TENTO PROJEKT JE SPOLUFINANCOVÁN
EVROPSKÝM SOCIÁLNÍM FONDEM
A STÁTNÍM ROZPOČTEM ČESKÉ REPUBLIKY

DYNAMICKÉ MODELOVÁNÍ

OLDŘICH LEPIL
LUKÁŠ RICHTEREK

Repronis 2007

Seminární materiál projektu Učíme fyziku moderně
Další vzdělávání učitelů fyziky Olomouckého kraje
Slovanské gymnázium Olomouc

ISBN 978-80-7329-156-3

Obsah

Úvod

Vytváření matematických modelů fyzikálních dějů patří k typickým příkladům využití počítačů ve výuce. Současně se tím otevírá široká oblast aplikací, založená na vzájemných analogiích dějů z různých oblastí fyziky. Typické jsou například analogie dějů z dynamiky pohybu hmotného bodu a dějů v elektrických obvodech, které jsou popsány diferenciálními rovnicemi. Jako dynamické modelování pak označujeme použití počítače k numerickému řešení těchto diferenciálních rovnic. Didaktickou předností metody dynamického modelování je možnost řešit jednoduchými matematickými postupy i úlohy, jejichž analytické řešení není žákovi známo, nebo je náročné z matematického hlediska.

Pro didaktické účely metodu dynamického modelování použil pravděpodobně jako první jeden z nejvýznamnějších fyziků 20. století, nositel Nobelovy ceny za rok 1965, Richard P. Feynman, který ji popisuje ve své proslulé učebnici fyziky [1]. Feynman ještě v době, kdy nebyla běžně k dispozici potřebná výpočetní technika, řešil metodou dynamického modelování úlohy, jejichž cílem bylo usnadnit studentům pochopení podstaty pohybových rovnic. Přímo na přednáškách prováděl se studenty potřebné výpočty s použitím kapesního kalkulátoru (viz [1], s. 128). Řešení čtyř Feynmanových úloh, které jsou v jeho učebnici uvedeny bez řešení, je zařazeno v tomto studijním textu.

V naší didaktické literatuře se poprvé setkáváme s náměty na využití metody dynamického modelování v publikaci [2], kde jsou uvedeny programy zpracované v programovacím jazyce BASIC. Malá přehlednost těchto programů společně s nízkou úrovní tehdy dostupné výpočetní techniky (IQ 151) byla příčinou malého zájmu o tuto metodu mezi učiteli. Významný pokrok v používání metody dynamického modelování znamenalo rozšíření programu FAMULUS [3], který umožňuje, aby se uživatel soustředil hlavně na fyzikální jádro řešeného problému a mohl snadno vytvářet a zejména modifikovat jednotlivé modely i bez hlubších znalostí programování. FAMULUS však umožňoval modelování fyzikálních dějů jen na počítačích s operačním systémem MS DOS a jeho verze pro operační systém Windows již nevznikla.

Jiným programem, kterým lze na základě numerického řešení diferenciálních rovnic modelovat nejrůznější fyzikální děje, je program *Interactive Physics* [4]. Tento program umožňuje velmi názorné simulace modelovaných dějů, ale pracuje s předem definovanými objekty a parametry fyzikálních situací, takže uživatel jen pomocí myši model sestaví na displeji počítače, vybere potřebné konstanty, zadá počáteční hodnoty veličin a po spuštění pozoruje probí-

hající děj. Tím do jisté míry žákovi uniká podstata modelovaného děje a zejména jeho popis odpovídající pohybovou rovnicí, popř. jinou matematicky vyjádřenou zákonitostí.

Proto je dále pro vytváření modelů využit program Modelování (Modeling), který je jednou ze čtyř základních součástí softwaru systému Coach 5, určeného především pro podporu reálných experimentů počítačem. I bez použití hardwaru systému Coach 5 je možné k modelování použít demoverzi programu, která je volně dostupná na webu [5]. Demoverze je plně funkční s tím, že není možný zápis vytvořeného modelu. Ten je však možné zkopírovat do textového editoru a samostatně uložit. Program Modelování má v podstatě stejnou koncepci jako program FAMULUS, tzn. v jednom okně programu se vypíší proměnné, konstanty a počáteční hodnoty fyzikálních veličin a ve druhém okně se vypíše program pro numerické řešení diferenciálních rovnic modelovaného děje. Postup vytváření modelu je uveden dále.

Metoda dynamického modelování

Podstatu metody dynamického modelování ukážeme nejprve na příkladech mechanických pohybů. Abychom vytvořili dynamický model pohybu tělesa (hmotného bodu), musíme znát:

1. pohybovou rovnici pro daný děj,
2. počáteční podmínky modelovaného děje.

Pohybová rovnice je aplikací druhého pohybového zákona

$$m\boldsymbol{a} = m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{F}$$

na konkrétní případ pohybu hmotného bodu v inerciální vztažné soustavě. Síla, která působí na hmotný bod, může být konstantní, popř. se může měnit v závislosti na poloze hmotného bodu a jeho rychlosti, nebo je to časově proměnná síla.

Nejčastěji modelujeme děje, při nichž na hmotný bod o hmotnosti m působí následující typy sil:

a) **konstantní tíhová síla** v homogenním tíhovém poli

$$\boldsymbol{F} = m\boldsymbol{g}$$

kde $\boldsymbol{g} = \mathbf{konst}$. je tíhové zrychlení;

b) **gravitační síla** v radiálním gravitačním poli centrálních sil, tzn. v gravitačním poli tělesa s velkou hmotností $M >> m$

$$\boldsymbol{F} = -\kappa\frac{Mm}{r^3}\boldsymbol{r},$$

kde $\boldsymbol{r}$ je polohový vektor určující vzdálenost hmotného bodu od středu centrálního tělesa (počátek vztažné soustavy);

c) **síla pružnosti** působící na hmotný bod při jeho vychýlení z rovnovážné polohy (v ní je počátek vztažné soustavy)

$$\boldsymbol{F} = -k\boldsymbol{r}\,,$$

kde k je tuhost pružiny vytvářející sílu pružnosti.

Uvedené tři druhy sil představují síly konzervativní, při jejichž působení se zachovává mechanická energie. Na hmotný bod působí obvykle současně se silami konzervativními také síly disipativní, tzn. síly, při jejichž působení dochází k nevratné přeměně mechanické energie na jinou formu energie (obvykle na vnitřní energii pohybujícího se tělesa a jeho okolí). Nejčastější jsou disipativní síly závisející na rychlosti $\boldsymbol{v}$ pohybu hmotného bodu (tělesa). Tohoto druhu jsou síly:

d) **síla odporu** viskózního prostředí při pomalých pohybech

$$\boldsymbol{F} = -b\boldsymbol{v},$$

kde b je součinitel úměrnosti mezi silou odporu a rychlostí tělesa (Stokesův zákon: $F = 6\pi\eta rv$, kde η je dynamická viskozita a r je poloměr tělesa ve tvaru koule);

e) **síla odporu** prostředí při rychlejších pohybech (při proudění odporujícího prostředí kolem tělesa nastává vírové obtékání)

$$\boldsymbol{F} = -Kv\boldsymbol{v}$$

(Newtonův vzorec: $\boldsymbol{F} = CS\rho v\boldsymbol{v}/2$, C je součinitel odporu, S je plocha příčného řezu tělesa, ρ je hustota prostředí).

Síla působící na hmotný bod může být také funkcí času. Nejčastěji je to harmonicky proměnná síla o velikosti

$$F = F_{\mathrm{m}}\sin\omega t\,,$$

kde F_{m} je amplituda síly a ω je úhlová frekvence.

Při dynamickém modelování obvykle řešíme případy pohybu po přímce nebo v rovině. Jestliže vhodně zvolíme vztažný bod a směr os vztažné soustavy, můžeme použít pohybové rovnice vyjádřené souřadnicemi výsledné síly, které můžeme obecně napsat ve tvaru:

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = F_x = f_x(t,x,y,z,v_x,v_y,v_z)$$

$$m\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = F_y = f_y(t,x,y,z,v_x,v_y,v_z)$$

$$m\frac{\mathrm{d}^2z}{\mathrm{d}t^2} = F_z = f_z(t,x,y,z,v_x,v_y,v_z)$$

Další zjednodušení modelu nastane, jestliže působící síla závisí jen na některé z veličin, popř. jejich souřadnic, nebo když je konstantní.

Na základě matematického modelu je pomocí pohybové rovnice určována poloha hmotného bodu (pohybujícího se tělesa) a jeho okamžitá rychlost v závislosti na čase. To znamená, že k určité posloupnosti časů $\{t_i\}$ je nalezena odpovídající posloupnost hodnot příslušných polohových vektorů $\{\boldsymbol{r}(t_i)\}$ a okamžitých rychlostí $\{\boldsymbol{v}(t_i)\}$, popř. souřadnic těchto veličin. Vypočtené hodnoty jsou pak zobrazeny tabelárně nebo častěji graficky. Zobrazuje se v určitém měřítku buď přímo trajektorie pohybu hmotného bodu ve zvolené vztažné soustavě, nebo je studovaný děj popsán grafy závislosti jednotlivých veličin na čase (časovými diagramy).

Při vytváření dynamického modelu má značný význam hodnota tzv. *časového kroku h*, tzn. rozdílu dvou po sobě následujících hodnot aritmetické posloupnosti $\{t_i\}$:

$$h = t_{i+1} - t_i$$

Z matematického hlediska jsou pohyby hmotného bodu, ale i průběh napětí a proudu v elektrickém obvodu nebo i děje z jiných oblastí fyziky často popsány diferenciálními rovnicemi. Dynamické modelování pomocí počítače je případem použití algebraické metody přibližného řešení těchto rovnic. Existuje řada metod, kterými lze přibližné řešení diferenciálních rovnic realizovat. Jednotlivé metody se liší zejména přesností a v řadě případů jsou tyto metody poměrně komplikované. Pro výukové účely však vystačíme s elementárními metodami, z nichž nejjednodušší je tzv. Eulerova metoda, na kterou se dále zaměříme.

Tuto metodu numerického řešení obyčejných diferenciálních rovnic publikoval již v roce 1768 švýcarský matematik Leonhard Euler jako řešení problému, jak v daném čase získat co nejlepší aproximaci derivace. Podstata této

metody je dobře patrná z obecné podoby dynamického modelu, který budeme dále aplikovat na jednotlivé případy. Tvoří ho pohybová rovnice

$$a = \frac{F(t,r,v)}{m} = a(t,r,v)\,, \tag{1}$$

která je východiskem výpočtu posloupností $\{ r_i \}$, $\{ v_i \}$ na základě vztahů

$$r_{i+1} = r_i + vh \tag{2}$$
$$v_{i+1} = v_i + ah \tag{3}$$
$$t_{i+1} = t_i + h \tag{4}$$

Hodnotu a volíme v intervalu (a_i, a_{i+1}), kde

$$a_i = a(t_i, v_i, r_i)$$
$$a_{i+1} = a(t_{i+1}, v_{i+1}, r_{i+1})$$

a hodnotu v volíme v intervalu (v_i, v_{i+1}). Při numerickém výpočtu je třeba vztahy (1) až (4) použít v určitém pořadí, přičemž (1) musí předcházet (3).

V dynamickém modelu jsou diferenciální rovnice nahrazeny diferenčními rovnicemi, přičemž pro velikost diferencí $\Delta r = r_{i+1} - r_i$ a $\Delta v = v_{i+1} - v_i$ je rozhodující velikost časového kroku. Ta také určuje přesnost numerického řešení diferenciálních rovnic. Použití jednoduché Eulerovy metody tedy předpokládá, že zvolíme co nejmenší časový krok. Ve většině modelů, které budeme řešit na střední škole, s tímto postupem vystačíme. Mohou však nastat případy, kdy volba malého časového kroku nedává dostatečně přesné výsledky a musíme použít postupy, které dávají výsledky lepší.

Eulerova metoda v podstatě odpovídá prvním dvěma členům tzv. Taylorova rozvoje, který vyjadřuje hodnotu funkce $f(x)$ v okolí bodu a jako řadu

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^3(a)}{3!} + \ldots = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k\,.$$

Pro zpřesnění přibližného výpočet nám postačí, když k prvním dvěma členům rozvoje přidáme ještě člen třetí, takže vztah pro výpočet polohy hmotného bodu bude mít tvar

$$r_{i+1} = r_i + v_i h + \frac{1}{2} a h^2 .$$

Další zpřesnění numerického řešení diferenciálních rovnic představuji metody Runge-Kuttovy, které lze obecně zapsat ve tvaru (viz [6]):

$$y_{n+1} = y_n + h \sum_{i=1}^{p} w_i k_i$$

$$k_i = f\left(t + \alpha_i h,\ y_n + h \sum_{j=1}^{i-1} \beta_{ij} k_j \right)$$

Koeficienty *k* u těchto metod jsou vypočteny tak, aby metoda řádu *p* odpovídala Taylorovu rozvoji funkce funkce *y*(*t*) stejného řádu. To znamená, že Eulerova metoda je vlastně metodou Runge-Kuttovou prvního řádu.

V praxi se nejvíce používá metoda Runge-Kuttova čtvrtého řádu, jejíž koeficienty mají pro model pohybu hmotného bodu tvar (viz [7]):

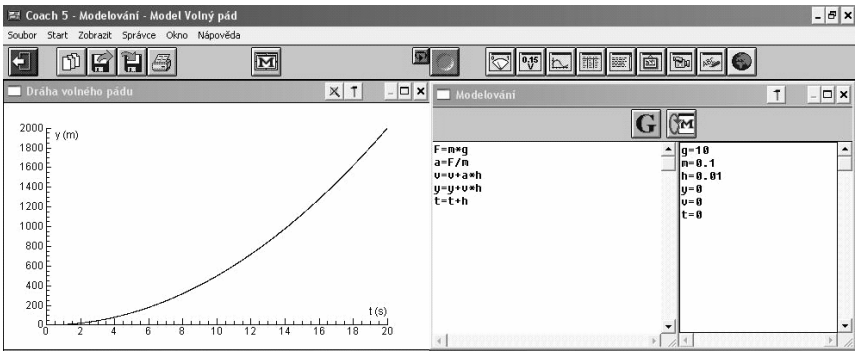
$$\begin{aligned}
k_1 &= a_i \\
k_2 &= a\left(t_i + h/2,\ v_i + k_1 h/2,\ r_i + v_i h/2 + k_1 h^2/8\right) \\
k_3 &= a\left(t_i + h/2,\ v_i + k_2 h/2,\ r_i + v_i h/2 + k_2 h^2/8\right) \\
k_4 &= a\left(t_i + h,\ v_i + k_3 h,\ r_i + v_i h + k_3 h^2/2\right)
\end{aligned}$$

Rovnice dynamického modelu pak mají tvar:

$$\begin{aligned}
r_{i+1} &= r_i + v_i h + \left(k_1 + k_2 + k_3\right) h^2/6 \\
v_{i+1} &= v_i + \left(k_1 + 2k_2 + 2k_3 + k_4\right) h/6 \\
t_{i+1} &= t_i + h
\end{aligned}$$

Dynamické modely lze realizovat na počítači některým z běžných programovacích jazyků (viz druhou část tohoto studijního materiálu). V této části studijního materiálu jsou dynamické modely fyzikálních dějů vytvářeny v prostředí programu Modelování softwaru Coach 5 (obr. 1). V levé části okna

pro zápis modelu jsou uvedeny proměnné, konstanty a počáteční hodnoty fyzikálních veličin a v pravé části je zapsán příslušný model. Při zápisu je třeba používat syntax jazyka Coach, která je obdobná jako u programu FAMULUS, rozdíl je např. v tom, že kompilátor programu nerozlišuje malá a velká písmena, desetinná čísla je třeba zapisovat s nulou na místě jednotek (např. 0.1 místo .1) a před poznámky nebo části programu, které má kompilátor ignorovat, je třeba psát znak <‘> místo znaku <!>. Není povoleno používat číslici jako první znak jména proměnné a jména identická s klíčovými slovy. Výrazy pro jednotlivé operace modelu se oddělují jen mezerou (pokud jsou na jednom řádku), popř. oddělovačem <Enter>, kdy se každý výraz zapíše na samostatnou řádku. Stručný přehled jazyka Coach je v příloze.



Obr. 1

Jistým omezením programu Modelování je maximální počet cyklů výpočtu, který je 16 300. Pro většinu modelů je však tento počet dostačující a umožňuje volbu dostatečně malého časového kroku.

V další části studijního materiálu jsou popsány jednotlivé modely vhodné pro didaktické využití ve středoškolské fyzice v následující posloupnosti

1. Pohyby v silových polích.
2. Kmitání mechanických oscilátorů.
3. Přechodné děje v elektrických obvodech.
4. Kmitání elektromagnetických oscilátorů.
5. Modely z fyziky mikrosvěta.
6. Feynmanovy úlohy.

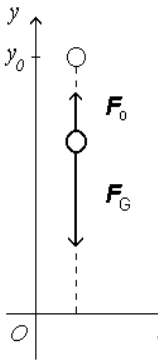
1 Pohyby v silových polích

1.1 Volný pád v odporujícím prostředí

Uvažujeme pád koule o poloměru r a hmotnosti m. Vztažnou soustavu pro popis pohybu umístíme tak, že vztažný bod soustavy je totožný s místem dopadu koule a pád koule probíhá v záporném směru osy y (obr. 2). Počáteční poloha koule v čase $t = 0$ je určena souřadnicí y_0.

Při pádu v odporujícím prostředí na padající těleso působí jednak tíhová síla $\boldsymbol{F}_G = m\boldsymbol{g}$, jednak odporová síla $\boldsymbol{F}_o$, jejíž velikost je určena Newtonovým vzorcem

$$F_o = \frac{1}{2} C S \rho v^2 .$$



Obr. 2

Součinitel C odporu prostředí závisí na tvaru tělesa a pro kouli $C = 0{,}5$, plocha S příčného řezu koule $S = 4\pi r^2$.

Vzhledem k přímočarému pohybu koule napíšeme pohybovou rovnici pomocí souřadnic. Poněvadž tíhová síla má opačný směr, než je kladný směr osy y, je její souřadnice záporná a souřadnice síly odporu prostředí je naopak kladná. Souřadnice výsledné síly působící na kouli je tedy vyjádřena rovnicí

$$F = -F_G + F_o$$

čili

$$F = -mg + \frac{1}{2} C S \rho v^2 .$$

Souřadnice zrychlení je tedy vyjádřena vztahem

$$a = -g + kv^2 ,$$

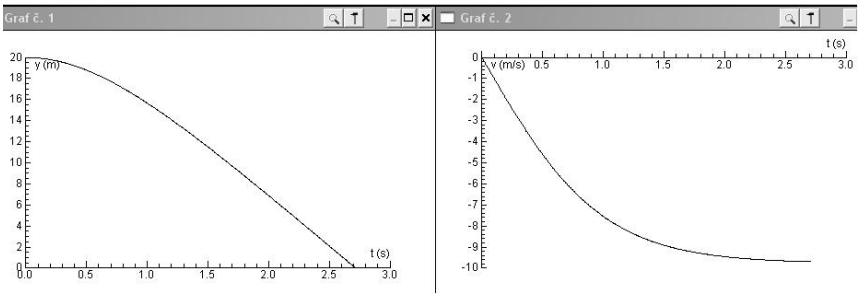
kde $k = \pi r^2 C \rho / (2m)$ je pro daný pohyb konstanta.

Na základě vztahu pro zrychlení můžeme napsat dynamický model pádu v odporujícím prostředí:

Model	Proměnné, konstanty, počáteční hodnoty
a=-g+k*v^2 v=v+a*h y=y+v*h t=t+h if y<0 then stop endif	g=10 r=0.2 m=0.3 ro=1 C=0.5 k=pi*r^2*C*ro/(2*m) h=0.005 y=20 v=0 t=0

Grafy na obr. 3 zachycují souřadnici tělesa a jeho rychlost jako funkci času:

$$y = f(t); \quad v = f(t)$$



Obr. 3

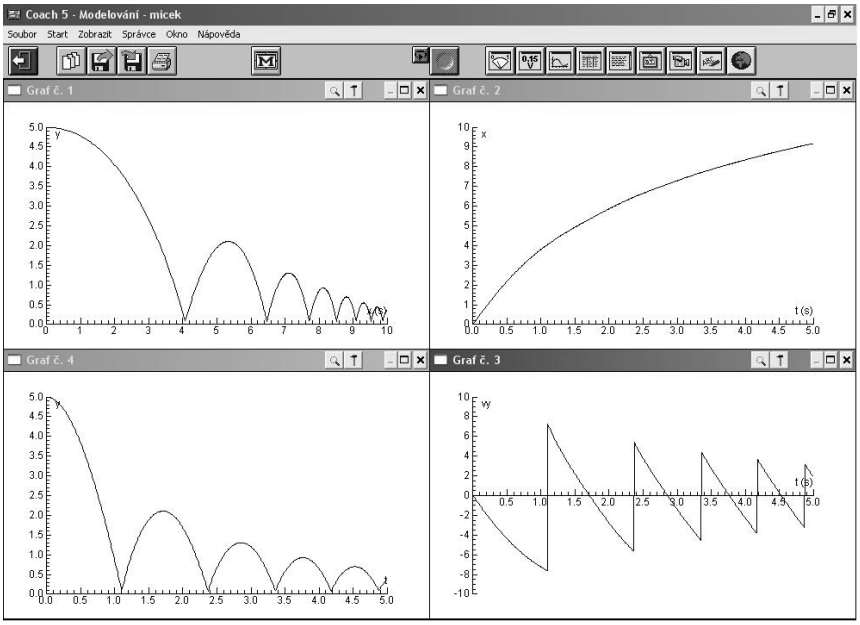
1.2 Pád míčku s odrazem

Modeluje se vodorovný vrh pružného míčku, který dopadá na pevnou podložku. Jeho kinetická energie se při dopadu mění na potenciální energii pružnosti deformovaného míčku a míček odskočí od podložky zmenšenou rychlostí, která závisí na součiniteli restituce (v modelu RestK). Rychlost má po odrazu opačný směr a míček vystoupí do výšky odpovídající vrhu svislému vzhůru při dané počáteční rychlosti. Po dosažení maximální výšky se celý děj opakuje, přičemž po každém odrazu se část mechanické energie míčku přemění nevratně na jinou formu energie (teplo). Do celkové energetické bilance pohybu jsou zahrnuty také ztráty způsobené odporem prostředí, v němž se míček pohybuje. Souřadnice síly odporu prostředí jsou stejné jako při šikmém vrhu (viz model

1.4). Model je poučný pro pochopení rozdílu mezi souřadnicí a velikostí vektorové veličiny. Průběh jednotlivých veličin je patrný z grafů na obr. 4.

