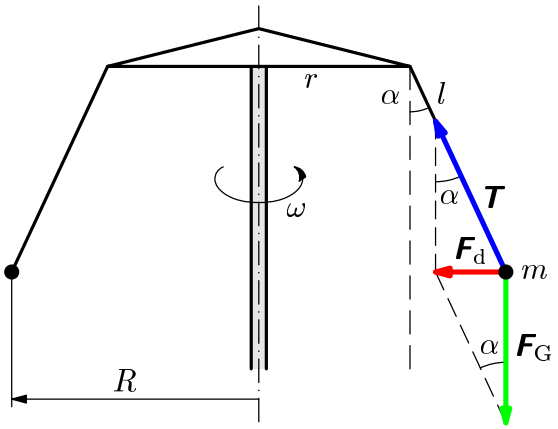
Poznámka:

Hledání lokálního minima za pomoci druhé derivace je možno nahradit použitím programů vykreslujících průběh funkce zpracovaných v této práci.

Příklad 2 – řetízkový kolotoč

Řetízkový kolotoč se otáčí úhlovou rychlostí $\omega = 1\ \mathrm{rad}\cdot\mathrm{s}^{-1}$. Poloměr ramene, na němž je sedačka zavěšena, je $r = 3$ m, délka závěsu je $l = 6$ m. Určete úhel α, který svírá závěs sedačky kolotoče se svislým směrem.

Řešení



Obr. 14 Řetízkový kolotoč

Úlohu budeme řešit v inerciální vztažné soustavě souřadnic spojené s povrchem Země. Na sedačku působí tíhová síla a tahová síla závěsu. Výslednicí těchto sil je dostředivá síla. Platí

$$F_\mathrm{d} = F_\mathrm{G}\operatorname{tg}\alpha,$$

$$m\omega^2 R = mg\operatorname{tg}\alpha,$$

kde $R = r + l\sin\alpha$.

Po dosazení za R dostaneme $\omega^2(r + l\sin\alpha) = g\dfrac{\sin\alpha}{\cos\alpha}$,

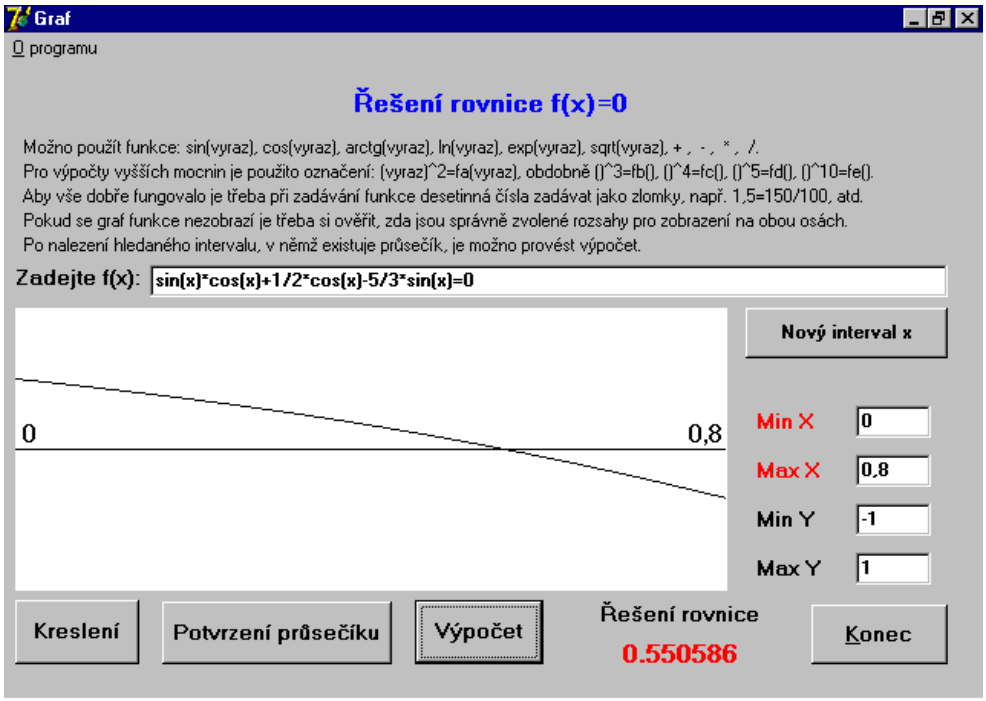
po dalších úpravách postupně dostáváme

$$\frac{\sin\alpha}{\cos\alpha} = \frac{\omega^2}{g}(r + l\sin\alpha),$$

$$\omega^2 \sin\alpha\cos\alpha + \frac{r}{l}\omega^2 - \frac{g}{l}\sin\alpha = 0.$$

Po dosazení zadaných hodnot dostaneme rovnici o proměnné α:

$$\sin\alpha\cos\alpha + \frac{1}{2}\cos\alpha - \frac{5}{3}\sin\alpha = 0.$$



Obr. 15 Řešení transcendentní rovnice $\sin\alpha\cos\alpha + \frac{1}{2}\cos\alpha - \frac{5}{3}\sin\alpha = 0$

Dostali jsme $\alpha = 0{,}550586$ rad $\doteq 31{,}5°$. Závěs sedačky svírá při pohybu úhlovou rychlostí $\omega = 1\ \mathrm{rad}\cdot\mathrm{s}^{-1}$ se svislou osou úhel 31,5°.