Model	Proměnné, konstanty, počáteční hodnoty

`v=sqrt(vx^2+vy^2)`
`Fx=-k*v*vx`
`Fy=-k*v*vy-m*g`
`ax=Fx/m`
`ay=Fy/m`
`vx=vx+ax*h`
`vy=vy+ay*h`
`x=x+vx*h`
`y=y+vy*h`
`if y<R then vy=-RestK*vy endif`
`t=t+h`
	`g=10`
	`R=0.1`
	`m=0.3`
	`S=4*pi*R^2`
	`ro=1`
	`C=0.5`
	`k=C*ro/2*S`
	`RestK=0.95`
	`t=0 h=0.01`
	`x=0 y=5`
	`vx=5 vy=0`



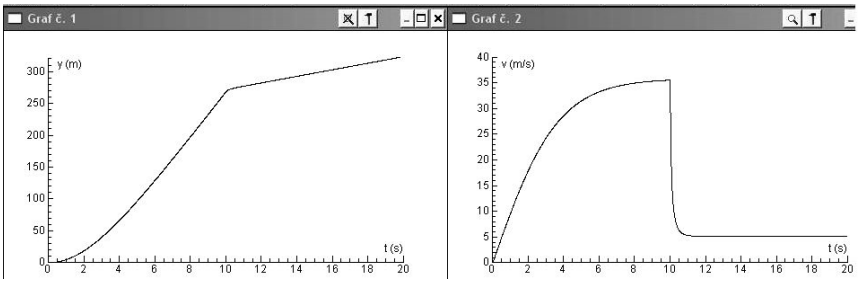
Obr. 4

1.3 Pohyb výsadkáře s opožděným otevřením padáku

Modeluje se pohyb výsadkáře, který po výskoku z letadla padá 10 s volným pádem, pak se mu otevře padák a rychlost pádu se v krátké době ustálí na konstantní hodnotě, kterou se pohybuje rovnoměrně. Obě fáze pohybu se liší hodnotami součinitele C odporu prostředí a obsahem plochy S příčného řezu těla výsadkáře v první fázi pohybu a otevřeného padáku ve druhé fázi pohybu. Model je v tabulce a grafy na obr. 5 zobrazují souřadnici y pohybu jako funkci času a exponenciální průběh změn rychlosti výsadkáře.

Model	Proměnné, konstanty, počáteční hodnoty

`if t>10 then C=C2 S=S2 endif`
`a=g-C*S*ro*v^2/(2*m)`
`y=y+v*h`
`v=v+a*h`
`t=t+h`
	`g=9.8`
	`m=95`
	`ro=1.2`
	`C1=1.1`
	`S1=1.1`
	`C2=1.33`
	`S2=45`
	`S=S1`
	`C=C1`
	`h=0.01`
	`y=0`
	`v=0`
	`t=0`



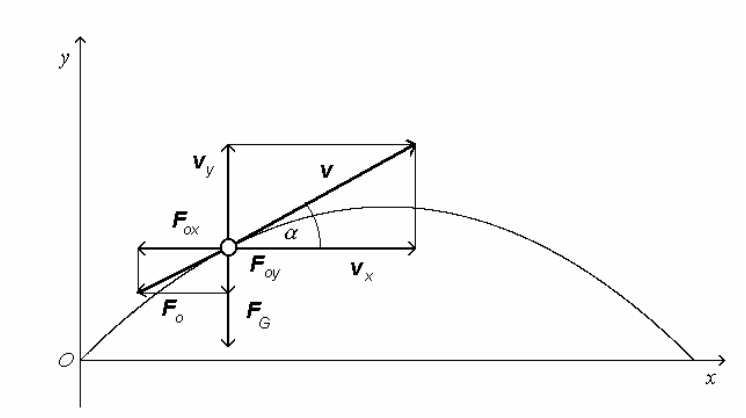
Obr. 5

1.4 Šikmý vrh

Při šikmém vrhu vzhůru má těleso v počátečním okamžiku rychlost o velikosti v_0 a vektor počáteční rychlosti $\boldsymbol{v}_0$ svírá s vodorovným směrem úhel α (obr. 6). Z obr. 6 je zřejmé, že v určitém bodě trajektorie je velikost okamžité rychlosti $v = \sqrt{v_x^2 + v_y^2}$. V odporujícím prostředí na těleso působí vedle tíhové síly $\boldsymbol{F}_G$ také síla odporu prostředí $\boldsymbol{F}_o$, pro jejíž složky platí

$$F_{ox} = mkv^2 \cos\alpha = mkv^2 (\frac{v_x}{v}) = mkvv_x,$$

$$F_{oy} = mkv^2 \sin\alpha = mkv^2 (\frac{v_y}{v}) = mkvv_y.$$



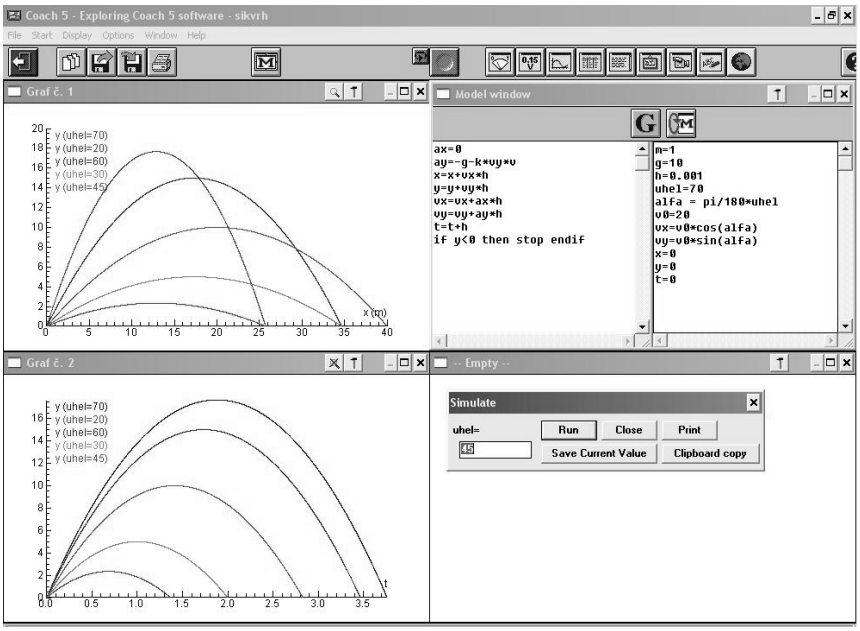
Obr. 6

Rovnice pro souřadnice složek zrychlení budou mít s ohledem na směr jednotlivých vektorů tvar:

```
ax = - k*v*vx/m
ay = - g - k*v*vy/m
```

Dynamický model šikmého vrhu bez odporu prostředí je v tabulce. Zkoumáme závislost dálky vrhu pro různé úhly vrhu. Odpovídající grafy trajektorií a doby pohybu vrženého tělesa jsou na obr. 7.

Model	Proměnné, konstanty, počáteční hodnoty
ax=0 ay=-g x=x+vx*h y=y+vy*h vx=vx+ax*h vy=vy+ay*h t=t+h if y<0 then stop endif	m=1 g=10 h=0.001 uhel=45 alfa = pi/180*uhel v0=20 vx=v0*cos(alfa) vy=v0*sin(alfa) x=0 y=0 t=0

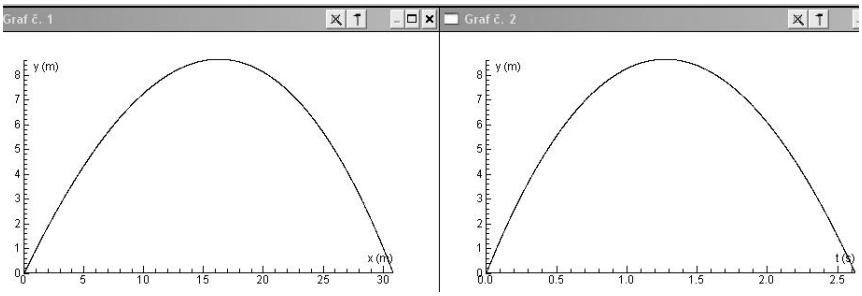


Obr. 7

Model pohybu tělesa při šikmém vrhu v odporujícím prostředí po balistické křivce je v tabulce. Na obr. 8 jsou grafy balistické křivky a doby pohybu po této křivce.

Model	Proměnné, konstanty, počáteční hodnoty

`v=sqrt(vx^2+vy^2)`
`ax=-k*vx*v`
`ay=-g-k*vy*v`
`x=x+vx*h`
`y=y+vy*h`
`vx=vx+ax*h`
`vy=vy+ay*h`
`t=t+h`
`if y<0 then stop endif`
	`m=1`
	`g=10`
	`h=0.001`
	`uhel=45`
	`k=0.03`
	`alfa=pi/180*uhel`
	`v0=20`
	`vx=v0*cos(alfa)`
	`vy=v0*sin(alfa)`
	`x=0`
	`y=0`
	`t=0`



Obr. 8

Stejný dynamický model použijeme i pro znázornění trajektorie vodorovného vrhu (úhel vrhu $\alpha = 0$ a $y_0 > 0$), popř. pro jiné případy vrhů.

1.5 Start rakety

Pohyb rakety startující svisle vzhůru je příkladem pohybu tělesa, jehož hmotnost se při pohybu mění. Úbytek hmotnosti rakety vlivem úniku hořících plynů označíme jako hmotnostní tok $Q = \Delta m/\Delta t$. Jestliže je startovní hmotnost rakety m_0, pak okamžitá hmotnost rakety $m = m_0 - Qt$. Předpokládáme, že pokud pracuje raketový motor, koná raketa rovnoměrně zrychlený pohyb a mění se její hybnost. Při vyjádření změny hybnosti musíme uvážit jak změnu rychlosti Δv, tak změnu hmotnost Δm. Jestliže v čase t má raketa hybnost p, pak po uplynutí času Δt bude

$$p + \Delta p = (m + \Delta m)(v + \Delta v) + u\Delta m,$$

kde u je rychlost plynů vystupujících z motoru rakety, měřená vzhledem ke vztažné soustavě, vůči které se raketa pohybuje. Pro zjednodušení modelu zanedbáme součin $\Delta m \Delta v$ a pro změnu hybnosti dostaneme

$$\Delta p = m\Delta v + v\Delta m + u\Delta m.$$

Celková síla F, která na raketu působí, má dvě složky, gravitační sílu F_1 (na počátku pohybu ji ztotožníme s tíhovou silou, čili $F_1 = mg$) a aerodynamickou odporovou sílu, kterou pro zjednodušení modelu zanedbáme. Pro raketu tedy platí:

$$m\Delta v + (v + u)\Delta m = -F_1\Delta t$$

Relativní rychlost spálených plynů vůči raketě je $c = v + u$, takže pro změnu rychlosti platí

$$\Delta v = -\frac{c\Delta m}{m} - g\Delta t$$

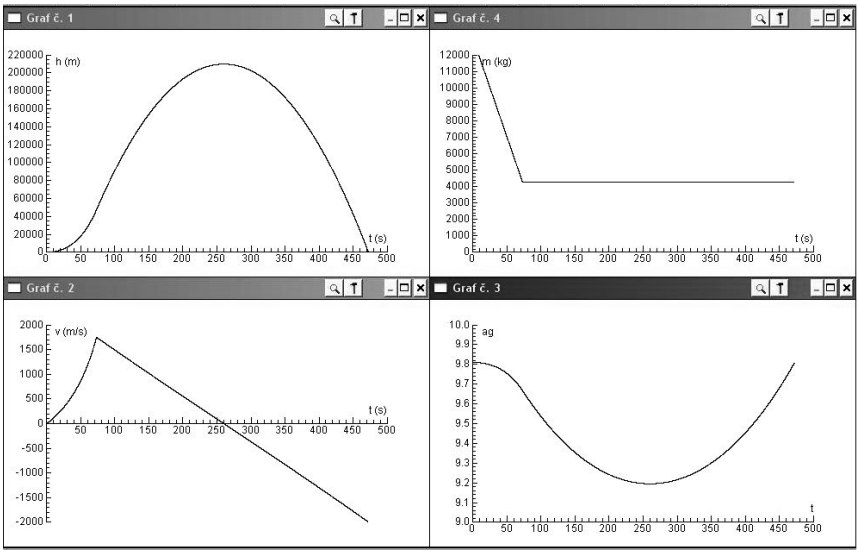
a zrychlení rakety $a = \Delta v/\Delta t$.

Model sestavíme pro konkrétní příklad německé rakety V2, kterou Němci na konci 2. světové války ohrožovali Londýn. Její startovní hmotnost byla 13 000 kg, z toho hmotnost paliva byla 8 750 kg a palivo se spalovalo rychlostí 120 kg $\cdot$ s^{-1}. Rychlost spálených plynů byla 2 200 m $\cdot$ s^{-1} a motor rakety pracoval přibližně 73 s. Za těchto podmínek raketa dosáhla konečnou rychlost

1,8 km · s^{-1} a motor rakety přestal pracovat ve výšce 48 km. V modelu se počítá také se změnou gravitačního zrychlení ($a_g = g\, y_0^2 / y^2$). Po zhasnutí motorů raketa pokračuje v pohybu jako při vrhu svislém vzhůru. Charakter pohybu je patrný z grafů na obr. 9.

Model	Proměnné, konstanty, počáteční hodnoty

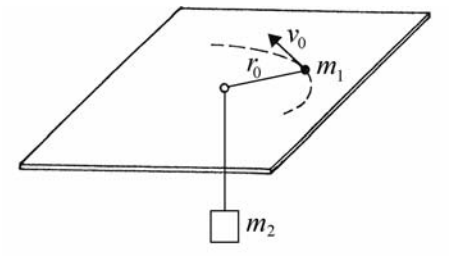
`t=t+h`
`dm=-q*h`
`m=m+dm`
`if m<=mk then q=0 endif`
`ag=g*y0^2/y^2`
`dv=-(c/m)*dm-ag*h`
`v=v+dv`
`y=y+v*h`
`h=y-y0`
`if h<=0 then stop endif`
	`t=0`
	`h=0.05`
	`q=0.12e3`
	`c=2.2e3`
	`m=13e3`
	`g=9.81`
	`v=0`
	`y0=6.38e6`
	`y=y0`
	`mk=4.25e3`



Obr. 9

1.6 Pohyb v centrálním poli (provrtaný stůl)

Úloha popisuje pohyb tělesa o hmotnosti m_1 po rovinné desce, které je spojeno vláknem procházejícím otvorem v desce se závažím o hmotnosti m_2. Těleso se v počátečním okamžiku pohybuje rychlostí v_0, která je kolmá k vláknu (obr. 10). Při zjednodušeném pohledu by se zdálo, že závaží přitáhne těleso po spirálové trajektorii k otvoru (silové působení by odpovídalo známé úloze o pohybu těles spojených vláknem). Situace však simuluje pohyb v centrálním poli a to znamená, že se zachovává velikost plošné rychlosti.



Obr. 10

Podrobným teoretickým řešením, které vede ke složitému integrálnímu výrazu pro trajektorii tělesa, by se ukázalo, že trajektorie tělesa leží v oblasti mezi dvěma kružnicemi, jejichž poloměry závisejí na hmotnostech tělesa a závaží, na počáteční poloze tělesa a jeho rychlosti. Simulaci pohybu umožňuje pohybová rovnice pro velikost centrální síly

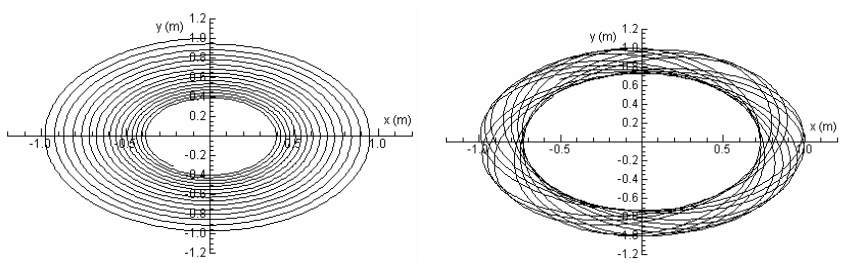
$$F = \frac{m_1 m_2}{m_1 + m_2}\left(\frac{r_0^2 v_0^2}{\sqrt{\left(x^2 + y^2\right)^3}} + g \right),$$

kde x a y jsou souřadnice tělesa (počátek soustavy souřadnic je v místě otvoru), g je tíhové zrychlení. Z rovnice určíme složky a_x a a_y zrychlení a sestavíme dynamický model. Volbou počáteční rychlosti pak můžeme ovlivnit tvar trajektorie. Složky zrychlení odpovídající třecí síle jsou $a_{xt} = mgf \cos \alpha/m = gfv_x/v$ a $a_{yt} = mgf \sin \alpha/m = gfv_y/v$, kde f je součinitel smykového tření. Při počáteční rychlosti $v_0 = 2{,}55\ \mathrm{m \cdot s^{-1}}$ se těleso pohybuje bez tření po kružnicové trajektorii a s třením po spirále (obr. 11 vlevo). Při jiných hodnotách rychlosti v_0 (např.

$v_0 = 2{,}0\ \mathrm{m} \cdot \mathrm{s}^{-1}$) se těleso pohybuje po složité trajektorii a v případě bez tření jeho vzdálenost od otvoru kolísá mezi největší a nejmenší hodnotou (obr. 11 vpravo).

Model	Proměnné, konstanty, počáteční hodnoty

`r=sqrt(x^2+y^2)`
`v=sqrt(vx^2+vy^2)`
`axt=g*f*vx/v`
`ayt=g*f*vy/v`
`ax=-m1/(1+m1/m2)*((R0^2*v0^2)/`
`  r^(3/2)+g)*x/r-axt`
`ay=-m1/(1+m1/m2)*((R0^2*v0^2)/`
`  r^(3/2)+g)*y/r-ayt`
`vx=vx+ax*h`
`vy=vy+ay*h`
`x=x+vx*h`
`y=y+vy*h`
`wait(0.001)`
`t=t+h`
`if t>10 then stop endif`
	`r0=1`
	`m1=2`
	`m2=1`
	`g=9.8`
	`vx=-2.5`
	`vy=0`
	`x=0`
	`y=r0`
	`t=0`
	`h=0.01`
	`f=0.01`



Obr. 11

1.7 Pohyb umělé družice v centrálním poli Země

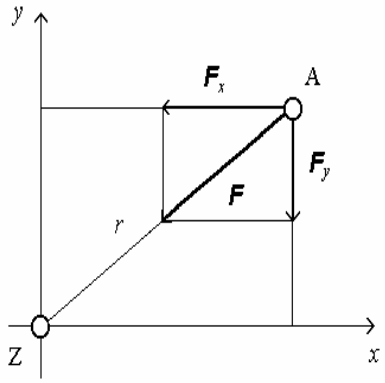
Budeme uvažovat pohyb umělé družice Země ve vztažné soustavě, jejíž vztažný bod je totožný se středem Země. V určitém okamžiku je družice v bodě A vztažné soustavy, jehož vzdálenost od středu Země je r (obr. 12). Na družici působí gravitační síla $\boldsymbol{F}_{\mathrm{g}}$, která družici uděluje zrychlení. Podle obr. 12 rozložíme tuto sílu na složku $\boldsymbol{F}_{gx}$ a $\boldsymbol{F}_{gy}$. Pro kladné x bude souřadnice složky F_{gx} záporná a platí $F_{gx}/F = -x/r$. Podobně pro souřadnici složky F_{gy} platí $F_{gy} = -y/r$.

Tyto rovnice napíšeme ve tvaru

$$ma_x = -\kappa M_Z m x / r^3,$$
$$ma_y = -\kappa M_Z m y / r^3,$$

kde κ je gravitační konstanta ($\kappa = 6{,}67 \cdot 10^{-11}$ N · m^2 · kg^{-2}; v počítačovém modelu je použita značka `k`), M_Z je hmotnost Země ($M_Z = 5{,}9 \cdot 10^{24}$ kg) a $r = x^2 + y^2$. V dynamickém modelu použijeme rovnice pro zrychlení:

```
ax = -k*Mz*m*x/r^3
ay = -k*Mz*m*y/r^3
```



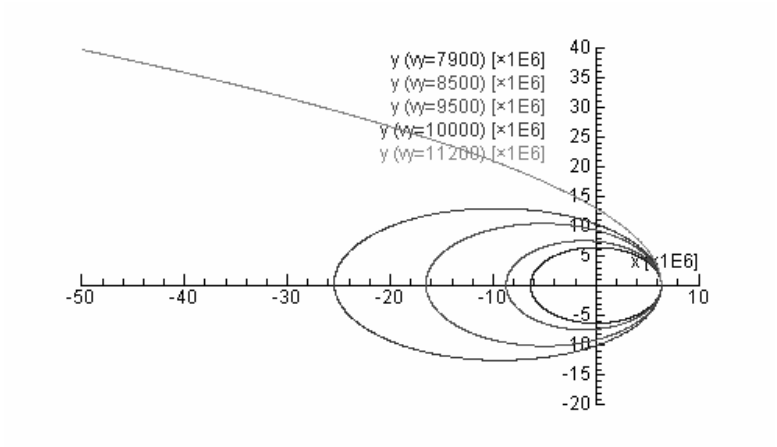
Obr. 12

Nejvhodnějším příkladem je řešení pohybu družice ve vzdálenosti $r = R_Z$ (R_Z je poloměr Země; $R_Z = 6{,}371 \cdot 10^6$ m). Jako počáteční hodnotu rychlosti uvažujeme první kosmickou, čili kruhovou rychlost $v_k = 7{,}9$ km $\cdot$ s^{-1}, kterou zvolíme např. jako y-ovou souřadnici počáteční rychlosti ($v_x = 0$). Postupným zvětšováním hodnoty počáteční rychlosti ukážeme, že trajektorie umělé družice Země se mění v elipsu a při druhé kosmické čili parabolické rychlosti $v_p = 11{,}2$ m $\cdot$ s^{-1} se trajektorie mění na parabolu.

Konkrétní podoba dynamického modelu pohybu umělé družice Země je v tabulce. Na obr. 13 jsou trajektorie pro některé kosmické rychlosti.

Model	Proměnné, konstanty, počáteční hodnoty

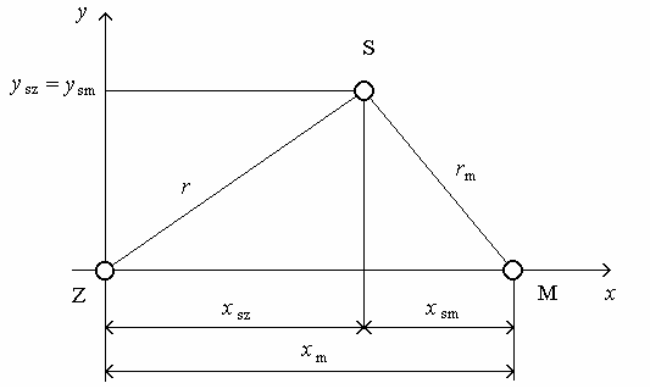
`t=t+h`
`x=x+vx*h`
`y=y+vy*h`
`r=sqrt(x*x+y*y)`
`ax=-k*x*Mz/r^3`
`ay=-k*y*Mz/r^3`
`vx=vx+ax*h`
`vy=vy+ay*h`
	`k=6.67e-11`
	`Mz=5.98e24`
	`rz=6.378e6`
	`ax=k*Mz/rz^2`
	`ay=0`
	`h=20`
	`x=6.378e6`
	`y=0`
	`vy=7.9*1e3`
	`v0x=0*1e3`



Obr. 13

1.8 Pohyb kosmické sondy

Pohyb kosmické sondy je příkladem pohybu tělesa, na které gravitačně působí dvě jiná kosmická tělesa. Budeme uvažovat pohyb kosmické sondy ze Země k Měsíci. To znamená, že na sondu působí gravitačními silami Země o hmotnosti $M_Z = 5{,}983 \cdot 10^{24}$ kg a Měsíc o hmotnosti $M_M = 7{,}374 \cdot 10^{22}$ kg. Vzájemná vzdálenost těles je $60{,}13R_Z$ (poloměr Země $R_Z = 6{,}371 \cdot 10^6$ m). Celá soustava je na obr. 14. Model pohybu kosmické sondy bude značně zjednodušen. Zjednodušení spočívá v tom, že neuvažujeme vlastní pohyb Měsíce kolem Země a pohyb sondy není při jejím pohybu korigován např. silovým působením raketových motorů.



Obr. 14

Pro určení gravitačních sil, které působí na kosmickou sondu, musíme určit nejen souřadnice polohy vzhledem k Zemi (x_{SZ}, y_{SZ}), ale i relativní souřadnice sondy vzhledem k Měsíci: $x_{SM} = x_{SZ} - x_M$, $y_{SM} = y_{SZ} - y_M$, kde x_M, y_M jsou souřadnice Měsíce ve vztažné soustavě spojené se Zemí. Podle obr. 14 volíme $x_M = 60{,}13R_Z$, $y_M = 0$. Současně je také zřejmé, že $y_{SM} = y_{SZ}$.

Poněvadž gravitační působení Měsíce je mnohem menší než gravitační působení Země, je nutné přesněji určit počáteční hodnoty rychlosti, při nichž sonda buď Měsíc obletí a vrátí se zpět na Zemi, nebo se přiblíží k Měsíci a po opakovaném obletu Země na Měsíci přistane. Je třeba si také uvědomit, jaká situace nastane při střetu sondy se Zemí nebo Měsícem. Střet se projeví velkým zvětšením rychlosti sondy, která se začne z uvažované modelové soustavy po

přímočaré trajektorii vzdalovat. Proto je třeba při realizaci počítačového modelu zajistit, aby se po přiblížení sondy na určitou vzdálenost k cíli další pohyb sondy zastavil.

Z obdobné úvahy jako v příkladu 1.7 vyplývá, že složky zrychlení sondy způsobené gravitačním působením Země budou mít souřadnice

$$a_{x\mathrm{Z}} = -x_{\mathrm{SZ}} M_{\mathrm{Z}} / r^3 ,$$
$$a_{y\mathrm{Z}} = -y_{\mathrm{SZ}} M_{\mathrm{Z}} / r^3 .$$

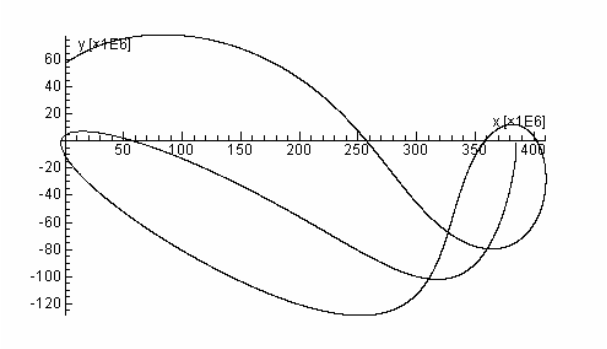
Obdobně souřadnice složek zrychlení způsobené gravitačním působením Měsíce budou

$$a_{x\mathrm{M}} = -x_{\mathrm{SM}} M_{\mathrm{M}} / r_{\mathrm{M}}^3 ,$$
$$a_{y\mathrm{M}} = -x_{\mathrm{SM}} M_{\mathrm{M}} / r_{\mathrm{M}}^3 .$$

Souřadnice zrychlení ve výsledném počítačovém modelu budou určeny součtem souřadnic obou složek. Konkrétní model je v tabulce. V něm jsou také uvedeny hodnoty souřadnic počáteční rychlosti při startu sondy z parkovací kružnicové trajektorie o poloměru $10R_Z$. Tvar trajektorie kosmické sondy pro tento případ je na obr. 15.

Model	Proměnné, konstanty, počáteční hodnoty

`x=x+vx*h`
`y=y+vy*h`
`r=sqrt(x*x+y*y)`
`ax=-k*x*Mz/r^3`
`ay=-k*y*Mz/r^3`
`az=sqrt(ax*ax+ay*ay)`
`xx=x-xm`
`yy=y-ym`
`rm=sqrt(xx*xx+yy*yy)`
`axm=-k*Mm/rm^2`
`aym=-k*Mm/rm^2`
`am=sqrt(axm*axm+aym*aym)`
`vx=vx+ax*h+axm*xx*h/rm`
`vy=vy+ay*h+aym*yy*h/rm`
`t=t+h`
`if rm<1e6 then stop endif`
	`k=6.67e-11`
	`Mz=5.98e24`
	`rz=6.378e6`
	`h=100`
	`Mm=7.374e23`
	`Rm=3.844e8`
	`x=0`
	`y=9*rz`
	`xm=60.13*rz`
	`ym=0`
	`vy=1.78e3`
	`vx=2.83e3`
	`t=0`


y [x1E6]
60
40
20
-20
-40
-60
-80
-100
-120
50
100
150
200
250
300
350
x [x1E6]

Obr. 15

2 Mechanický oscilátor

2.1 Vlastní kmitání mechanického oscilátoru

Vlastní kmitání mechanického oscilátoru je takové, že v počátečním okamžiku vložíme do soustavy oscilátoru určitou energii (nejčastěji vychýlením oscilátoru z rovnovážné polohy) a pak oscilátor volně kmitá bez vnějšího působení. Pomocí dynamického modelu řešíme obvykle případ netlumeného oscilátoru a tlumeného oscilátoru.

a) Netlumený oscilátor

Netlumený oscilátor je fyzikální abstrakce harmonického oscilátoru, jehož pohybová rovnice má tvar (pro případ pohybu oscilátoru ve směru osy y)

$$ma = -ky,$$

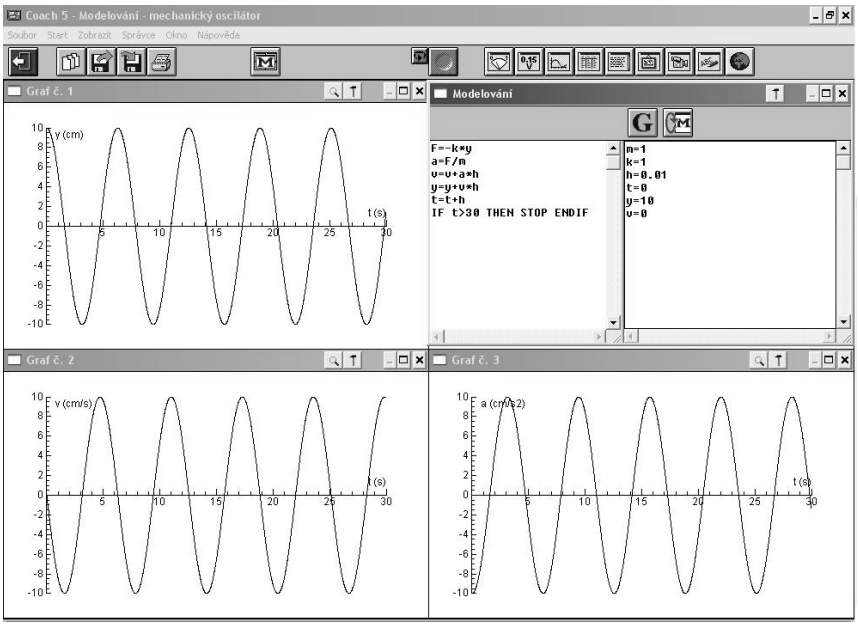
kde hmotnost m a tuhost k jsou parametry oscilátoru. To znamená, že v dynamickém modelu vyjádříme souřadnici zrychlení rovnicí

$$a = -ky/m.$$

Model netlumeného oscilátoru je v tabulce. Časové diagramy okamžité výchylky, rychlosti a zrychlení oscilátoru jsou na obr. 16.

Model	Proměnné, konstanty, počáteční hodnoty

`F=-k*y`
`a=F/m`
`v=v+a*h`
`y=y+v*h`
`t=t+h`
`if t>=30 then stop endif`
	`m=1`
	`k=1`
	`h=0.01`
	`t=0`
	`y=10`
	`v=0`



Obr. 16

b) Tlumený oscilátor

Kmitání reálného mechanického oscilátoru (tělesa zavěšeného na pružině) je vždy tlumené, poněvadž proti jeho pohybu působí tlumící síla F_t. Tato síla je při malých rychlostech lineární funkcí rychlosti: $F_t = -bv$, kde b je součinitel úměrnosti tlumící síly a rychlosti. Pohybová rovnice má tvar

$$ma = -ky - bv$$

a v počítačovém modelu zapíšeme souřadnici zrychlení rovnicí:

```
a = -(ky + bv)/m
```

Dynamický model umožňuje řešit také zvláštní případy pohybu oscilátoru. Nejdůležitější je případ kritického tlumení, který je mezním případem tlumené-

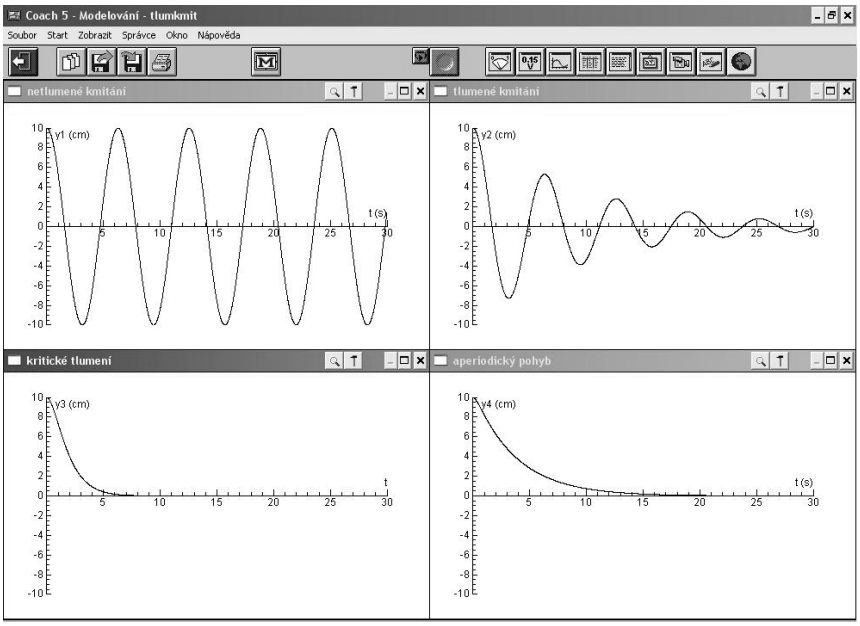
ho kmitání, kdy oscilátor přestává kmitat a v nejkratší době přejde do rovnovážné polohy. Z teorie kmitání pro tento případ plyne vztah

$$\omega_0 = \delta,$$

kde ω_0 je úhlová frekvence vlastních kmitů netlumeného oscilátoru ($\omega_0 = \sqrt{k/m}$) a δ je součinitel tlumení ($\delta = b/2m$). To znamená, že kritické tlumení pohybu oscilátoru nastane, když součinitel b bude mít hodnotu $b_k = 2\sqrt{km}$. Je-li součinitel tlumení větší ($b > b_k$), je pohyb oscilátoru aperiodický, oscilátor se zvolna vrací do rovnovážné polohy.

Počítačový model tlumeného oscilátoru je v tabulce a na obr. 17 jsou znázorněny časovými závislostmi okamžité výchylku pro jednotlivé případy (netlumený oscilátor, tlumený oscilátor, kritické tlumení, aperiodický pohyb).

Model	Proměnné, konstanty, počáteční hodnoty
F1=-k*y1-b1*v1 a1=F1/m v1=v1+a1*h y1=y1+v1*h F2=-k*y2-b2*v2 a2=F2/m v2=v2+a2*h y2=y2+v2*h F3=-k*y3-b3*v3 a3=F3/m v3=v3+a3*h y3=y3+v3*h F4=-k*y4-b4*v4 a4=F4/m v4=v4+a4*h y4=y4+v4*h t=t+h if t>30 then stop endif	m=1 k=1 b1=0 b2=0.2 b3=2 b4=4 h=0.01 t=0 y1=10 y2=10 y3=10 y4=10 v1=0 v2=0 v3=0 v4=0



Obr. 17

2.2 Nucené kmitání mechanického oscilátoru

Jde o případ kmitání, kdy vnější síla F_v působící na mechanický oscilátor je periodickou funkcí času

$$F_\mathrm{v} = F_\mathrm{m}\sin\Omega t \ ,$$

kde F_m je amplituda vnější síly a Ω je její úhlová frekvence. Pohybová rovnice má tvar

$$ma = -ky - bv + F_\mathrm{v} \ .$$

Při nuceném kmitání oscilátor koná netlumené kmity s úhlovou frekvencí Ω a jejich amplituda Y_m závisí na parametrech oscilátoru (tzn. na úhlové frek-

venci vlastního kmitání), součiniteli b a úhlové frekvenci Ω nuceného kmitání. Z teorie vyplývá pro okamžitou výchylku nuceného kmitání vztah

$$y = Y_{\mathrm{m}} \sin\left(\Omega t + \gamma\right),$$

kde

$$Y_{\mathrm{m}} = \frac{F_{\mathrm{m}}/m}{\sqrt{\left(\omega^2 - \Omega^2\right)^2 - 4\delta^2\Omega^2}}$$

a γ je fázový posun nucených kmitů vzhledem k fázi vnější síly. Platí pro něj vztah

$$\operatorname{tg}\gamma = -\frac{2\delta\Omega}{\omega^2 - \Omega^2}.$$

Při rezonanci dosahuje amplituda nucených kmitů největší hodnoty a u oscilátoru s úhlovou frekvencí vlastního kmitání ω je určena vztahem

$$Y_{\mathrm{m}} = \frac{F_{\mathrm{m}}}{2m\delta\omega}.$$

Tyto poměrně složité teoretické výsledky interpretující kmitavé děje v různých druzích oscilátorů lze lépe pochopit na základě dynamického modelu, v němž vztah pro zrychlení, potřebný k vytvoření dynamického modelu, najdeme pro každý konkrétní případ z pohybové rovnice. Ostatní rovnice dynamického modelu mají stejný tvar jako v předcházejících případech. Model je v tabulce.

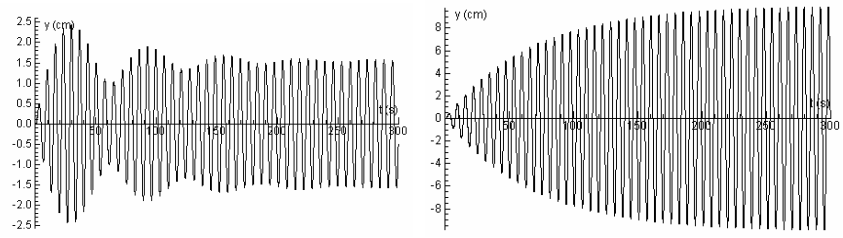
Pro numerické řešení pohybu oscilátoru musíme znát hodnoty všech veličin ve vztazích (m, k, b, F_{m}, Ω) a počáteční podmínky (např. $t = 0$ a $y = 0$, $v = 0$). V tomto případě vlastně oscilátor v počátečním okamžiku nekmitá a jde tedy o přechodný děj, kdy oscilátor působením vnější harmonické síly přechází do kmitavého pohybu. Oscilátor začíná kmitat postupně a teprve po určité době nastane ustálený stav, kdy oscilátor kmitá s konstantní amplitudou výchylky.

To je patrné z obr. 18 (vlevo), na kterém je časový diagram odpovídající případu, kdy frekvence f nuceného kmitání je menší než rezonanční frekvence

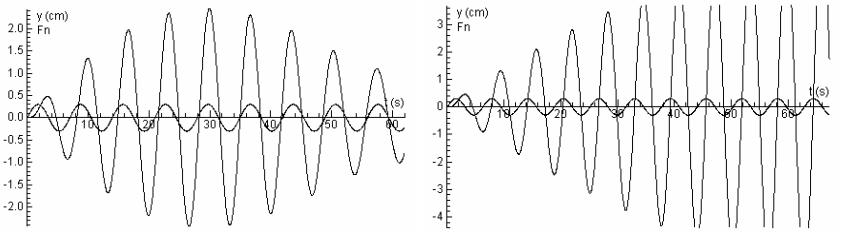
f_{rez} ($f = 0{,}9\, f_{rez}$, $F_m = 0{,}1$, $b = 0{,}08$). Na obr. 18 (vpravo) je časový diagram v případě, kdy $f = f_{rez}$. Kdyby oscilátor nebyl tlumen, tzn. když $b = 0$, narůstá amplituda kmitů lineárně bez omezení. Na obr. 19 jsou časové diagramy pro stejné případy jako na obr. 18, ale současně je znázorněn časový průběh vnější síly. Snadno si tak např. ověříme, že při rezonanci, kdy $\omega = \Omega$, je $\gamma = \pi/2$, což je z časového diagramu dobře patrné.

Model	Proměnné, konstanty, počáteční hodnoty

`Fv=Fm*sin(w*t)`
`F=-k*y-b*v+Fv`
`a=F/m`
`y=y+v*h`
`v=v+a*h`
`t=t+h`
	`m=1`
	`k=1`
	`w=0.9`
	`Fm=0.3`
	`b=0.06`
	`t=0`
	`y=0`
	`v=0`
	`h=0.03`



Obr. 18

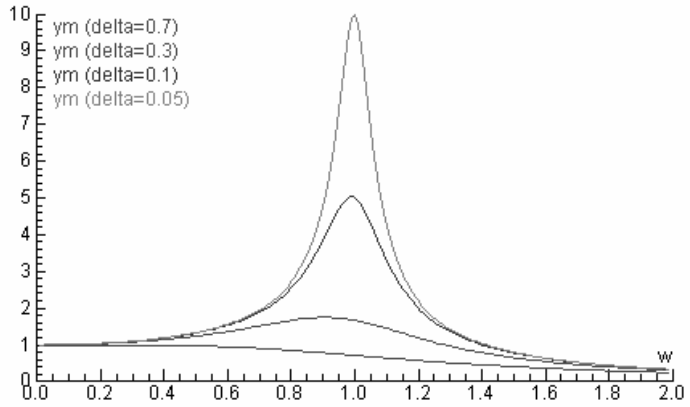


Obr. 19

Model doplníme zobrazením rezonanční křivky nuceného kmitání pro různé hodnoty tlumení. V modelu je prováděn výpočet amplitudy výchylky pro různé hodnoty součinitele δ ($\delta = b/2m$). Ukážeme, že při kritickém tlumení již křivka nemá maximum a amplituda se při všech úhlových frekvencích jen zmenšuje. Pro nucené kmitání musí být splněna podmínka $\omega^2 = 2\delta^2$, takže při rezonanční frekvenci $\omega = 1\ \mathrm{s}^{-1}$ nastane kritické tlumení pro $\delta = 1/\sqrt{2} \approx 0{,}7\ \mathrm{s}^{-1}$. Rezonanční křivky pro některé hodnoty součinitele tlumení jsou na obr. 20.

Model	Proměnné, konstanty, počáteční hodnoty

`w=w+dw`
`ym=F0/m/sqrt((w0^2-w^2)^2+4*w^2*delta^2)`
`wait(0.02)`
`if w>2 then stop endif`
	`dw=0.01`
	`F0=1`
	`m=1`
	`w0=1`
	`w=0`
	`delta=0.05`



Obr. 20

2.3 Pružinové kyvadlo

Jako pružinové kyvadlo pro tento model označíme oscilátor tvořený pružinou se závažím, který může kývat kolem bodu závěsu. Může tak vykonávat v podstatě dva mody kmitů – podélné a příčné. Na závaží působí jednak tíhová síla, jejíž souřadnice ve směru osy y je $F_{Gy} = -mg$, jednak síla pružnosti F_{p}. Budeme uvažovat pohyb kyvadla v rovině a tedy zobrazovat souřadnice x a y závaží. Souřadnice sil pružnosti vyjadřují rovnice

$$F_{\mathrm{p}x} = -k(r-l)\frac{x}{r},$$

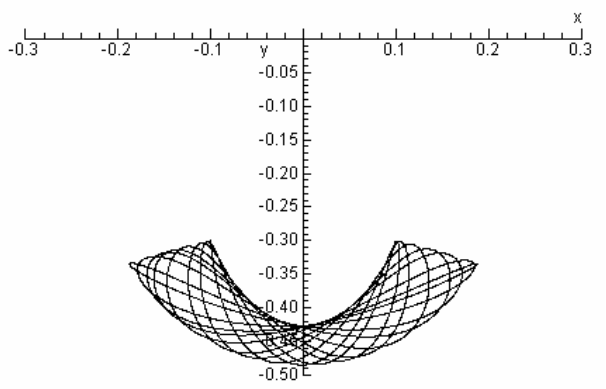
$$F_{\mathrm{p}y} = -k(r-l)\frac{y}{r},$$

kde $r = \sqrt{x^2 + y^2}$ je vzdálenost od bodu upevnění kyvadla, který je i počátkem vztažné soustavy *Oxy*.

Model pohybu pružinového kyvadla je v tabulce a graf pohybu pro hodnoty uvedené v modelu je na obr. 21.