Úloh vedoucích k řešení transcendentních a složitých rovnic existuje velké množství, další zde už neuvádím vzhledem k omezenému rozsahu a hlavní náplni této práce, což je naprogramování základních numerických metod.

Závěr – porovnání metod a použitých programů

Porovnání použitých programů

Práce se zabývá numerickým řešením rovnice $F(x) = 0$. V průběhu práce jsou postupně zpracovány různé metody, jak řešit tuto rovnici.

Ve všech případech jsou uvedeny programy kreslící graf funkce z důvodů usnadnění separace.

Zpracování metod pomocí programu Famulus je přehledné, rychlé – bohužel se tento program vzhledem k tomu, že pracuje pod DOSem a není již dále inovován, přestává používat. Zadání funkce v tomto programu se provádí ve zdrojovém kódu, před spuštěním výpočtu, meze pro zobrazení grafu se nastavují předem.

Sestavení programů v jazyce Pascal je složitější než v programu Famulus (i když princip je podobný). Na rozdíl od programu Famulus se tato „větší složitost“ vyplatila, a to tak, že při vykreslování grafu si program automaticky sám nastavuje patřičné měřítko pro zobrazení grafu.

Jako nejlepší se mi jeví řešení pomocí programu Mathematica. Je velmi snadné, rychlé, k provedení separace je opět vhodné vykreslit graf funkce, což se v tomto programu provede „na jednom řádku“ – měřítka a popis os si program nastavuje automaticky. Program umožňuje rychlé a přehledné zpracování. Pro ilustraci jsem u některých metod uvedla dvojí přístup k řešení rovnice: buď s volbou přesnosti řešení nebo s volbou počtu kroků. Na rozdíl od programů Famulus i Pascal si tento program sám provede numerickou derivaci funkce pomocí jednoduchého příkazu (pro případ použití metody tečen. V programech Famulus i Pascal je z důvodů přehlednosti derivace funkce přímo zadána, programování numerické derivace by podstatně zkomplikovala výše popsané programy.

Na závěr bylo řešení rovnice $F(x) = 0$ zpracováno v prostředí DELPHI. Řešení rovnice je v tomto případě zpracováno metodou půlení intervalu. Program v DELPHI je zpracován uživatelsky, nejprve je nutno vykreslit graf funkce s cílem najít separační interval, v případě existence intervalu s průsečíkem program rovnici vyřeší. Na rozdíl od předchozích programů je možno spustit vytvořený exe soubor na libovolném počítači, nezávisle na jeho softwarovém vybavení.

Porovnání jednotlivých metod

V úvodu jsem uvedla metodu postupným přibližováním, která je velmi zdlouhavá – vysoký počet kroků. Naproti tomu metoda sečen provádí řešení s podstatně nižším počtem kroků, ale má tu nevýhodu, že nalezne řešení i mimo zadaný interval, což je nevyhovující v případech, kdy existuje vyšší počet průsečíků – pak může nalézt i průsečík mimo separační interval. Metoda tečen ještě vyžaduje provedení derivace funkce (buď numericky, nebo zadat derivaci ještě jako další funkci), navíc metoda tečen je závislá na správné volbě

vnitřního bodu intervalu – nemusí vždy konvergovat. Metoda iterační nemusí vždy konvergovat, a proto je v mnoha případech její použití nevhodné, jak je uvedeno v kapitole věnované této metodě.

Za nejspolehlivější metodu považuji metodu půlení intervalu.

Tyto výše uvedené důvody mě vedly k tomu, že při vytváření programu v prostředí DELPHI jsem použila právě metodu půlení intervalu. Tato metoda sice potřebuje k nalezení řešení vyšší počet iterací než např. metoda sečen nebo tečen (ale to je asi její jediná nevýhoda), což ale dle mého názoru při současném stavu výpočetní techniky není až tak na závadu.

Literatura

[1] Černá, R., Machalický, M., Vogel, J., Zlatník, Č.: *Základy numerické matematiky a programování*, SNTL, Praha 1987.

[2] Dvořák, L., Ledvinka, T., Sobotka, M.: *Famulus 3.1 – Výukové programy I.*, Famulus Etc., Praha 1992.

[3] Trojovský, P.: *Matematika prostřednictvím programu Mathematika*, Gaudeamus, Hradec Králové 1997.

[4] Olehla, J., Olehla, M.: *BASIC u mikropočítačů*, Nakladatelství dopravy a spojů, Praha 1988.

[5] Holan, T.: *DELPHI v příkladech*, BEN – technická literatura, Praha 2001.

[6] Kadlec, V.: *Učíme se programovat v DELPHI a jazyce Object Pascal*, Computer Press, Praha 2001.