Model	Proměnné, konstanty, počáteční hodnoty

`r=sqrt(x^2+y^2)`
`ax=-d*(r-l)/r*x`
`ay=-g-d*(r-l)/r*y`
`vx=vx+ax*h`
`vy=vy+ay*h`
`x=x+vx*h`
`y=y+vy*h`
`t=t+h`
`if t>20 then stop endif`
	`x=-0.1`
	`y=-0.3`
	`l=0.3`
	`k=11.5`
	`g=9.81`
	`h=0.001`
	`m=0.12`
	`d=k/m`
	`vx=0`
	`vy=0`
	`t=0`



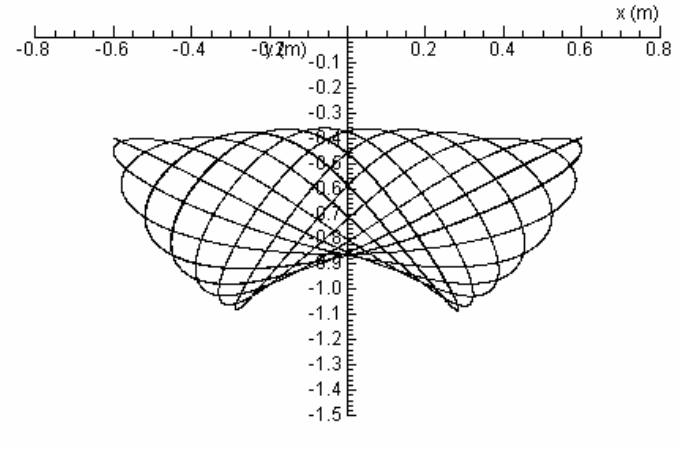
Obr. 21

2.4 Kyvadlo s gumovým vláknem

V modelu se uvažuje kyvadlo tvořené kuličkou o hmotnosti m, která je zavěšena na gumovém vlákně, které má bez zatížení délku l (viz [7], s. 32). Vlákno je dokonale pružné, má tuhost k a síla pružnosti je lineární funkcí prodloužení. Nebudeme uvažovat žádné další silové působení, které by mohlo pohyb kyvadla ovlivnit (odpor prostředí, moment setrvačnosti, rotaci kuličky). Jestliže kyvadlo vychýlíme bez prodloužení závěsu, kmitá jen působením tíhové síly $F_G = mg$. Při současném prodloužení závěsu se ještě projeví síla pružnosti $F_p = -(r - l)$, kde $r = \sqrt{x^2 + y^2}$ je vzdálenost kuličky od bodu závěsu. Tato síla na kuličku působí jen v případě, že $r > l$. V případě, že $r < l$, působí na kuličku jen tíhová síla. Za těchto podmínek má model podobu, která je v tabulce. Pohyb kuličky podstatným způsobem závisí na počátečních podmínkách a trajektorie kuličky vzniká superpozicí příčných a podélných kmitů (obr. 22).

Model	Proměnné, konstanty, počáteční hodnoty

r=sqrt(x^2+y^2)
IF r>l then
ax=-k*(r-l)/r*x/m
ay=-g-k*(r-l)/r*y/m
else
ax=0
ay=-g
endif
vx=vx+ax*h
vy=vy+ay*h
x=x+vx*h
y=y+vy*h
wait(0.001)
t=t+h
if t=30 then stop endif
	m=0.05
	l=0.5
	k=2
	g=9.81
	h=0.001
	x=-0.6
	y=-0.4
	vx=0
	vy=0
	t=0



Obr. 22

2.5 Rotátor

Jako rotátor označíme kyvadlo tvořené tuhou tyčí, jejíž hmotnost je malá ve srovnání s hmotností závaží na konci tyče délky *l*. Na rozdíl od matematického kyvadla, které má vlastnosti lineárního oscilátoru, pokud úhel rozkyvu nepřekročí 5°, rotátor je kyvadlo s velkou výchylkou, která může mít úhlovou dráhu φ až 180°, popř. i větší, takže kyvadlo rotuje, přičemž jeho úhlová rychlost není konstantní. Pohyb rotátoru působením vnější vynucující síly může vést k jeho chaotickému pohybu. V této souvislosti se rotátor používá jako jeden z modelových objektů pro objasnění jevů souvisejících s deterministickým chaosem (viz [8]).

V našem modelu budeme uvažovat pohyb způsobený tečnou složkou tíhové síly do směru pohybu

$$F = - mg \sin \varphi ,$$

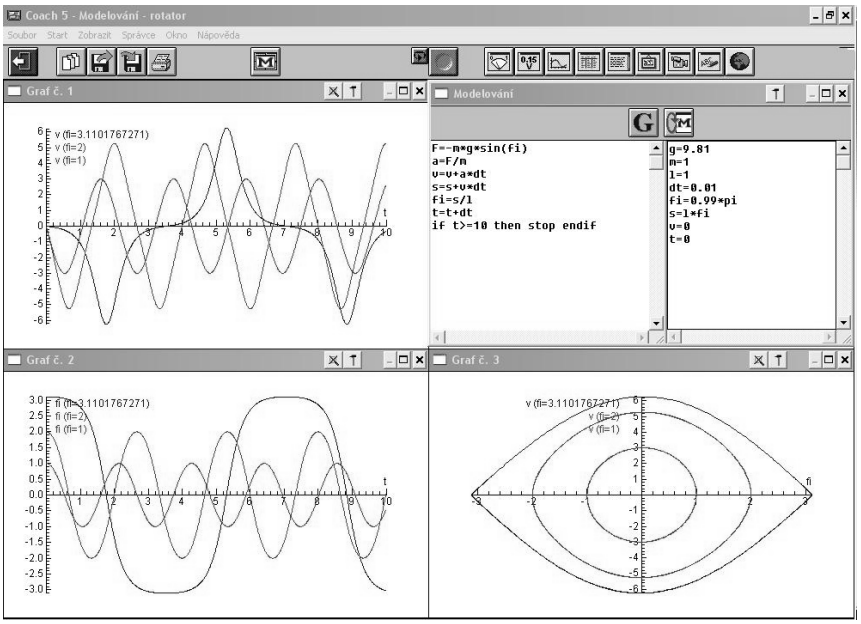
kde φ je úhlová dráha pohybu kyvadla. Jestliže polohu závaží kyvadla na kruhovém oblouku označíme *s*, platí:

$$\varphi = \frac{s}{l}$$

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} = -mg \sin \varphi$$

Model pohybu rotátoru je v tabulce a z grafů na obr. 23 je patrné, že při velké počáteční výchylce je pohyb rotátoru anharmonický. Jako největší výchylka bylo zvoleno $\varphi = 0{,}99\pi$. Pokud bychom zvolili $\varphi = \pi$, rotátor zůstane v klidu v labilní poloze, kdy je závaží v nejvyšší poloze nad bodem závěsu. Aby se kyvadlo začalo pohybovat, musí být výchylka nepatrně menší, popř. větší (zkuste např. $\varphi = 1{,}01\pi$). Nejzajímavější je graf č. 3 (vpravo dole), který zobrazuje pohyb rotátoru v tzv. fázovém prostoru. Vidíme, že při malých počátečních výchylkách je fázovým obrazem netlumeného pohybu rotátoru (atraktorem) kružnice, kdežto při velké výchylce má atraktor tvar dvou spojených oblouků.

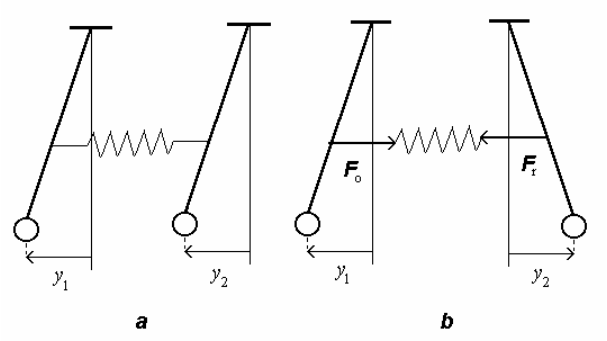
Model	Proměnné, konstanty, počáteční hodnoty
F=-m*g*sin(fi) a=F/m v=v+a*h s=s+v*h fi=s/l omega=v/l t=t+h if t>=10 then stop endif	g=9.81 m=1 l=1 h=0.01 fi=0.99*pi s=l*fi v=0 t=0



Obr. 23

2.6 Vázané mechanické oscilátory

Jde o soustavu dvou oscilátorů (oscilátor a rezonátor), v níž při kmitavém pohybu na sebe oscilátory navzájem působí periodickými silami. Budeme uvažovat případ, kdy síly vzájemné vazby jsou funkcí výchylky oscilátoru (spřažení výchylkou). Pro jednoduchost budeme také předpokládat, že oscilátor i rezonátor mají stejné parametry ($m_1 = m_2 = m$, $k_1 = k_2 = k$).



Obr. 24

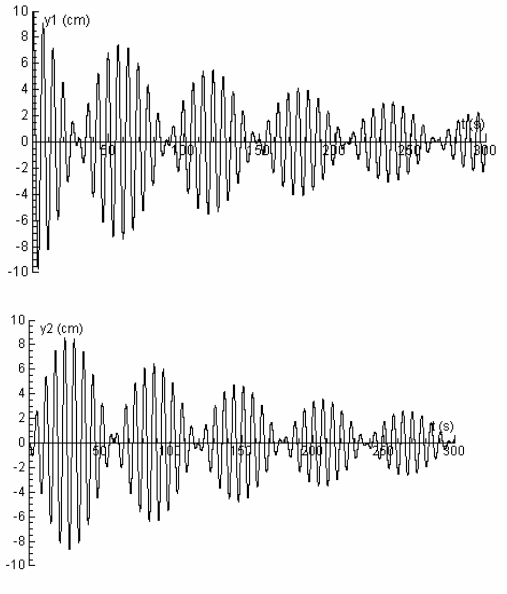
Vázané oscilátory lze znázornit jako dvojici spřažených kyvadel (obr. 24). Jejich kmitání si můžeme představit jako superpozici dvou modů oscilací. První mod odpovídá stejnosměrnému (symetrickému) kmitání (obr. 24a), kdy se spřažení neprojeví a oscilátory kmitají tak, jako by mezi nimi vazba nebyla. Druhý mod představuje protisměrné (antisymetrické) kmitaní (obr. 24b), kdy vznikají v důsledku spřažení síly, které jsou funkcí rozdílu výchylek obou oscilátorů. Poněvadž jde o síly akce a reakce, má síla $\boldsymbol{F}_O$ působící na oscilátor opačný směr než síla $\boldsymbol{F}_R$ působící na rezonátor. Pohybové rovnice tlumeného oscilátoru a rezonátoru mají tvar

$$ma_1 = -ky_1 - by_1 - c(y_1 - y_2),$$
$$ma_2 = -ky_2 - by_2 - c(y_2 - y_1),$$

kde c je součinitel vzájemné vazby. Z této rovnice snadno najdeme tvary rovnice pro souřadnici zrychlení pohybu vázaných oscilátoru, z níž sestavíme počítačový model (viz tabulku). Průběh kmitání je patrný z obr. 25, který řeší případ, kdy v počátečním okamžiku je výchylka oscilátoru rovna amplitudě vý-

chylky a výchylka rezonátoru je nulová (v počítačovém modelu volíme $y_1 = 10$ cm, $y_2 = 0$, $v_1 = 0$, $v_2 = 0$).

Model	Proměnné, konstanty, počáteční hodnoty
F1=-k*y1-b*v1-c*(y2-y1) F2=-k*y2-b*v2-c*(y1-y2) a1=F1/m1 a2=F2/m2 v2=v2+a2*h v1=v1+a1*h y1=y1+v1*h y2=y2+v2*h t=t+h	m1=1 m2=1 k=1 c=0.1 b=0.01 h=0.02 t=0 y1=10 y2=0 v1=0 v2=0



Obr. 25

3 Přechodné děje v elektrických obvodech

Postup při dynamickém modelování dějů v elektrických obvodech je analogický jako při vytváření modelů mechanických dějů. Pro pochopení těchto modelů je nejvhodnější případ sériového spojení obvodových prvků s parametry R, L a C. Děje v těchto obvodech rozdělíme do čtyř skupin podle zdroje napětí, ke kterému je elektrický obvod připojen:

(I) Obvod je připojen ke zdroji konstantního elektromotorického napětí

$$U_0 = \text{ konst.}$$

(II) Obvod je připojen ke kondenzátoru o kapacitě C, který je nabit nábojem Q, tzn. že napětí v počátečním okamžiku

$$u_C = \frac{Q}{C}.$$

(III) Obvod je připojen k cívce, kterou prochází elektrický proud, a při změně velikosti proudu vzniká na cívce indukované napětí

$$u_L = -L\frac{\mathrm{d}i}{\mathrm{d}t}.$$

(IV) Obvod je připojen ke zdroji harmonického napětí s konstantní amplitudou $U_0 = \text{konst.}$

$$u = U_0 \sin \omega t\,.$$

Všechny uvedené případy lze chápat jako přechodné děje, kdy obvod přechází z počátečního ustáleného stavu do následujícího ustáleného stavu. Nejčastěji jsou řešeny případy:

(i) V počátečním okamžiku proud obvodem neprochází (obvod je rozpojen) a po připojení ke zdroji stálého napětí vzniká v obvodu ustálený proud.

(ii) Kondenzátor odpojený od elektrického obvodu je nabit ze zdroje stejnosměrného napětí a pak je připojen k obvodu s odporem *R*, popř. s *RL*.

(iii) Obvodem s *RL* prochází elektrický proud a paralelně k cívce je připojen obvod s *C*, popř. s *RC*. Po přerušení proudu procházejícího cívkou se na cívce indukuje napětí, kterým se nabíjí kondenzátor v paralelním obvodu.

(iv) Obvod je v počátečním okamžiku připojen ke zdroji harmonického napětí a přechodným dějem vzniká v obvodu střídavý proud fázově posunutý vzhledem k napětí.

Východiskem pro vytvoření modelu příslušného děje je zápis 2. Kirchhoffova zákona pro daný obvod, který je při modelování dějů v elektrických obvodech obdobou pohybové rovnice při popisu mechanického děje. Na základě rovnice pro elektrický obvod je určen vztah pro přírůstek proudu d*i* (obdoba přírůstku rychlosti za časový krok *h* v modelu mechanického děje: *ah*) a další rovnice modelu určují okamžitou hodnotu elektrického náboje kondenzátoru, proudu v obvodu a změnu časového kroku:

$$q_{i+1} = q_i + i_i h\,, \tag{1}$$

$$i_{i+1} = i_i + \mathrm{d}i\,, \tag{2}$$

$$t_{i+1} = t_i + h\,. \tag{3}$$

3.1 Přechodný děj v obvodu *RC*

Přechodný děj v obvodu *RC* představuje nabíjení kondenzátoru (odpovídá případu (i)) a vybíjení kondenzátoru (případ (ii)). Děj nabíjení kondenzátoru je popsán na základě 2. Kirchhoffova zákona rovnicí

$$u_R + u_C = U\,,$$

čili

$$Ri + \frac{q}{C} = U\,.$$

Odtud vyplývá pro okamžitou hodnotu proudu i vztah

$$i = \frac{U - \frac{q}{C}}{R}$$

a počítačový model je tvořen rovnicemi (za předpokladu, že počáteční podmínky jsou: $q = 0$, $i = U/R$):

```
q = q + i*h
i = ((U - q/C)/R
t = t + h
```

Při vybíjení kondenzátoru je zdrojem napětí nabitý kondenzátor a platí rovnice

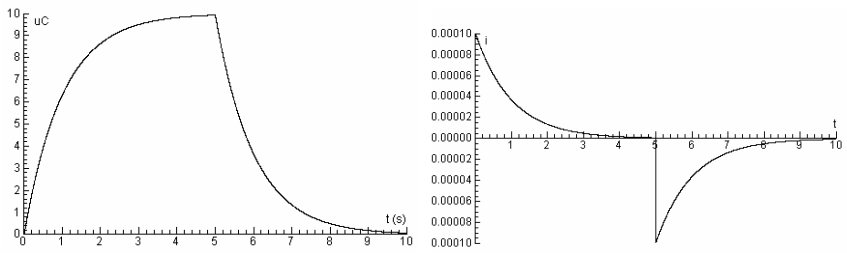
$$u_R + u_C = 0\,,$$

takže rovnice (2) má tvar

$$i = -\frac{q}{RC}\,.$$

Rovnice (1) a (3) se nemění. Model přechodného děje v obvodu RC je v tabulce. Model je upraven tak, že zobrazuje jak nabíjení, tak vybíjení kondenzátoru. Grafy u_C a i při přechodném ději jsou na obr. 26.

Model	Proměnné, konstanty, počáteční hodnoty
t=t+h uC=q/C in=(U-uC)/R1 iv=-uC/R2 if t<5 then i=in else i=iv endif q=q+i*h if t>10 then stop endif	C=1e-5 R1=1e5 R2=1e5 U=10 h=0.001 q=0 t=0



Obr. 26

3.2 Přechodný děj v obvodu *RL*

Přechodný děj v obvodu *RL* představuje připojení sériového obvodu tvořeného cívkou s indukčností *L* a odporem *R* ke zdroji stejnosměrného napětí, popř. přerušení obvodu a připojení rezistoru k cívce paralelně. To je vyjádřeno rovnicemi

$$u_R + u_L = U$$

a

$$u_R + u_L = 0\,.$$

Dosazením vztahů pro jednotlivá napětí ($u_R = Ri$, $u_L = L\mathrm{d}i/\mathrm{d}t$) dostaneme pro přírůstek proudu d*i* vztahy (místo d*t* dosadíme časový krok *h*)

```
di = (U - R*i)*h/L,
```

popř.

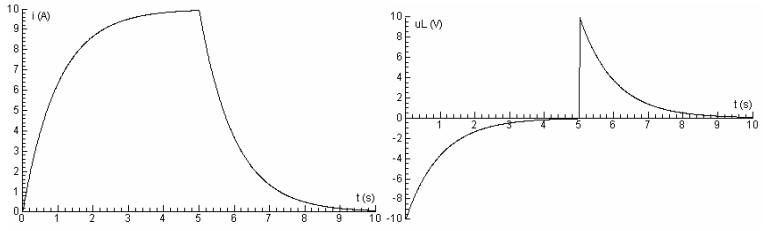
```
di = -R*i*h/L.
```

Model přechodného děje je v tabulce a na obr. 27 jsou grafy jeho časového průběhu. Volíme hodnoty $L = 0{,}1$ H a $R = 10\ \Omega$.

Pro přechodné děje v obvodu *RC*, popř. *RL* je charakteristickou veličinou časová konstanta $\tau = RC$, popř. $\tau = L/R$, za kterou $u_C = 0{,}63U$, popř. $u_L = 0{,}37U$. Hodnoty *R* a *C*, popř. *R* a *L* jsou v modelovaných příkladech voleny tak, aby časová konstanta obou obvodů byla stejná, tzn. $\tau = 1$ s.

Model	Proměnné, konstanty, počáteční hodnoty

`if t<5 then`
`di=(U-R*i)/L*h`
`else`
`di=-R*i/L*h`
`endif`
`i=i+di`
`uL=-L*di/h`
`t=t+h`
`if t>10 then stop endif`
	`L=1`
	`U=10`
	`h=0.01`
	`R=1`
	`i=0`
	`t=0`
	`uL=0`



Obr. 27

3.3 Přechodný děj v obvodu *RLC*

Uvažujeme obvod s *RLC* v sérii, který je připojen ke zdroji stejnosměrného napětí *U*. Rovnice vyjadřující 2. Kirchhoffův zákon má tvar

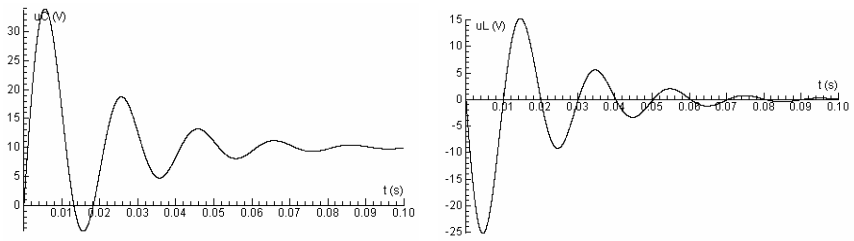
$$u_R + u_L + u_C = U$$

a pro přírůstek proudu d*i* platí

$$\texttt{di = (U - R*i - q/C)*h/L.}$$

Poněvadž obvod obsahuje jak indukčnost, tak kapacitu, jsou splněny podmínky pro vznik oscilací a obvod je vlastně elektromagnetickým tlumeným oscilátorem. Model je v tabulce a na obr. 28 jsou grafy časového průběhu napětí na kondenzátoru a na cívce.

Model	Proměnné, konstanty, počáteční hodnoty
q=C*uC q=q+i*h uC=q/C uR=R*i uL=U0-uR-uC i=i+((U0-uR-uC)/L)*h t=t+h if t>0.1 then stop endif	C=1e-5 L=1 R=100 'R=sqrt(4*L/C) 'R=632 h=1E-5 U=10 U0=U uC=0 i=U0/R t=0



Obr. 28

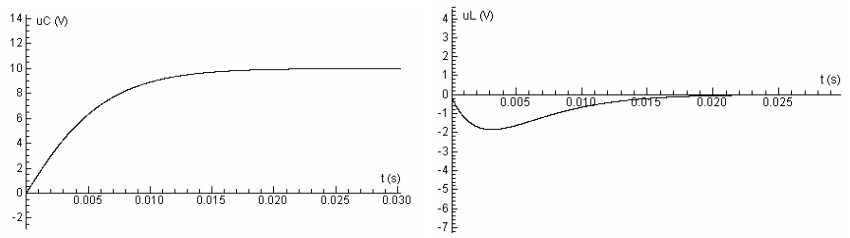
Poněvadž obvod *RLC* má vlastnosti oscilačního obvodu, má také úhlovou frekvenci vlastního kmitání ω. Pro tlumený oscilační obvod platí

$$\omega = \sqrt{\omega_0^2 - \delta^2} = \sqrt{\frac{1}{LC} - \frac{R^2}{4L^2}} ,$$

kde $\omega_0 = 1/\sqrt{LC}$ je úhlová frekvence vlastního kmitání netlumeného elektromagnetického oscilátoru a $\delta = R^2/4L^2$ je součinitel tlumení. Kritické tlumení nastane, když $\omega_0 = \delta$, a z této podmínky vyplývá hodnota odporu, při němž se kmitání obvodu v nejkratší době utlumí

$$R_{\mathrm{k}} = \sqrt{\frac{4L}{C}} \ .$$

V našem modelu je $R_{\mathrm{k}} = 632\ \Omega$. Na obr. 29 jsou grafy přechodného děje s kritickým tlumením.



Obr. 29

4 Elektromagnetický oscilátor

Již při výkladu přechodných dějů v elektrických obvodech jsme dospěli k modelu elektromagnetického oscilátoru. V této kapitole se zaměříme na srovnání elektromagnetického oscilátoru s mechanickým oscilátorem.

4.1 Vlastní kmitání elektromagnetického oscilátoru

Základním typem elektromagnetického oscilátoru je oscilační obvod s parametry indukčnost L, kapacita C, odpor R. Děje v takovém obvodu jsou obdobně jako u mechanického oscilátoru popsány diferenciálními rovnicemi a mezi modely těchto dějů jsou velmi těsné analogie. Pro numerické řešení diferenciálních rovnic popisujících elektromagnetické kmitání tedy použijeme stejný algoritmus a příslušné rovnice dynamického modelu snadno přepíšeme na základě vzájemných analogií, uvedených v následující tabulce:

mechanický oscilátor	elektromagnetický oscilátor
y	q
$v = \mathrm{d}v/\mathrm{d}t$	$i = \mathrm{d}q/\mathrm{d}t$
a	$\mathrm{d}i/\mathrm{d}t$
b	R
m	L
k	$1/C$
F	U

Pomocí této tabulky můžeme "přeložit" vztahy pro síly v mechanickém oscilátoru a získáme vztahy pro napětí na jednotlivých obvodových prvcích elektromagnetického oscilátoru:

$F = ma$	$u_L = L\dfrac{\mathrm{d}i}{\mathrm{d}t}$
$F_\mathrm{r} = bv$	$u_R = Ri$
$F_\mathrm{p} = ky$	$u_C = \dfrac{q}{C}$

Pohybovou rovnici při modelování elektromagnetického oscilátoru nahrazuje 2. Kirchhoffův zákon. Jestliže uvažujeme uzavřený oscilační obvod, jehož kondenzátor byl v počátečním okamžiku nabit (ale obvod není připojen k žádnému zdroji elektromotorického napětí), bude mít rovnice vyjadřující 2. Kirchhoffův zákon tvar

$$u_L + u_R + u_C = 0\,,$$

čili

$$L\frac{\mathrm{d}i}{\mathrm{d}t} + Ri + \frac{q}{C} = 0\,.$$

Odtud najdeme vztah $\mathrm{d}i/\mathrm{d}t$ odpovídající rovnici (1) u mechanického oscilátoru a můžeme přikročit k vytvoření dynamického modelu, který bude posloupností rovnic (diferenciál dt nahradíme časovým krokem h):

```
q = q + i*h
i = i + di
t = t + h
```

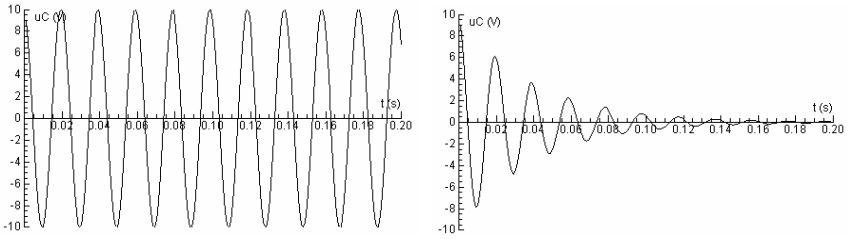
V případě netlumeného oscilátoru ($R = 0$) je $\mathrm{d}i = -(q/LC)h$ a u tlumeného oscilátoru je $\mathrm{d}i = -(q/C + Ri)h/L$. V modelu řadíme rovnice v pořadí:

```
di, i, q, t
```

Model obsahující jak variantu netlumeného, tak tlumeného elektromagnetického oscilátoru je v tabulce. Grafy napětí na obvodu jsou na obr. 30.

Model	Proměnné, konstanty, počáteční hodnoty

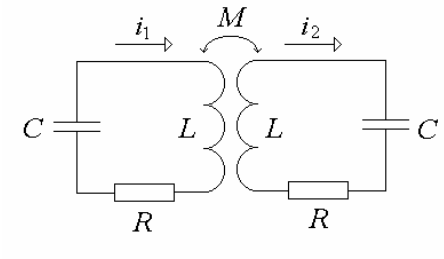
`di=-q*h/(L*C)`
`'di=-((q/C)+R*i)*h/L`
`i=i+di`
`q=q+i*h`
`uC=q/C`
`t=t+h`
	`C=1e-4`
	`L=1`
	`R=10`
	`U=10`
	`q=U*C`
	`h=0.001`
	`t=0`
	`i=0`



Obr. 30

4.2 Vázané elektromagnetické oscilátory

Poněkud složitější případ představují vázané elektromagnetické oscilátory, které můžeme znázornit jako soustavu oscilačních obvodů vázaných indukční vazbou (obr. 31).



Obr. 31

Vlivem vzájemné indukce vzniká v rezonátoru napětí, které je funkcí časové změny proudu v oscilátoru a naopak. Jestliže označíme činitel vzájemné vazby mezi oscilátorem a rezonátorem M, můžeme napsat 2. Kirchhoffův zákon pro vázané oscilátory ve tvaru:

$$L\frac{\mathrm{d}i_1}{\mathrm{d}t} + M\frac{\mathrm{d}i_2}{\mathrm{d}t} + Ri_1 + \frac{q_1}{C} = 0$$

$$L\frac{\mathrm{d}i_2}{\mathrm{d}t} + M\frac{\mathrm{d}i_1}{\mathrm{d}t} + Ri_2 + \frac{q_2}{C} = 0$$

Z těchto rovnic najdeme úpravou vztahy:

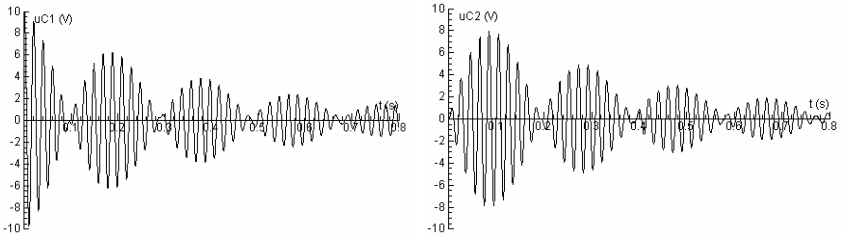
$$\mathrm{d}i_1 = \frac{Lq_1 - Mq_2 - RC(Li_1 - Mi_2)}{C(M^2 - L^2)}\,\mathrm{d}t$$

$$\mathrm{d}i_2 = \frac{Lq_2 - Mq_1 - RC(Li_2 - Mi_1)}{C(M^2 - L^2)}\,\mathrm{d}t$$

Tyto vztahy použijeme k vytvoření počítačového modelu analogického vázaným mechanickým oscilátorům (viz tabulku). Časový diagram kmitání oscilátoru (vlevo) a rezonátoru (vpravo) je na obr. 32.

Model	Proměnné, konstanty, počáteční hodnoty

`q1=q1+i1*h`
`q2=q2+i2*h`
`uC1=q1/C`
`uC2=q2/C`
`uR1=R*i1`
`uR2=R*i2`
`i1=i1-(L1*(uC1+uR1)-`
`M*(uC2+uR2)/(L1^2-M^2))*h`
`i2=i2-(L2*(uC2+uR2)-`
`M*(uC1+uR1)/(L2^2-M^2))*h`
`t=t+h`
`wait(0.01)`
`if t>0.8 then stop endif`
	`C=1e-5`
	`R=5`
	`M=0.1`
	`U=10`
	`L1=1`
	`L2=1`
	`uC1=U`
	`uC2=0`
	`i1=0`
	`i2=0`
	`t=0`
	`h=0.001`
	`q1=C*uC1`
	`q2=C*uC2`



Obr. 32

4.3 Nucené kmitání elektromagnetického oscilátoru

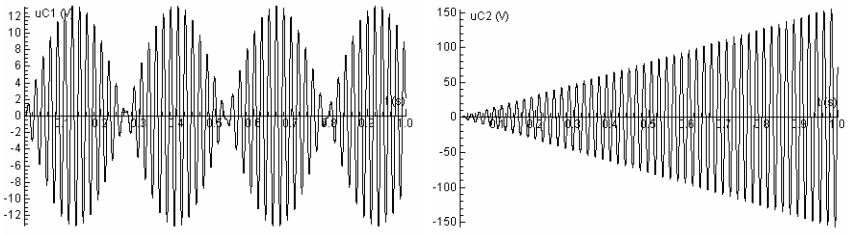
Model sleduje vznik nuceného kmitání v elektromagnetickém oscilátoru jako přechodný děj, při němž je oscilátor v počátečním okamžiku připojen ke zdroji harmonického napětí o stálé amplitudě. Pro nucené kmitání elektromagnetického oscilátoru platí 2. Kirchhoffův zákon ve tvaru

$$L\frac{\mathrm{d}i}{\mathrm{d}t} + Ri + \frac{q}{C} = U_{\mathrm{m}} \sin \omega t \,.$$

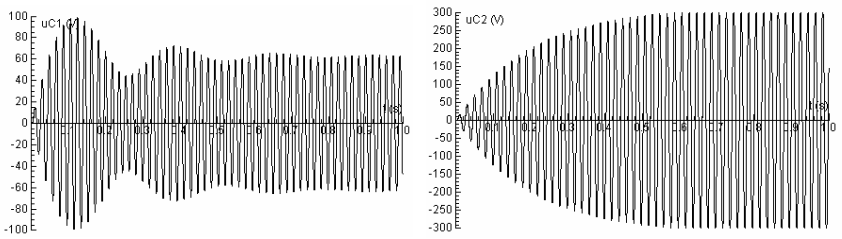
Na základě této rovnice sestavíme dynamický model (viz tabulku), v němž jsou současně modelovány případy, kdy $\omega_1 < \omega_0$ a $\omega_2 = \omega_0$ pro $R = 0$ (obr. 33) a $R = 10\ \Omega$ (obr. 34).

Model	Proměnné, konstanty a počáteční hodnoty

`t=t+h`
`u1=U*sin(omega1*t)`
`u2=U*sin(omega2*t)`
`q1=q1+i1*h`
`q2=q2+i2*h`
`uC1=q1/C`
`uC2=q2/C`
`uR1=R*i1`
`uR2=R*i2`
`uL1=u1-uR1-uC1`
`uL2=u2-uR2-uC2`
`i1=i1+((u1-uR1-uC1)/L)*h`
`i2=i2+((u2-uR2-uC2)/L)*h`
`if t>1 then stop endif`
	`C=1e-5`
	`L=1`
	`R=10`
	`omega1=340`
	`omega2=1/sqrt(L*C)`
	`U=10`
	`q1=C*u1`
	`q2=C*u2`
	`u1=0`
	`u2=0`
	`i1=0`
	`i2=0`
	`t=0`
	`h=1E-4`



Obr. 33



Obr. 34

5 Modely z fyziky mikrosvěta

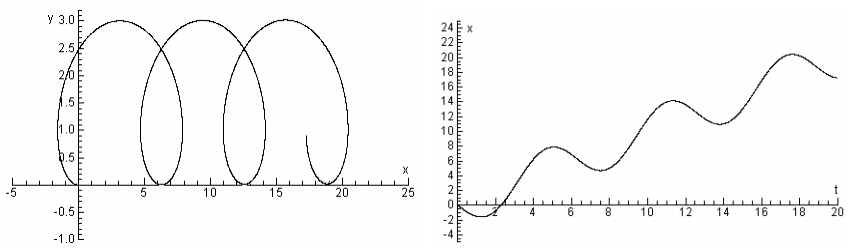
Dynamické modelování najde uplatnění i při výkladu dějů, při nichž na částici mikrosvěta působí elektrická, popř. magnetická síla. Příkladem je pohyb částice s nábojem působením Lorentzovy síly a pohyb částice α v blízkosti kladného jádra atomu (Rutherfordův pokus). Obdobou modelování pohybů je také modelování dějů, které jsou rovněž popsány diferenciálními rovnicemi, nejde však přímo o pohyby. Jako příklad takového děje uvedeme časový průběh přeměny radionuklidů, pro kterou platí zákon radioaktivní přeměny.

5.1 Působení Lorentzovy síly na částici s nábojem

Model zobrazuje pohyb částice s kladným náboje Q v elektrickém a magnetickém poli, přičemž vektor intenzity $\boldsymbol{E}$ elektrického pole má směr osy y a vektor magnetické indukce $\boldsymbol{B}$ magnetického pole má směr osy z. Na částici působí Lorentzova síla $\boldsymbol{F}_L = \boldsymbol{F}_e + \boldsymbol{F}_m = Q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$. Z modelu v tabulce je patrný výpočet souřadnic složek Lorentzovy síly. S ohledem na Flemingovo pravidlo pravé ruky je souřadnice složky F_y záporná. Na obr. 35 vlevo je trajektorie částice a na obr. 35 vpravo je časová závislost hodnoty souřadnice x pohybu částice. Vidíme, že částice při zobrazení v rovině Oxy opisuje cykloidu. V případě, že by na částici působila jen magnetická síla, pohybovala by se po kružnicové trajektorii.

Výpočet je značně citlivý na velikost časového kroku, což se projeví narůstajícím poloměrem trajektorie. Proto je výpočet rychlosti doplněn korekčním členem, který nepřesnost výpočtu v podstatě eliminuje a omezuje tak vliv volby časového kroku, jehož velikost limituje omezený počet cyklů výpočtu v programu Coach.

Model	Proměnné, konstanty, počáteční hodnoty
x=x+vx*h Fx=q*B*vy Fy=-q*B*vx+q*E ax=Fx/m ay=Fy/m vx=vx+ax*h-kor*vx*h^2 vy=vy+ay*h-kor*vy*h^2 x=x+vx*h y=y+vy*h t=t+h if t>=20 then stop endif	B=1 E=0.5 q=1 m=1 h=0.005 kor=0.5*(q*B/m) x=0 y=0 vx=-1 vy=0 t=0



Obr. 35

5.2 Rutherfordův pokus

Model historického Rutherfordova pokusu řeší pohyb částice α, čili nuklidu $^{4}_{2}\mathrm{He}$ ($Q_{\mathrm{He}} = 2e$, $m_{\mathrm{He}} = 4$ u) v centrálním elektrickém poli jádra atomu zlata $^{197}_{79}\mathrm{Au}$ ($Q_{\mathrm{Au}} = 79e$), které zaujímá malý objem. Na částici α působí elektrická síla

$$F_{\mathrm{e}} = \frac{1}{4\pi\varepsilon_0}\frac{Q_{\mathrm{He}}Q_{\mathrm{Au}}}{r^2}.$$

Podobně jako např. při pohybu družice v centrálním poli Země určíme složky této síly pro okamžitou vzdálenost r částice od jádra v rovině Oxy. Platí

$$r = \sqrt{x^2 + y^2},$$

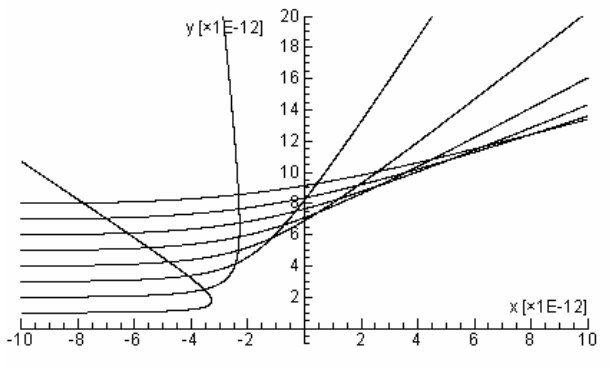
$$F_{ex} = F_e \frac{x}{r},$$

$$F_{ey} = F_e \frac{y}{r}.$$

Úlohu zjednodušíme předpokladem, že poloha jádra atomu zlata se nemění a trajektorie částic α budeme zobrazovat při menší rychlosti $v_0 = 10^6$ m · s^{-1} (skutečná rychlost částic α vyzařovaných izotopem radia je $1{,}9 \cdot 10^7$ m · s^{-1}).

Výpočet se v modelu opakuje pro 7 trajektorií ve vzájemných vzdálenostech $y_0 = 10^{-12}$ m. Model je v tabulce a na obr. 36 jsou zobrazeny charakteristické trajektorie částic α prokazující existenci kladného jádra atomu.

Model	Proměnné, konstanty, počáteční hodnoty
x=x+vx*h y=y+vy*h r2=x^2+y^2 r=sqrt(r2) Fx=kQQ/r2*x/r Fy=kQQ/r2*y/r ax=Fx/ma ay=Fy/ma vx=vx+ax*h vy=vy+ay*h t=t+h if y>ymax then I=I+1 vx=v0 vy=0 x=x0 y=y0+I*dy0 endif if I>Imax then stop endif	u=1.7e-27 eps0=8.85e-12 q=1.6e-19 ma=4*u Qa=2*q Qj=79*q kQQ=1/(4*pi*eps0)*Qa*Qj x0=-1e-11 y0=1e-12 v0=1e6 dy0=y0 Imax=7 ymax=2e-11 h=1e-20 I=0 x=x0 y=y0+I*dy0 vx=v0 vy=0 t=0



Obr. 36

5.3 Přeměna radionuklidu

Děj postupné přeměny jader radionuklidu modelujme pomocí rovnice

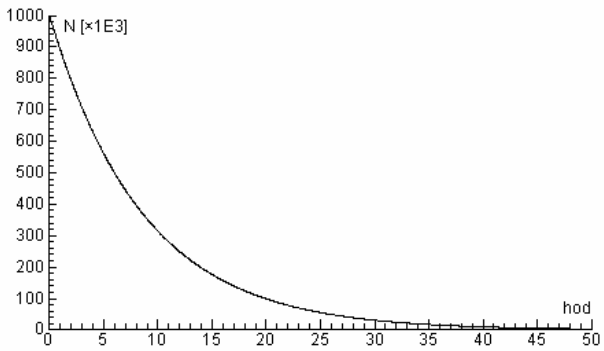
$$\frac{\Delta N}{N} = -\lambda \Delta t \ ,$$

ve které ΔN je úbytek jader radionuklidu za dobu Δt z počátečního počtu N jader. Přeměnová konstanta λ souvisí poločasem přeměny $T_{1/2}$ vztahem

$$T_{1/2} = \frac{\ln 2}{\lambda} = \frac{0{,}693}{\lambda} \ .$$

Tyto rovnice jsou východiskem pro sestavení modelu, který je v tabulce. Pro model je vhodné zvolit konkrétní radionuklid s krátkým poločasem přeměny, např. $^{99}_{43}\mathrm{Tc}$, což je zářič γ, který se používá v nukleární medicíně k tzv. scintigrafii při vyšetřování vnitřních orgánů. Jeho poločas přeměny $T_{1/2} = 6$ h = 21 600 s, takže přeměnová konstanta $\lambda = \ln 2/(2{,}16 \cdot 10^4 \text{ s}) = 3{,}21 \cdot 10^{-5} \text{ s}^{-1}$. Graf přeměny radionuklidu je na obr. 37, na němž se přesvědčíme, že za 6 hodin se přemění polovina z počátečního počtu jader.

Model	Proměnné, konstanty, počáteční hodnoty
dN=k*N*h N=N-dN t=t+h hod=t/3600 if hod>50 then stop endif	tp=2.16e4 k=ln(2)/tp N0=1E6 h=100 N=N0 t=0



Obr. 37

6 Feynmanovy úlohy

Jak již bylo zmíněno v úvodu, R. Feynman použil metodu numerického integrování v učebnici [1] jako prostředek k objasnění smyslu dynamických rovnic. V kapitole 9 věnované Newtonovým zákonům dynamiky jsou na s. 138 čtyři úlohy, které mají čtenáři sloužit k procvičení zmíněné metody, autor však neuvádí jejich řešení. Zadání úloh je vcelku jednoduché a představují základní typy úloh, předurčených pro řešení v podobě dynamického modelu, poněvadž jiný postup řešení by byl z matematického hlediska nepřiměřeně náročný. Uvedeme řešení těchto úloh jako námět pro práci i se středoškolskými studenty.

Úloha 1

Závaží zavěšené na pružině je v klidu. Po nárazu zdola se začne pohybovat a v prvním okamžiku je jeho rychlost jednotková. Hmotnost závaží a tuhost pružiny jsou takové, že pohybová rovnice má tvar $\ddot{x} = -x$.

Numerickým integrováním pohybové rovnice najděte maximální výšku, jíž závaží dosáhne.

Řešení

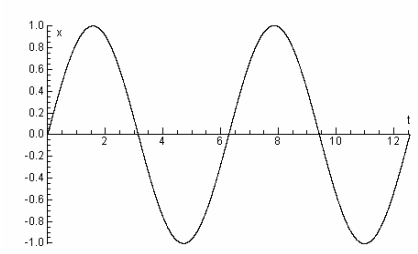
Ze zadání úlohy je zřejmé, že jde o pružinový oscilátor, který kmitá bez tlumení, takže jeho pohybová rovnice by měla tvar $ma = -kx$, kde m je hmotnost závaží a k je tuhost pružiny. Zadání tedy odpovídá případu, kdy poměr $k/m = 1$. Poněvadž je oscilátor uveden do pohybu nárazem zdola (předpokládá se, že osa x je svislá), má rychlost směr kladné osy x, čili $v_0 = 1\ \mathrm{m \cdot s^{-1}}$. Maximální výška závaží při kmitavém pohybu bude odpovídat amplitudě kmitů. Model, jak bychom ho vepsali do oken programu Coach, je v tabulce.

Model	Proměnné, konstanty, počáteční hodnoty

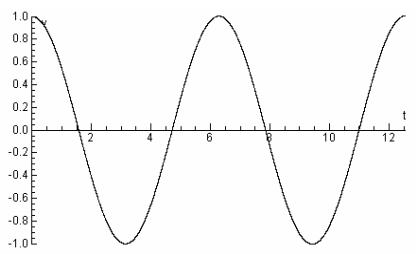
`a=-x`
`x=x+v*h`
`v=v+a*h`
`t=t+h`
`if t >= 4*pi then stop endif`
	`v0=1`
	`v=v0`
	`t=0`
	`x=0`
	`h=0.001`

Maximální výšku závaží odečteme z grafu souřadnice x na obr. 38a: $x_{max} = 1$ m. Na obr. 38b je graf rychlosti. V tomto jednoduchém případě ovšem můžeme amplitudu určit i ze vztahu $a = -\omega x = -x$, ze kterého vyplývá, že $\omega = 1\ \mathrm{s^{-1}}$,

a poněvadž $v_0 = v_m = \omega x_m = 1\ \mathrm{m} \cdot \mathrm{s}^{-1}$, je $x_m = 1$ m. Perioda kmitání $T = 2\pi/\omega \approx$ $\approx 6{,}28$ s, což je z grafu dobře patrné.



Obr. 38a



Obr. 38b

Úloha 2

Těleso o hmotnosti m se pohybuje přímočaře. Pohyb je brzděn silou úměrnou rychlosti tělesa $F = -kv$. V počátečním okamžiku $t = 0$ bylo $x = 0$ a $v = v_0$. Numerickým integrováním najděte x jako funkci času. Určete dobu $t_{1/2}$, za níž těleso ztratí polovinu své rychlosti, a maximální vzdálenost x_m, kterou projde.

Poznámky:

a) Zvolte měřítka času a vzdálenosti tak, aby pohybová rovnice měla jednoduché číselné koeficienty.

b) Navrhněte schéma výpočtu, analogické uvedenému v textu přednášek, jež by dávalo dobrou přesnost při poměrně hrubém kroku Δt.

c) Na základě úvah o rozměrnosti objasněte, jak $t_{1/2}$ a x_m závisejí na v_0, k a m. Pohybovou rovnici řešte jen pro jednu vhodnou hodnotu v_0, např. $v_0 = 1$ (v modifikovaných jednotkách x a t).

Řešení

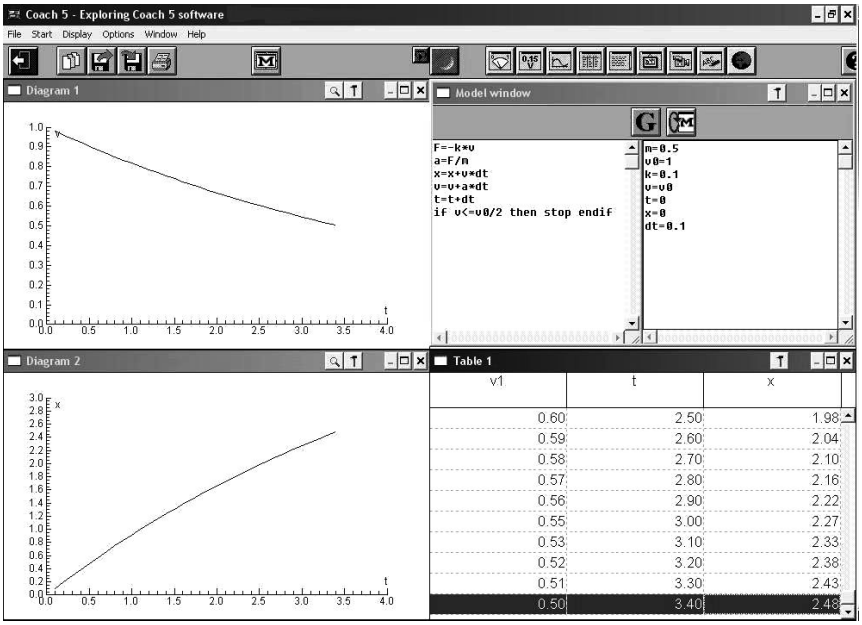
V počátečním okamžiku se těleso pohybuje přímočaře rychlostí v_0 a jediná síla, která na něj působí, je brzdící síla F. Pohybovou rovnici tedy zapíšeme ve tvaru

$$ma = -kv$$

a odtud určíme vztah pro zrychlení, který je výchozí rovnicí dynamického modelu pohybu tělesa. Jeho podoba je v tabulce. Zvolené hodnoty veličin m a k jsou v tabulce. Z grafů na obr. 39 je patrné, že rychlost tělesa se zmenší na polovinu ($v = 0{,}5\ \mathrm{m \cdot s^{-1}}$) za dobu $t = 3{,}4$ s a těleso urazí vzdálenost $x = 2{,}48$ m.

Model	Proměnné, konstanty, počáteční hodnoty

`F = -k*v`
`a = F/m`
`x = x+v*h`
`v = v+a*h`
`t = t+h`
`if v< = v0/2 then stop endif`
	`m = 0.5`
	`v0 = 1`
	`k = 0.1`
	`v = v0`
	`t = 0`
	`x = 0`
	`h = 0.1`



Obr. 39

Úloha 3

Nabitá částice se pohybuje v elektrickém a magnetickém poli podle pohybových rovnic:

$$\frac{\mathrm{d}v_x}{\mathrm{d}t} = -2v_y \quad \text{a} \quad \frac{\mathrm{d}v_y}{\mathrm{d}t} = 1 + 2v_x$$

Při $t = 0$ je částice v bodě o souřadnicích (0,0) a její rychlost má složky $v_x = 1{,}00$ a $v_y = 0$. Numerickým integrováním určete druh pohybu částice. Řiďte se poznámkou b) z předchozí úlohy.

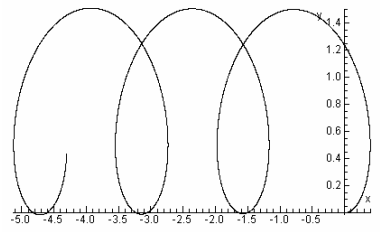
Řešení

Na částici s kladným nábojem Q v elektrickém poli o intenzitě $\boldsymbol{E}$ a v magnetickém poli o magnetické indukci $\boldsymbol{B}$ působí Lorentzova síla, pro kterou platí $\boldsymbol{F}_\mathrm{L} = \boldsymbol{F}_\mathrm{e} + \boldsymbol{F}_\mathrm{m} = Q(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})$. Z rovnic v zadání úlohy vyplývá, že při pohybu částice v kladném směru osy x na ni působí síla $\boldsymbol{F}_\mathrm{m}$ v kladném směru osy y, což podle Flemingova pravidla levé ruky znamená, že vektor $\boldsymbol{B}$ magnetické indukce míří za nákresnu. Současně na ni působí stálá elektrická síla $\boldsymbol{F}_\mathrm{e}$, která má směr kladné osy y. Složka v_y rychlosti částice je v počátečním okamžiku nulová, ale její zvětšení znamená vznik síly, jejíž složka v ose x má záporný směr. Tvar trajektorie částice najdeme pomocí modelu, který je v tabulce, a trajektorii znázorňuje graf na obr. 40a. Vidíme, že trajektorie částice znázorněná v rovině Oxy odpovídá pohybu po šroubovici.

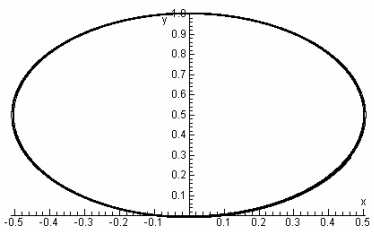
Můžeme si ještě vyzkoušet, jak se částice bude pohybovat v případě, že na ni nepůsobí elektrická síla (druhou rovnici modelu upravíme na tvar $a_y = 2v_x$). Částice se pohybuje po kružnici (obr. 40b, graf je zkreslen nestejným měřítkem na osách). Řešení úlohy je značně citlivé na velikost časového kroku. Již při použitém časovém kroku $h = 0{,}001$ je patrné postupné narůstání průměru šroubovice.

Model	Proměnné, konstanty, počáteční hodnoty

`ax = -2*vy`
`ay = 1 + 2*vx`
`vx = vx + ax*h`
`vy = vy + ay*h`
`x = x + vx*h`
`y = y + vy*h`
`t = t + h`
`if t>20 then stop endif`
	`t = 0`
	`h = 0.001`
	`vx = 1`
	`vy = 0`
	`x = 0`
	`y = 0`



Obr. 40a



Obr. 40b

Úloha 4

Mina vyletí z hlavně minometu rychlosti 300 m/s pod úhlem 45° k obzoru. Její pohyb je brzděn silou úměrnou třetí mocnině rychlosti ($F = -kv^3$). Koeficient úměrnosti k je takový, že při rychlosti 300 m/s je síla odporu prostředí dvakrát větší než tíha miny. Metodou numerického integrování najděte přibližnou hodnotu maximální výšky, do níž mina vyletí, a vzdálenost od místa výstřelu k místu, kde dopadne na zem. Výsledky srovnejte s hodnotami, které byste dostali, kdyby nepůsobil odpor vzduchu.

Řešení

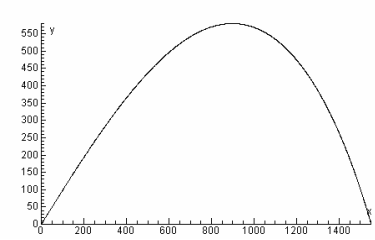
Pohyb miny je příkladem pohybu po balistické křivce a pro zobrazení trajektorie je třeba dynamickým modelem určit časové závislosti souřadnic x a y pohybu miny v rovině Oxy. Zadaný úhel je třeba převést na radiány a vypočítat počáteční hodnoty složek rychlosti v_x a v_y. V každém kroku výpočtu pak musíme nově vypočítat rychlost v miny ($v = \sqrt{v_x^2 + v_y^2}$) a z podmínky uvedené v zadání určíme velikost koeficientu k

$$k = \frac{2mg}{v_0^3} .$$

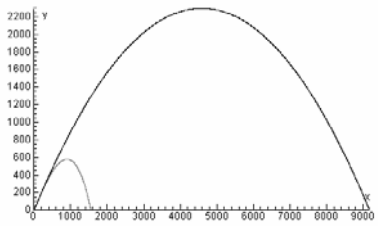
Pro numerické řešení úlohy zvolíme $m = 10$ kg.

Odpovídající model je v tabulce a graf balistické křivky je na obr. 41a. Na obr. 41b je tvar trajektorie v případě, že na minu nepůsobí odpor vzduchu. Z grafu, popř. přesněji z tabulky vypočítaných hodnot určíme maximální výšku $y_{max} = 579$ m a vzdálenost místa dopadu miny $x_{max} = 1\,551$ m. Pokud by na minu nepůsobila brzdící síla, bylo by $y_{max} = 2\,295$ m a $x_{max} = 9\,175$ m.

Model	Proměnné, konstanty, počáteční hodnoty
v=sqrt(vx^2+vy^2) ax=-k*vx*v^2 ay=-g-k*vy*v^2 x=x+vx*h y=y+vy*h vx=vx+ax*h vy=vy+ay*h t=t+h if y<0 then stop endif	m=10 g=9.81 v0=300 k=2*m*g/v0^3 h=0.01 uhel=45 alfa=pi/180*uhel vx=v0*cos(alfa) vy=v0*sin(alfa) x=0 y=0 t=0



Obr. 41a



Obr. 41b

Literatura

[1] *Feynman, R. P. – Leighton, R. B. – Sands, M.*: Feynmanovy prednášky z fyziky s řešenými příklady, Fragment, Praha 2000.

[2] *Maršák, J. – Pekárek, L. – Tomášek, V.*: Využití mikropočítače ve výuce fyziky na gymnáziu, I. díl, SPN, Praha 1988.

[3] *Dvořák, L. – Ledvinka, T. – Sobotka, M.:* Famulus 3.5, referenční příručka, Famulus Etc., Praha 1992.

[4] *Janeček, P.*: Interactive Physics – moderní nástroj ve výuce fyziky. MFI 14 (2005), č. 7, s. 433.

[5] http://www.cma.science.uva.nl/

[6] http://cs.wikipedia.org/wiki/ (Numerické řešení obyčejných diferenciálních rovnic)

[7] *Šedivý, P.*: Modelování pohybů numerickými metodami. Knihovnička fyzikální olympiády č. 38, Gaudeamus, Hradec Králové 1999. Dostupné na webu: http://fo.cuni.cz/texty/modelov.pdf

[8] *Lepil, O.*: Deterministický chaos. In. Nové poznatky ve fyzice, Repronis, Ostrava 2007, s. 32.

Příloha[1]

1. Jazyk Coach

1.1 Úvod

Součástí integrovaného prostředí Coach jsou programy Modelování a Řídicí prostředí, ve kterých je možno navrhovat, zapisovat, ladit a provádět modelové výpočty a řídicí programy. Instrukce programu musí být formulovány a pospojovány podle pravidel jazyka Coach, tj. způsobem, který je srozumitelný kompilátoru programů v prostředí Coach.

Před provedením modelového nebo řídicího programu se musí jeho text přeložit do strojového kódu (tzv. kompilace). Kompilátor také ověřuje, zda byly jednotlivé příkazy správně použity. Jestliže najde chybu, vypíše na obrazovku chybové hlášení.

V průběhu výpočtů se mohou vyskytnout chyby jiného druhu, např. hodnota argumentu funkce může překročit definiční interval této funkce nebo jmenovatel ve zlomku nabude nulové hodnoty. V takovém případě se programový výpočet zastaví a opět se vypíše chybové hlášení (viz §6, odst. Chybová hlášení).

1.2 Srovnání jazyka Coach s jinými programovacími jazyky

Jazyk Coach je relativně jednoduchý programovací jazyk odvozený od jazyka Pascal. Máte-li již nějaké zkušenosti s programováním v jazycích jako BASIC nebo Pascal, bude vám připadat většina výrazů a příkazů povědomá a srozumitelná. Abyste mohli začít psát programy v jazyce Coach, musíte se naučit pravidla pro zápis slova symbolů ve výrazech a příkazech (**syntaxe**) a samozřejmě také smysl těchto výrazů a příkazů (**sémantika**).

Na rozdíl od jiných univerzálnějších programovacích jazyků není v *jazyce Coach* možno definovat "pole" a "typy proměnných" nebo zadávat v průběhu výpočtu hodnoty z klávesnice a vkládat soubory z disku.

1.3 Použití jazyka Coach

Jazyk Coach umožňuje v programu Modelování zapsat příkazové řádky programu pro výpočet *modelu* spolu s tabulkou *počátečních hodnot;* takto je možno řešit i úlohy vedoucí k řešení diferenciálních rovnic. Programy se skládají z jednoho nebo více příkazů (programových řádek), které jsou sestaveny ze slov, čísel a znaků, resp. symbolů.

[1] (Zpracováno podle Přílohy č. 2 manuálu k systému IP Coach.)

2. Identifikátory, čísla a symboly

2.1 Identifikátory

Při zápisu vzorce nebo programu se pro označení konstant, proměnných, procedur a funkcí *používají jména (identifikátory),* která mohou být až na několik málo omezení libovolná. Je vhodné, když mají v některém živém jazyce smysl (např. proměnná *cas* nebo *time* je lepší než proměnná *XDR2W)*.

Při zápisu jmen **jsou povoleny** následující znaky z následujících sad:
– velká písmena {A, B, C, ...Z},
– malá písmena {a,b, c, ...z},
– číslice {1 2 3 … **O**},
– znaky {£ _ & ~ ! I { } [ ] };
– znaky ASCII kódy s výjimkou znaků π, ÷, Σ, √.

Při zápisu jmen se **nesmí** použít tyto znaky:
– znaky s ASCII kódy {0 ... 31},
– znaky označující aritmetické operace { ( ) - + * / + ^ = <>},
– vyhrazené znaky {' . , 7. " : ; $ # @} včetně mezery.

Poznámka č. 1

Není dovoleno používat:
– číslici jako první znak jména (např. 1strom),
– jména identicky shodná s klíčovými slovy, např. slovo 'sin'.

Poznámka č. 2

Omezení uvedená v poznámce č. 1 neplatí v tom případě, že jsou jména zapsána v hranatých závorkách. Je-li první znak jména "[" a poslední "]", je např. [2πr] povolené jméno proměnné.

Velká a malá písmena

Kompilátor nerozlišuje mezi malými a velkými písmeny. Například není možno současně použít znak R pro odpor žárovky a znak r pro výkon.

2.2 Čísla a číslice

Při zápisu čísel je povoleno použít znaky z následující sady:

{0 1 2 3 4 5 6 7 8 9 + – . e E}

Maximální počet platných míst je jedenáct: 1,0e–36 je nejmenší a 1,0e+36 je největší kladné číslo, která je možné zadat. Při překročení tohoto rozsahu je vypisováno chybové hlášení.

Při označení mocnin 10 se používá písmeno e nebo E (tzv. semilogaritmický zápis), např.: 0,5e-3 = 0,5E-3 = 0,0005 = 0.5*10^-3.

UPOZORNĚNÍ
Jazyk Coach používá jako oddělovače desetinných míst desetinnou tečku (.).
V textu této příručky je však používána jako oddělovač desetinných míst (ve shodě s národními zvyklostmi) desetinná čárka. Použití desetinné čárky jako oddělovače kompilátor *jazyka Coach* nepřipouští a vypisuje chybové hlášení.
Číslo nesmí začínat desetinnou tečkou a kompilátor vyžaduje zápis "0.1", nikoliv ".1".

2.3 Znaky a klíčová slova

Znaky

Některé *znaky* mají zvláštní význam a nemohou být použity ve jménech konstant, proměnných, funkcí a procedur:

{ + – * / ÷ ^ √ = > < . , ( ) : ; π ' }

Také některé *dvojice znaků* mají zvláštní význam:

{<= >= := <>}

Klíčová slova

Zvláštní význam mají i některá *slova,* která se nesmí používat pro označení konstant, proměnných, funkcí nebo procedur. Tato slova jsou jmenovitě vypsána v následující tabulce. Klíčová slova On (**Zap**), Off (**Vyp**) a Pi (π) označují konstanty, které mají hodnotu 255, 0, resp. 3,141592654 a které **nesmí být měněny.**

Abs	Fac	ResetAbsolute
And	Function	Round
ArcCos	If	RunningTime
ArcSin	Interval	SaveData
ArcTan	Level	Set
Becomes	Ln	SetAbsolute
Bit	Log	Sign
Column	Max	Sin
Cos	Min	Sound

Count	Not	Sqr
Counter	Off	Sqrt
Do	On	Stop
Else	Or	Stopwatch
EndDo	Pi	Tan
EndFunction	Print	Then
EndIf	Procedure	Until
EndProcedure	Rand	Wait
EndRedo	Redo	Latched
Entier	Repeat	Unlatched
Exp	Reset	While

Obr. 2.2: Přehled klíčových slov jazyka Coach

3. Výrazy

Výraz použitý v programu má číselnou nebo logickou hodnotu. Výrazy mohou být jednoduché nebo složené, např:

– samostatná čísla jako 6.13 nebo 105 a dále výsledky základních aritmetických operací s nimi, např. "6.13 +105", - konstanty (Pi, On, Off),
– proměnné, viz §3.1,
– volané funkce, viz §5.

3.1 Proměnné

Jméno (identifikátor) proměnné může být považováno za "úkryt" číselné hodnoty. Byla-li jménu proměnné přiřazena hodnota *aritmetického výrazu,* znamená to, že se tato hodnota uchovává v paměti počítače na místě označeném jménem proměnné, např. "Součet := 5 + 6" znamená, že v paměti počítače má proměnná "Součet" hodnotu "11". Pro identifikátory proměnných platí omezení podle §2.

Výsledkem *logického výrazu* je rovněž hodnota uchovávaná v paměti. Proměnná je pak označována jako *logická proměnná* (nebo booleovská proměnná).

Je-li výsledkem logického výrazu hodnota **On** (Zap, Ano, True, Pravda, logická jednička), zapisuje se do paměti číselná hodnota 255. Je-li výsledkem **Off** (Vyp, Ne, False, nepravda, logická nula), zapisuje se číselná hodnota 0.

Proměnné používané v programech mají obvykle *globální* význam, což znamená, že jsou dostupné v celém zbytku programu stejně jako v procedurách a funkcích, viz §5. *Lokální* proměnné se používají jen v definicích procedur a funkcí a ve zbytku programu je nelze přímo použít.

3.2 Aritmetické a logické operace

Výrazy se tvoří z proměnných (příp. konstant nebo funkcí) a operačních znamének (operátorů), které vyjadřují matematické operace mezi nimi. Z hodnot jedné či více proměnných obsažených ve výrazu se tak vypočítává jediná hodnota výrazu.

Operace je tedy předpis, podle kterého se z hodnot proměnných vypočítává jiná hodnota výrazu. Ve výrazech s více operacemi je pořadí výpočtu určeno tzv. prioritou jednotlivých operací, např.:

3*(V+1) - operace násobení "*" působí mezi hodnotami "3" a "(V+1)".

Jestliže operační znaménko působí na jedinou hodnotu, musí být tato hodnota zapsána za znaménkem. Jestliže operace probíhá mezi dvěma hodnotami, pak se znaménko zapisuje mezi nimi. Nejdříve se vykonávají operace s prioritou jedna, pak s prioritou dvě atd.

Aritmetické operace

Operace	Popis činnosti	Počet hodnot	Priorita
–	opačná hodnota	1	1
^	mocnina	2	1
*	násobení	2	2
/	dělení	2	2
+	sčítání	2	3
–	odčítání	2	3

Obr. 3.1: Aritmetické operace a pořadí jejich priority

V aritmetice mají operace 'sčítání' a 'odčítání' stejnou prioritu. Obdobně je tomu s operacemi 'násobení' a 'dělení'. Jsou-li ve výrazu použity operace stejné priority, výraz se vyhodnocuje zleva doprava.

Relační operátory

Při vzájemném porovnávání dvou hodnot se používají relační operátory. výsledkem porovnávání je logická hodnota Ano nebo Ne (v *jazyce Coach* odpovídají hodnotám On nebo Off, viz §3.1). Všechny relační operátory mají stejnou prioritu.

Pomocí logických operátoru mohou být vytvářeny složitější výrazy s relačními operátory pro porovnávání více hodnot (relační výrazy). V takových výrazech musí být vzájemně porovnávané dvojice hodnot vždy umístěny do kulatých závorek.

Operátor	Význam	Počet hodnot	Priorita
=	je rovno	2	4
<>	není rovno	2	4
<	menší než	2	4
>	větší než	2	4
<=	menší nebo rovno	2	4
>=	větší nebo rovno	2	4

Obr. 3.4: Relační operátory

3.3 Syntaxe výrazů

Použití mezer a klávesy <Enter>

V logických výrazech musí být identifikátory odděleny od operátoru *mezerami*. V aritmetických nebo relačních výrazech není použiti mezer jako oddělovačů povinné. výrazy se nemusí zapisovat na samostatnou řádku. Jestliže je výraz složitější a obsahuje více operátoru, bývá vhodné jej pro větší srozumitelnost rozepsat na více řádek.

Výrazy musí být vzájemně odděleny mezerou nebo znakem <Enter>, který se do textu programu vloží po stisknutí klávesy <Enter>. Jsou-li dva nebo více výrazU odděleny mezerou, mohou být zapsány na stejnou řádku. Při použití oddělovače <Enter> se každý výraz zapíše na samostatnou řádku.

***Použití závorek* ( )**

Kulaté závorky se použijí tehdy, je-li zapotřebí změnit prioritu operací. Mohou být použity také pro zpřehlednění zápisu výrazu.

4. Příkazy

Příkaz je takový prvek programu, který může být proveden samostatně. Nadále budeme rozlišovat mezi jednoduchými a složenými příkazy.

4.1 Jednoduché příkazy

Přiřazení

Příkazem *přiřazení* se původní hodnota proměnné nahradí výsledkem vypočteného výrazu. Symbolem přiřazovacího operátoru je ' := ' *('dvojtečka rovnítko')*.

V *jazyce Coach* je rovněž povoleno použití samostatného symbol ' = ' (*'rovnítko')* nebo slova ' becomes ' (stává se, nabývá hodnoty). Oddělování operátorů pomocí mezer není nezbytné:

```
proměnná := výraz
```

Příklady:
```
X:= Y + Z
Mokro becomes (déšť) and (not(deštník))
```

Volání procedury vzniklé spojením několika příkazů

Procedura se skládá z několika příkazů a je označena identifikátorem (jménem). Příkazy obsažené v proceduře se vykonávají na tom místě programu, na kterém je zapsáno její jméno (tzv. volání procedury). Definice procedury často obsahuje několik parametrů, viz §5.1. Používáním procedur se programy zpřehledňují a zlepšuje se jejich srozumitelnost.

4.2 Podmíněné příkazy typu 'If … Then ... ' (Jestliže ... Pak ...)

Podmíněné příkazy 'If ... Then .. .' (Jestliže ... Pak ...) mohou být dvojího druhu:

```
If Výraz                    Then Jestliže Výraz Pak
  Příkazy          tj.        Příkazy
EndIf                       KonecJestliže
```

Příkazy uvedené mezi klíčovými slovy Then a EndIf se provedou jen tehdy, má-li výraz následující za klíčovým slovem If logickou hodnotu On (Ano).

If Výraz Then
  Příkazy
Else
  Příkazy
Endlf
	tj	Jestliže Výraz Pak
		  Příkazy
		Nebo
		  Příkazy
		KonecJestliže

Výpočet výrazu zapsaného mezi klíčovými slovy If a Then musí dávat logickou hodnotu On (Ano) nebo Off (Ne). Jestliže má výraz logickou hodnotu On, pak se vykonají příkazy zapsané mezi klíčovými slovy Then a Else. Je-li naopak logická hodnota tohoto výrazu Off, vykonají se příkazy zapsané mezi klíčovými slovy Else a EndIf.

Několik příkladů výše uvedených podmíněných příkazů je uvedeno v následující tabulce na obr. 4.1.

IfX> 0 Then Y:= sqrt(X) Endlf	Je-li splněna podmínka (X > 0), přiřadí se proměnné Y hodnota funkce sqrt(X).

If (a < 2) and (b > 5) Then
  a := a + 1 b:= b - 5
Else
  a := a - 1 b:= b + 3
Endlf	

If a> b Then
  Maximum = a
Else
  Maximum = b
Endlf	

If a = 1 Then b := c Else
  If a = 2 Then b := -c Else
  b:= O
  Endlf
Endlf
	Hodnota přiřazená proměnné b závisí na hodnotě proměnné a.
	
	Pozor! Dva podmíněné příkazy jsou vnořeny do sebe a každý z nich musí být ukončen klíčovým slovem Endlf!

Obr. 4.1 Příklady použití podmíněného příkazu *'If ... Then ... '*

Podmíněný příkaz 'Repeat ... Until ... ' (Opakuj ... Dokud ...)

Příkazy zapsané mezi klíčovými slovy Repeat a Until jsou vykonávány tak dlouho, dokud není splněna podmínka uvedená za klíčovým slovem Until. Příkaz má syntaxi:

Repeat Příkazy Until Výraz tj. Opakuj Příkazy Dokud Výraz

Příkazy a výraz musí být odděleny od klíčových slov Repeat a Until mezerami. Příkazy se nejprve jedenkrát provedou a teprve pak se vyhodnocuje, zda je logická hodnota výrazu On (Ano) nebo Off (Ne). Jinak řečeno, příkazy se poprvé provedou bez ohledu na to, jaký je výsledek vyhodnocení výrazu.

Výsledkem výrazu musí být logická hodnota. Je-li výsledkem výrazu logická nula Off (Ne), provedení příkazů se opakuje. Je-li naopak výsledkem logická jednička On (Ano), pak se smyčka příkazů 'Repeat ... Until ...' ukončí a program pokračuje dále následujícím příkazem.

n:= 1 Repeat n:= n*2 Until n > 1000	Zdvojnásobování hodnoty n se opakuje tak dlouho, dokud hodnota n nepřesáhne 1 000 (tj. poslední a tím výsledná hodnota n bude 1024).
Repeat Záblesk Until Level(1) >= 4	Provádění procedury záblesk se opakuje tak dlouho, dokud hodnota level(1) (úroveň napětí v 1. kanálu karty AUR) je větší nebo rovna 4 (V).

Obr. 4.2 Příklady použití podmíněného příkazu *'Repeat ... Until ...' (Opakuj ... Dokud ...)*

Podmíněný příkaz 'ReDo ... EndRedo' (Opakuj ... KonecOpakování)

Tento podmíněný příkaz provádí pevně stanovený počet opakování skupiny příkazů a jeho syntaxe je následující:

Redo Výraz		Opakuj Výraz
Příkazy	tj.	Příkazy
EndRedo		KonecOpakování

Výraz a příkazy musí být odděleny mezerou jak navzájem, tak od klíčových slov ReDo a EndRedo. Výsledkem výrazu musí být číselná hodnota, která se chápe jako hodnota čítače a která se snižuje o jedničku po každém opakování skupiny příkazů (smyčky). Opakování příkazů pokračuje tak dlouho, dokud hodnota v čítači nedosáhne nuly, resp. hodnoty menší než nula. Několik příkladů použití je uvedeno na obr. 4.3.

n := 5 y:= n x:= 1 Redo n-1 x:= x*y y:= y-1 EndRedo	Výpočet faktoriálu n!
n:= 2.001 Redo n x:= sqr(x) EndRedo	Tento příkaz se třikrát opakuje.

Obr. 4.3 Příklady použití podmíněného příkazu *'ReDo ... EndRedo' (Opakuj ... KonecOpakování)*

Podmíněný příkaz 'While ... Do ... ' (Pokud ... Dělej ...)

Provádění skupiny příkazů se opakuje tak dlouho, dokud je logická hodnota podmiňujícího výrazu **On** (Ano). Příkaz má následující syntaxi:

While Výraz Do
Příkazy
EndDo
	tj.	Pokud Výraz Dělej
		Příkazy
		KonecDělej

Výraz a příkazy musí být odděleny mezerou jak navzájem, tak od klíčových slov While, Do a EndDo. Jestliže výraz nabude logické hodnoty Off (Ne), opakování příkazů se ukončí.

sum:= 0 While n > 0 Do sum:= sum + n n:= n-1 EndDo	Výpočet hodnoty součtu řady n + (n-1) + (n-2) .. + (n-i) pokračuje, pokud (n-1) je větší než 0, tj. dokud (n-i) <= 0

Obr. 4.4 Příklad použití podmíněného příkazu *'While...Do...' (Pokud…Dělej...*)

5. Procedury a funkce

Procedury a funkce zlepšují srozumitelnost programu. Obsahují skupinu několika příkazů, které se provádějí společně po vyvolání jména funkce, resp. procedury. Výsledkem provedení procedury je vykonání příkazů, které procedura obsahuje. Výsledkem provedení funkce je vypočítaná hodnota funkce.

V *jazyce Coach* je možno používat velkého počtu standardních (tzv. knihovních) funkcí a procedur, nebo zadefinovat v programech funkce a procedury nové.

5.1 Definování funkce

V *jazyce Coach* je syntaxe definované funkce následující:

```
Function Jméno_funkce(p1;p2;p3; ... )
    Příkazy
Jméno_funkce := výraz
EndFunction
```

Příkazové řádky funkce se zapisují mezi klíčová slova Function a EndFunction.

– Nejprve zapište *Jméno_funkce* včetně *parametrů* pl, p2, p3, atd., které jsou v příkazech a výrazech považovány za lokální proměnné. Identifikátor *Jméno_funkce* musí být oddělen od klíčového slova Function mezerou.
– Pak zapište potřebné *příkazy,* které musí být vzájemně odděleny mezerami nebo znakem <Enter>.
– Definice funkce musí být ukončena *přiřazovacím příkazem,* kterým se přiřadí identifikátoru *Jméno_funkce* hodnota výrazu vyjadřujícího hodnotu funkce. Znak přiřazení nemusí být oddělen mezerami.

Při definování funkce nezapomeňte, že:

– jméno funkce se **nesmí** shodovat s klíčovým (vyhrazeným) slovem, jménem proměnné, procedury nebo již dříve definované funkce,
– parametry definované v zadání funkce mají význam lokálních proměnných, identifikátory (jména) těchto parametrů musí být odlišné od ostatních a stejná jména nesmí být použita jinde v programu,
– parametry musí být vzájemně odděleny středníkem (;).

Je však dovoleno použít:

– globální proměnné v příkazech a měnit zde jejich hodnotu,
– funkci pro definování sama sebe (rekurzivní použití),
– při definování funkce jiné funkce, které již byly definovány.

5.2 Volání funkce

V programu se funkce používá tak, že se vhodné proměnné přiřadí hodnota funkce vypočtená podle funkčního předpisu definovaného výše popsaným způsobem. Příkaz k tomuto výpočtu a přiřazení (tzv. volání funkce) musí mít následující syntaxi:

```
x:= Jméno_funkce(a1;a2;a3; .. )
```

Počet parametrů použitých při volání funkce musí být shodný s počtem parametrů použitých v definici funkce (v hlavičce na řádce Function *Jméno_funkce(pl; p2; p3;...*). Jednotlivé parametry musí být odděleny středníkem (;).

Hodnoty parametrů zadané při volání funkce se po řadě jednoznačně přiřadí hodnotám lokálních proměnných použitých v definičním předpisu funkce. Při výpočtu funkční hodnoty pak program pracuje s těmito lokálními proměnnými a po ukončení výpočtu jsou všechny lokální proměnné z paměti vymazány.

Function Pythagoras(a;b) Pythagoras := √(a*a + b*b) EndFunction x:=3 y:=4 z := Pythagoras(x;y) k := Pythagoras(3*5;max(x;y))	Definice funkce Volání funkce Stejný význam má k := Pythagoras(15;4).

Function Factoria/(a)
  If a> 1 Then
  Factorial := a*Factorial(a-1)
  Else
  F actorial := 1
  Endlf
EndFunction
x:= Factorial(7)
	Rekurentní předpis pro výpočet faktoriálu, který se ukončí po dosažení hodnoty a=1.
	
	Výpočet hodnoty 7!

Obr. 5.1 Příklady správně definovaných a použitých funkcí

5.3 Definování procedur

Postup při definování procedury je velmi podobný výše popsanému definování funkce. Rozdíl spočívá v tom, že definice funkce musí končit přiřazovacím příkazem *Jméno_funkce := výraz,* jehož obdobu definice procedury nemusí obsahovat. Syntaxe definice procedury je následující:

```
Procedure Jméno_procedury(p1;p2;p3; .. )
  Příkazy
EndProcedure
```

Příkazy vykonávané po vyvolání procedury se zapisují mezi klíčová slova Procedure a Endprocedure.

– Nejprve zapište *Jméno_procedury* včetně *parametrů* p1, p2, p3, atd., které jsou v příkazech považovány za lokální proměnné. Identifikátor *Jméno_procedury* musí být oddělen od klíčového slova Procedure mezerou.
– Pak zapište potřebné *příkazy,* které musí být vzájemně odděleny mezerami nebo znakem <Enter>.

Při definování procedury nezapomeňte, že:
– jméno procedury se **nesmí** shodovat s klíčovým (vyhrazeným) slovem, jménem proměnné, funkce nebo již dříve definované procedury,
– parametry definované v zadání procedury mají význam lokálních proměnných, identifikátory (jména) těchto parametrů musí být odlišné od ostatních a stejná jména nesmí být použita jinde v programu,
– parametry musí být vzájemně odděleny středníkem (;).

Je však dovoleno použít:
– globální proměnné v příkazech a měnit zde jejich hodnotu,
– volání již definované funkce v definici procedury.

5.4 Volání procedur

Příkaz pro volání procedury musí mít následující syntaxi:

```
Jméno_procedury ( a1;a2;a3;...)
```

Počet parametrů použitých při volání procedury musí být roven počtu parametrů použitých v definici procedury (v hlavičce na řádce Procedure *Jméno_procedury(pl;p2;p3;...)*). Jednotlivé parametry musí být odděleny středníkem (;).

Hodnoty parametrů zadané při volání procedury se po řadě jednoznačně přiřadí hodnotám lokálních proměnných použitých v definici procedury. Při provádění procedury pak program pracuje s těmito lokálními proměnnými.

Procedure Bliknutí(t1;t2;n) Redo n SetAbsolute(1;3;5;7) Wait(t1) ResetAbsolute(1;3;5;7) Wait(t2) EndRedo EndProcedure Bliknutí(2;3;15)	Procedura Bliknutí n-krát provede následující smyčku: Na dobu t1 [s] nastaví bity č. 1, 3, 5 a 7 výstupního portu na úroveň log. 1 a ostatní bity na úroveň log. 0. Pak na dobu 12 [s] nastaví výše vyjmenované bity výstupního portu na úroveň log. 0 a ostatní bity na úroveň log. 1.

Obr. 5.2 Příklad správně definované procedury a jejího volání

5.5 Standardní funkce

Jazyk Coach umožňuje využívat přibližně 20 standardních matematických funkcí, které jsou uvedeny v následující tabulce. Syntaxe příkazu pro volání těchto funkcí je tato:

```
y := f(x1;x2;...)
```

Věnujte pozornost správnému použití závorek při zápisu složitějších funkcí, např.:

$\sin^2(x)$ musí být zapsáno ve tvaru sin(x)^2 nebo (sin(x))^2 ,

$\sin(x^2)$ musí být zapsáno ve tvaru sin(x^2), zápis sin^2(x) způsobí chybu a výpis chybového hlášenÍ.

Funkce je možno kombinovat a vytvářet tzv. složené funkce.

Identifikátor	**Význam**	**Parametry** *
sin (x)	sinus argumentu x	x se zadává v radiánech
cos(x)	kosinus argumentu x	x se zadává v radiánech
tan(x)	tangens argumentu x	x se zadává v radiánech $x \neq (1/2+n)*\pi$, $n \in Z$
arcsin(x)	arkus sinus (výsledek v rad)	$x \in (-1;1)$

arccos(x)	arkus kosinus (výsledek v rad)	x ∈ (-1;1)
arctan(x)	arkus tangens (výsledek v rad)	
sqr(x)	druhá mocnina x	x ∈ (-√(max); √(max»
sqrt(x)	druhá odmocnina x	x ∈ (0; max)
√(x)	druhá odmocnina x	x ∈ (0; max)
ln(x)	přirozený logaritmus (základ e)	x ∈ (0; max)
log (x)	dekadický logaritmus (základ 10)	x ∈ (0; max)
exp(x)	x-tá mocnina čísla e, tj. (e^x)	x ∈ (-max; 81)
entier(x)	zaokrouhluje číslo dolů: entier(2,8)=2; entier(-3,2)=-4	
round(x)	běžné zaokrouhlování: round(2,8) = 3; round(-3,2) = -3	
abs(x)	absolutní hodnota x	
rand	generátor náhodného čísla z intervalu (0; 1)	
max(x1;x2; ..)	výběr největšího z i parametrů x1, x2, ... ,xi	jeden nebo více parametrů
min(x1;x2; ..)	výběr nejmenšího z i parametrů x1, x2, ... ,xi	jeden nebo více parametrů
sign(x)	funkce signum má hodnotu: -1 pro x < 0, +1 pro x > 0 a 0 pro x = 0	
fac(x)	výpočet hodnoty (round(x))! (factorial)	x E (0;+33,5)

*max $\approx 10^{15}$

Obr. 5.3 Standardní funkce jazyka Coach

6. Chybová hlášení

V programu se mohou vyskytnout chyby dvou různých druhů:
– *syntaktické chyby*, které jsou zjišťovány při překladu (kompilaci) modelu nebo řídícího programu; syntaktická správnost se ověřuje před výpočtem programu nebo při změně škálování os, resp. proměnných vynášených na jednotlivé osy grafu,
– *chyby vznikající při běhu programu*, které jsou průběžně zjišťovány při programovaných výpočtech

6.1 Chybová hlášení vypisovaná při překladu

Je-li při překladu (kompilaci) zjištěná syntaktická chyba, zobrazí se text programu na displeji a na stavovou řádku se vypíše chybové hlášení. Kurzor bliká za pozicí, na které byla chyba zjištěna, a tato chyba může být okamžitě opravena.

```
t := t + dt
If t > 13 Then a := F/m
Else
a:= F/(2*m)
v:= v + a*dt
x:= x + v*dt
```

Obr. 6.1 Model se syntaktickou chybou

Kurzor nemusí vždy blikat právě na pozici skutečné chyby. V modelu uvedeném na obr. 6.1 chybí ukončení podmíněného příkazu klíčovým slovem 'EndIf'. Kompilátor však tuto chybu nezjistí, dokud neprojde všemi příkazy za klíčovým slovem Else.

V tomto případě se vypíše chybové hlášení Kompilátor očekává: "EndIf" a blikající kurzor se objeví až za posledním znakem posledního příkazu, nikoliv za přiřazovacím příkazem "a := F/(2*m)"

Chybové hlášení	***Příčina***
Znak není očekáván	Znak se nachází na pozici, na které by neměl být.
Toto není číslo	Kompilátoru se nepodařilo přečíst číslo.
Toto jméno již má jiný význam	Chyba způsobená použitím stejného identifikátoru (jména) pro více různých proměnných, funkci nebo procedur.

Různé typy proměnných	Chyba způsobena pokusem o přirazení nějaké hodnoty identifikátoru procedury, resp. výsledku funkce mimo její definici.
Kompilátor očekává: „...“	Kompilátor očekává jméno, znak nebo symbol.
Kompilátor nechce „...“	Daný znak nebo symbol kompilátor nepřijal (příčina této chyby není většinou okamžitě zřejmá).
Příliš mnoho proměnných	Paměť vyhrazená pro jména proměnných byla vyčerpána; použijte kratší identifikátory.
Funkční hodnota není přiřazena	Definice funkce neobsahuje přiřazení *Jméno_funkce := výraz* (pro výpočet hodnoty funkce)

Obr. 6.2 Chybová hlášení, která se mohou při kompilaci vyskytnout

6.2 Chybová hlášení při běhu programu

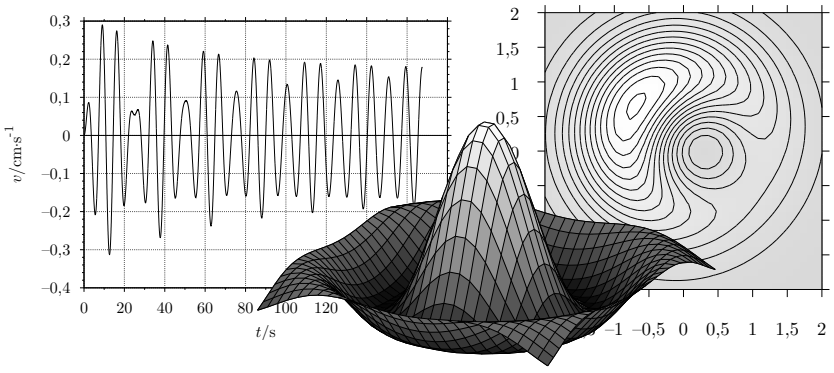
I když byl program úspěšně přeložen, není ještě zaručeno, že se po jeho spuštění neobjeví další chyby. Jestliže se tak stane, programový výpočet se ukončí a na obrazovku se vypíše chybové hlášení

Chybové hlášení	***Příčina***
Dělení nulou.	Proměnná ve jmenovateli zlomku nabyla nulové hodnoty.
Hodnota mimo povolený rozsah.	Číslo je příliš velké nebo příliš malé.
Hodnota nepatří do def. oboru funkce.	Hodnota argumentu standardní funkce není prvkem jejího definičního oboru (intervalu).

Obr. 6.3 Nejčastější chyby vznikající při programovém výpočtu a odpovídající chybová hlášení

Část II

DYNAMICKÉ MODELOVÁNÍ V PROGRAMU GNU OCTAVE



Obsah

Úvod

Řešení fyzikální úlohy má obvykle dvě samostatné části – obecnou a numerickou. V obecné části sestavíme na základě vhodných fyzikálních zákonů potřebné rovnice a z nich matematickými úpravami dojdeme ke vztahům, které vyjadřují hledané veličiny pomocí veličin daných a různých konstant. U jednodušších úloh pak v numerické části do těchto vztahů dosadíme číselné hodnoty daných veličin a konstant a získáme číselné hodnoty hledaných veličin, které doplníme příslušnými jednotkami, u složitějších někdy provádíme rozměrovou zkoušku.

V praxi se však často setkáváme se zdánlivě jednoduchými úlohami, jejichž řešení vede k transcendentním nebo diferenciálním rovnicím, jež buď neumíme řešit elementárními prostředky (analyticky), nebo je toto řešení – zejména na úrovni střední školy – velmi obtížné. Takové úlohy pak musíme řešit přibližnými numerickými metodami. Dnes, v době širokého rozšíření osobních počítačů a programovatelných kalkulátorů a bohaté nabídky využitelného software, se otevírají možnosti pro využití těchto metod prakticky všem zájemcům.

Problematika numerického řešení fyzikálních úloh na úrovni střední školy není zcela nová. Tento text proto navazuje na materiály [12, 19, 20] využívající český program *FAMULUS*, který si v minulých letech získal relativně velkou popularitu na školách. Jeho vazba na operační systém MS DOS a hlavně řada programů podobného zaměření vedly k tomu, že již není dále vyvíjen, i když zájemci si jej stále mohou stáhnout a nainstalovat např. z adresy `http://kdf.mff.cuni.cz/veletrh/sbornik/software/software.html`. Cílem toho textu je nabídnout jednu z možných volně dostupných alternativ tohoto programu, konkrétně *GNU Octave* (domovská stránka `http://www.octave.org/`).

Vzhledem k omezenému rozsahu zde nemůžeme nahradit podrobný uživatelský manuál k programu, namísto systematického výkladu se s ním budeme seznamovat prostřednictvím konkrétních příkladů. Zaměříme se na základní přehled zahrnující nejdůležitější vlastnosti programu, jeho výhody a případné nevýhody (na většinu z nich a možná i na další přijdou jistě uživatelé sami) a na konkrétních úlohách si ukážeme několik možností řešení transcendentních i jednoduchých diferenciálních rovnic. Více úloh spojených většinou s fyzikální olympiádou najdou zájemci v diplomové práci [9].

Část 7

Program GNU Octave

Jak napovídá sám název programu, je GNU Octave součástí *projektu GNU* (`http://www.gnu.cz/`), který patří k symbolům svobodného software. Zahrnuje celou škálu programů s různým využitím. Připomeňme nejprve základní principy projektu a s ním spojené licenční podmínky, které se vztahují i na software *GNU Octave*.

7.1 GNU a GPL

7.1.1 Projekt GNU

Projekt GNU byl založen v roce 1984 s cílem vytvořit klon operačního systému UNIX, který by byl šířen jako svobodný software (angl. free software). Odtud pochází i název – GNU je rekurzivní akronym pro „GNU's Not UNIX“ a vyslovuje se [gnů].

Obr. 7.1: Logo GNU projektu

V návaznosti na tento projekt byla v roce 1985 založena *Nadace pro svobodný software* (Free Software Foundation) s cílem podporovat práva uživatelů počítačů používat, studovat, kopírovat, modifikovat a redistribuovat počítačové programy. Nadace podporuje vývoj svobodného softwaru, jmenovitě pak operačního systému GNU. Dnes jsou hojně používány různé varianty operačního systému GNU s kernelem Linux; ačkoliv se těmto systémům často říká Linux, přesnější pojmenování pro ně je GNU/Linux.

Obr. 7.2: Richard Matthew Stallman (převzato z [1])

Základem filosofie projektu GNU je přesvědčení, že svobodný software je záležitost svobody: lidé by měli mít svobodu využívat software všemi způsoby, které přinášejí nějaký společenský užitek. Software se liší od hmotných objektů (židle, sendviče nebo benzínu) v tom, že může být mnohem snadněji kopírován a modifikován a uživatelům se dává možnost – u programů patřících do GNU projektu – tuto výhodu maximálně využít. Zakladatelem a ústřední postavou hnutí je *Richard Matthew Stallman* (obr. 7.2, osobní stránka `http://http://www.stallman.org/`); původně absolvent ba-

kalářského studia fyziky na Harvardově univerzitě je znám především jako autor editoru *GNU Emacs* a překladače gcc.

FSF/UNESCO adresář svobodného softwaru (FSF/UNESCO Free Software Directory, `http://directory.fsf.org/`) dnes zahrnuje více než 4000 balíčků svobodného softwaru poskytuje. Pro podporu šíření svobodného softwaru, GNU projekty i non-GNU projekty, poskytuje FSF FTP servery (`ftp://ftp.gnu.org/`) které jsou zrcadleny po celém světě. Pro vývojáře mnoha GNU i non-GNU projektů poskytuje Free Software Foundation CVS server Savannah (`http://savannah.gnu.org/`).

7.1.2 GPL – General Public Licence

Na všechny softwarové balíčky zařazené do projektu GNU (včetně programu *GNU Octave*) se vztahuje *všeobecná veřejná licence GNU* (v originále GNU General Public Licence) označovaná nejčastěji zkratkou *GNU GPL*[6] popřípadě některá z jejích slabších verzí (např. GNU LGPL). Podle svých principů a v kontrastu s běžně známým copyrightem bývá tato licence označována jako *copyleft*. Jejím cílem je zaručit:

- svobodu ke sdílení a úpravám svobodného softwaru;
- právo spouštět program za jakýmkoliv účelem;
- právo studovat, jak program pracuje, a přizpůsobit ho svým potřebám (předpokladem k tomu je přístup ke zdrojovému kódu, nejen k k programu samotnému);
- právo redistribuovat kopie dle svobodné vůle;
- právo vylepšovat program a zveřejňovat zlepšení, aby z nich mohla mít prospěch celá komunita (předpokladem je opět přístup ke zdrojovému kódu).

V tomto smyslu je svobodný (free) software takový, k němuž je k dispozici také zdrojový kód, spolu s právem tento software používat, modifikovat a distribuovat. Naprostá většina svobodného software je zdarma, ačkoliv to podle licence není podmínkou. Svobodný software neznamená nekomerční, komerční vývoj svobodného software není ničím neobvyklým. Za získání kopií svobodného software můžete platit, nebo je obdržet zdarma, ovšem bez ohledu na způsob, jak jste je získali, máte vždy svobodu kopírovat a měnit software, dokonce prodávat nebo darovat jeho kopie nebo pozměněné verze. Copyleft licence tak říká, že pokud redistribuujete originální nebo pozměněnou verzi programu, musíte tuto verzi redistribuovat pod stejnou licencí pod jakou jste získali původní program, nesmíte přidat žádná omezení, abyste tak neodepřeli zmíněné základní svobody ostatním.

Kromě toho bývá zvykem, že jednotlivé programové balíčky mají své vlastní internetové stránky, odkud je nejen možné si program stáhnout, ale uživatelé tam najdou i manuály (standardní bývá angličtina, překlady do dalších jazyků závisí většinou na množství uživatelů a dobrovolných překladatelů v dané jazykové

oblasti) a diskusní mailové skupiny, kam je možné posílat dotazy ohledně používání programu a poučit se na konkrétních problémech u zkušenějších uživatelů či přímo autorů programu. Pro zdatné programátory se také otevírá možnost přímo se na vývoji či vylepšování programu podílet (což dělá i běžný uživatel, pokud hlásí autorům chyby, s nimiž se při své práci setkal – každý, i sebedokonalejší program je koneckonců jen lidským výtvorem).

S pojem svobodného software volně souvisejí (a někdy jsou ne zcela správně zaměňovány) také *freeware* a *open source*. Termín „freeware“ nemá jasnou a přijatelnou definici, ale je běžně používán pro balíky programů, u kterých je dovolena distribuce a používání, ale ne modifikace (zdrojové kódy nejsou dostupné), mezi svobodný software proto nepatří. Termín „open source“ je často používán k označení víceméně týchž věcí jako svobodný software, i když jeho licence nemusí zaručovat tolik svobod jako GPL. Podrobnější popis jednotlivých kategorií najdeme např. na stránkách `http://www.gnu.org/philosophy/categories.cs.html`.

Jak již bylo zmíněno, do kategorie svobodného software dnes patří celá řada programu, které právě díky své volné dostupnosti mohou být právě ve školství vhodnou alternativou komerčně prodávaných proprietárních programů. Zmiňme např. program pro vykreslování a znázorňování dat *GNUplot* (domovská stránka `http://www.gnuplot.info/`, ukázky grafů `http://gnuplot.sourceforge.net/demo/`), který používá ke grafickému znázornění i *GNU Octave*, balík pro symbolické výpočty (tzv. počítačová algebra) *GNUMaxima*[1] (domovská stránka `http://maxima.sourceforge.net/`) nebo program pro grafické úpravy obrázků a fotografií *Gimp* (GNU Image Manipulation Program; domovská stránka `http://www.gimp.org/`, české stránky `http://www.gimp.cz`).

7.2 Program GNU Octave

GNU Octave představuje vyšší programovací jazyk („jazyk pro neprogramátory“) zaměřený na numerické operace podobný poměrně rozšířenému *Matlabu*® firmy The Mathworks (`http://www.mathworks.com/`). Jeho autor *John W. Eaton* – počítačový administrátor skupiny chemického inženýrství na University of Wisconsin, která se mimo jiné zabývá i návrhy chemických reaktorů

[1] *Maxima* vychází z projektu *Macsyma*, jenž byl vyvíjen v MIT (Massachusetts Institute of Technology) a financován United States Department of Energy a dalšími vládními organizacemi. O vývoj jedné z verzí Macsyma se staral od roku 1982 až do své smrti v roce 2001 Bill Schelter, jenž v roce 1998 získal svolení uveřejnit svou verzi pod GPL. Tuto verzi nazývanou *Maxima* nyní udržuje nezávislá komunita vývojářů a uživatelů.

– tento jazyk vytvořil roce 1988 jako pomůcku pro studenty, aby nemuseli zvládnout běžné programovací jazyky Fortran nebo C++. Samotné jméno „Octave“ prý pochází ze jména autorova bývalého učitele, který proslul svou schopností dělat rychlé kalkulace na kouscích papírků. Program je podobně jako *Matlab*® založen na efektivním počítání s maticemi, pracovat můžeme buďto interaktivně (postupné zadávání příkazů) nebo spouštěním delších předem připravených skriptů. Historie zadaných příkazů se zaznamenává a je možné se k nim postupně vracet nebo si historii uložit jako hotový skript. Zadávat lze i komplexní čísla, k dispozici jsou dále goniometrické, hyperbolické funkce, logaritmy, logické funkce, cykly, statistické funkce, polynomy, algoritmy pro numerické řešení diferenciálních rovnic, numerickou integraci, konverzi (a rozklad) obrazových i audio dat, zpracování signálů atd. Jak již bylo řečeno, pro zobrazování grafických výsledků se automaticky volá program *GNUplot*, z něhož lze obrázek uložit v různých formátech (postscript, png apod.).

7.3 Proč (ne)používat GNU Octave

Volba programu, který bude používat, patří vždy k výsostným záležitostem uživatele. V oblasti matematicky orientovaných programů je dnes k dispozici velké množství balíků, od komerčních proprietárních (např. *Matlab*®, *Maple*® nebo *Mathematica*®) až pro freeware nebo svobodný software (např. *Scilab* nebo *GNU Maxima*). Nabízí se proto otázka, jaké výhody či nevýhody jsou spojeny s používáním programu *GNU Octave* a proč si (ne)vybrat právě tento balík. Kromě ceny (je zdarma, takže oproti výše zmíněným proprietárním programům vychází jedna licence asi o 30 000,- Kč levněji) bychom měli připomenout a zvážit následující klady („+“) a zápory („−“):

− K efektivnímu používání musíme zvládnout alespoň základy programovacího jazyka, což ale platí pro většinu matematického software.
− Základem výpočtů jsou manipulace s maticemi, což v některých situacích vyžaduje zvláštní syntax (více v následující kapitole).
− Program nemá propracovaná grafické prostředí a proto „není moc o klikání“; většinu úkonů ovládáme pomocí klávesnice z příkazové řádky.
± *GNU Octave* není výukový program, proto se nezaměřuje ani na grafické ovládání či pohyblivé simulace. Tím se na druhé straně více blíží programům často užívaným v reálné technické praxi.
± Využití programu se neomezuje zdaleka jen na dynamické modelování, jemuž je převážně věnován tento text. Díky tomu se může zdát pro jeden daný účel použití příliš robustní a složitý, na druhou stranu – pokud si na program už jednou zvykneme – můžeme v jednou prostředí řešit hodně problémů.

± Jde o mezinárodní projekt, většina materiálů a manuálů i diskusní skupina používá angličtinu, jejíž znalost je bezesporu výhodou. Na druhou stranu je velmi pravděpodobné, že v široké komunitě někdo řešil a už vyřešil podobný problém jako my a bude nám umět poradit. Větší počet uživatelů také rychleji odhalí případné chyby a nedostatky programu.

\+ Programovací jazyk používaný *GNU Octave* má syntaxi velmi podobnou (i když ne 100% kompatibilní) v praxi široce používanému *Matlabu*®. Pokud si náš student zvykne na základní principy a filozofii, případný pozdější přechod mezi programy není obtížný.

\+ *GNU Octave* je multiplatformním programem, lze ho používat jak v operačním systému Windows, tak ve většině verzí Linuxu (v řadě distribucí jsou i předkompilovány instalační balíčky). Přiznejme, že používání pod Linuxem je v současné době pohodlnější.

\+ Základní manuály – na rozdíl např. od Scilabu (pokud je nám známo) – jsou k dispozici i v češtině ([11, 15]).

\+ S použitím *GNU Octave* se setkáme na řadě míst ve Wikipedii [1] a pokud s programem umíme pracovat, můžeme si přímo spustit skripty, které tam nalezneme.

V současné době roste popularita java-appletů a také simulačních programů typu *Interactive Physics*™ (domovská stránka v angličtině má adresu `http://www.design-simulation.com/IP/index.php`, informace v češtině najdeme např. na `http://www.ictphysics.upol.cz/` nebo `http://www.gjwprostejov.cz/projekty/sipvz03/`). Zatímco pro tvorbu vlastních appletů musíme zvládnout objektově orientovaný programovací jazyk *Java* vyvinutý firmou Sun Microsystems se vším, co k tomu patří, výhodou „interaktivní fyziky“ je bezesporu propracované grafické prostředí umožňující poměrně snadno vytvářet velmi působivé simulace reálných fyzikálních dějů. Jak již bylo řečeno v první části textu, řada předdefinovaných faktorů přitom jednak částečně omezuje oblast použití a jednak část fyzikálních vlastností zůstává před zvídavým žákem skryta. Proto jsme přesvědčeni, že i dnes má smysl se dynamickým modelováním zabývat, neboť při něm máme pod kontrolou všechny parametry a procedury. A hlavně – radost z dobře napsaného a fungujícího skriptu určitě stojí za to!

Část 8

První kroky s programem

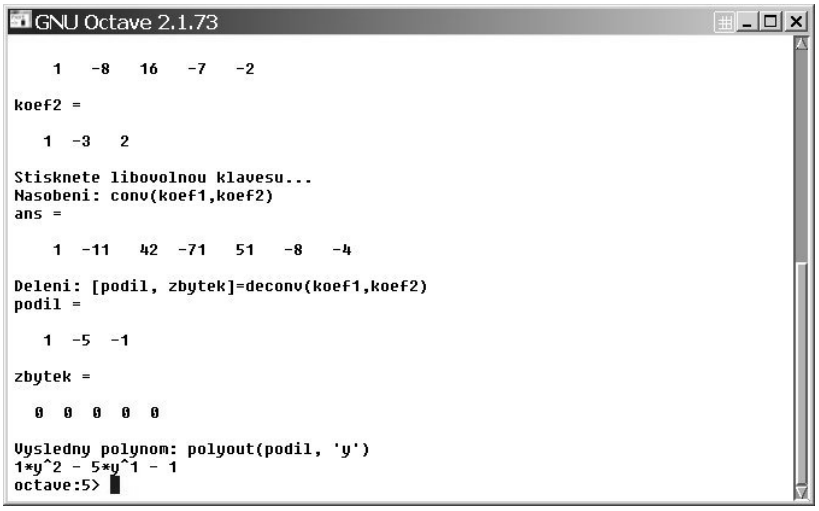
8.1 Příkazový řádek a terminál

Již jsme zmínili, že základním pracovním prostředím programu je terminál s příkazovým řádkem (obr. 8.1), pomocí něhož zadáváme programu jednotlivé příkazy. K jednou zadaným příkazům se můžeme vracet pomocí klávesy „se šipkou nahoru“, takže je není nutné vypisovat znovu. Terminál je také schopen doplňovat jména příkazů, definovaných funkcí a jmen souboru v aktuálním adresáři pomocí klávesy Tab (tabulátor); pokud např. napíšeme

```
octave:1> ta
```

a stiskneme tabulátor, program nabídne možná doplnění

```
table       tabulate     tan          tanh        taylorcoef
```



Obr. 8.1: Terminálové okno s příkazovým řádkem ve Windows XP

I když je možné historii příkazů ukládat, delší posloupnosti příkazů se bezesporu vyplatí uložit do souboru a volat všechny příkazy v tomto souboru (tj.

celý uložený skript) jedním příkazem, odpovídajícím jménu uloženého souboru. Podobně jako u *Matlabu*® soubory ukládáme s příponou `*.m`, seznam souboru v aktuálním adresáři zjistíme příkazem[1]

```
octave:3> ls
```

jehož odpověď může vypadat např. následovně

```
baliste.m     grafika.m   liss.m      micek.m      pruzkyv.m
balist.m      grafika2.m  lyzarfv.m   oscilb.m     raketa.m
foucault.m    haleboop.m  matice.m    parasut.m    skladkmitf.m
```

Soubory s příponou `*.m` můžeme přímo spustit, z výše uvedeného výpisu třeba

```
octave:4> foucault
```

spustí model kmitů Foucaultova kyvadla popsaný v části 10.3.2. Aktuální adresář, v němž momentálně *GNU Octave* pracuje, získáme příkazem

```
octave:5> pwd
```

a do jiného adresáře (složky) se přepneme příkazem[2]

```
octave:6> cd
```

K psaní či následné editaci vlastních skriptů lze využít většinu textových editorů, např. *Poznámkový blok* (Notepad), který je součástí instalace operačního systému Windows. Píšeme-li však skriptů více, patrně oceníme možnosti pokročilejších editorů, které umí číslovat řádky, barevně odlišovat komentáře, příkazy, funkce i číselné konstanty (tzv. syntax highlighting neboli zvýrazňování syntaxe). většinu těchto editorů lze využít i pro jiné programovací jazyky, psaní dokumentů v TEXu nebo editaci kódu www-stránek. I v této kategorii je nabídka freewarových programů poměrně široká – přímo s instalací *GNU Octave* pro Windows se nainstaluje editor *Scite* (`http://www.scintilla.org/SciTE.html`), z dalších stojí za zmínku např. původem český *PSPad* autora Jana Fialy (`http://www.pspad.com/cz/`), který lze do systému i integrovat jako náhradu až příliš jednoduchého Poznámkového bloku. U obou zmíněných programů lze nastavit řadu parametrů, včetně fontů a barev tak, jak to uživateli subjektivně nejlépe vyhovuje.[3] Nastavíme-li, aby se daný typ souboru otevíral vždy ve vybraném editoru, můžeme se pustit do psaní prvních programů.

[1] Modifikací `ls -l` získáme podrobný výpis souborů s jejich velikostí a datem poslední změny.

[2] I k němu existuje řada možných parametrů např. `cd /` přepíná do hlavního adresáře (složky), jimž v OS Windows většinou bývá disk `C:`. Obsahují-li jména adresářů mezery, musíme dát název do uvozovek, tj. `cd "/Documents and Settings"`.

[3] V současné verzi PSPadu používáme stejný zvýrazňovač pro *GNU Octave* i *Matlab*® označený jako MATLAB13.

8.2 Základní funkce a příkazy

Nyní jsme připraveni vyzkoušet první kroky s programem. Podobně jako *Matlab*® vypisuje *GNU Octave* čísla v různých formátech. V základním nastavení vypisuje dvě notoricky známé základní konstanty ve tvaru

```
octave:1> pi, e
pi = 3.1416
e = 2.7183
```

Chceme-li si výstup na více desetinných míst, použijeme

```
octave:2> format long
octave:3> pi, e
pi = 3.14159265358979
e = 2.71828182845905
```

k původnímu nastavení se vrátíme pomocí `format short`. Zdůrazněme, že tím *neovlivňujeme* přesnost, s jakou *GNU Octave* používá čísla při výpočtech. Maximální počet desetinných míst, která můžeme zobrazit je 42[4]; dosáhneme toho příkazem

```
octave:4> printf ("%.42f\n", pi);
```

kde `\n` značí přechod na nový řádek po vykonání příkazu. Podobně si můžeme nechat vypsat např. $\sqrt{10}$

```
octave:5> printf ("%.42f\n", sqrt(10));
3.162277660168379522787063251598738133907318
```

Základními objekty, s nimiž program efektivně pracuje, jsou matice, i číselné konstanty chápe jako matice s jedním prvkem. Jednořádkovou matici (řádkový vektor) zadáme ve tvaru

```
octave:6> m=[1 2 3 4 5 6]
m =

  1  2  3  4  5  6
```

matici k ní transponovanou (tj. sloupcový vektor)

[4]Chceme-li např. Ludolfovo číslo π na 1 000 desetinných míst, musíme použít program pro symbolické výpočty (např. *GNU Maxima*) nebo alespoň nainstalovat k programu *GNU Octave* knihovnu pro symbolické výpočty, s níž bude v takových případech spolupracovat (např. *GiNaC*).

```
octave:7> m'
ans =

  1
  2
  3
  4
  5
  6
```

což lze dosáhnout i přímo zadáním (řádky matice ukončujeme středníkem)

```
octave:8>  m=[1; 2; 3; 4; 5; 6]
m =

  1
  2
  3
  4
  5
  6
```

Ukončíme-li řádek středníkem, matice se vytvoří, ale nevypíše na obrazovce, což je užitečné u velkých matic s mnoha prvky

```
octave:9>  m=[1; 2; 3; 4; 5; 6];
```

Vyjdeme-li z původní řádkové matice `m=[1 2 3 4 5 6]`, můžeme násobení řádkové a sloupcové matice zapsat

```
octave:10> m*m'
ans = 91
```

z matematického hlediska jde o součet $\sum_i m_i^2$. Protože u násobení matic závisí na pořadí (není komutativní), záměnou matice m a matice transponované získáme matici

```
octave:11> A=m'*m
A =

   1   2   3   4   5   6
   2   4   6   8  10  12
   3   6   9  12  15  18
   4   8  12  16  20  24
```

```
   5  10  15  20  25  30
   6  12  18  24  30  36
```

neboli matici s prvky $A_{ij} = m_i m_j$. *Při násobení a umocňování matic proto musíme dát pozor!* Chceme-li umocnit prvky matice např. na druhou, *nemůžeme napsat* pouze `m^2` – chybová hláška nám napovídá, že *GNU Octave* se snaží umocnit matici jako celek a to lze jen v případě čtvercové matice

```
octave:12> m^2
error: for A^b, A must be square
error: evaluating binary operator '^' near line 11, column 2
```

Chceme-li získat matici s jednotlivými prvky m_i^2, musíme napsat

```
octave:12> m.^2
ans =

   1   4   9  16  25  36
```

nebo pomocí násobení `m.*m`. *Toto opomenutí bývá příčinou mnoha chybových hlášek*, u čtvercových matic, kdy se chybová hláška neobjeví, může vést ke špatnému výsledku. Dodejme, že na stejný zápis a „nepohodlí“ si musí zvyknout i uživatelé *Matlabu*®.

Kromě tištěných manuálů či návodů na internetu můžeme základní návody a informace získat přímo pomocí programu příkazem

```
octave:13> help
```

hledáme-li pomoc ke konkrétnímu příkazu, připojíme jeho jméno

```
octave:14> help plot
```

V terminálu se nám objeví popis, jímž můžeme postupně listovat, opustíme ho klávesou „q“, po jejímž stisknutí můžeme zadávat další příkazy. V případě potřeby můžeme výpis terminálového okna vymazat příkazem

```
octave:15> clc
```

nadefinované konstanty a proměnné vymažeme příkazem

```
octave:16> clear
```

pouze matici `m` vymažeme příkazem `clear m`. Hlášky vypisované na obrazovku a komunikující s uživatelem zadáváme pomocí příkazu `printf` nebo `fprintf`, např.

```
octave:17> fprintf ("Stisknete libovolnou klavesu...\n");
```

Význam to má zejména, pokud připravujeme programy, jež bude spouštět i někdo jiný (třeba naši studenti). Chceme-li v duchu předchozí ukázky do skriptu zařadit přerušení běhu programu s čekáním na stisk libovolné klávesy uživatelem, použijeme `pause`.

8.3 Některé matematické funkce

Z hlediska středoškolské matematiky a fyziky má *GNU Octave* předdefinováno několik zajímavých funkcí a procedur. Např. řešení soustavy lineárních rovnic

$$\begin{aligned} 4x + 3y - 2z &= 40 \\ 6x - 5y - 3z &= 50 \\ 3x + 2y + 5z &= 220 \end{aligned}$$

snadno najdeme pomocí matice soustavy

```
octave:1> A=[4 3 -2;6 -5 3;3 2 5]
A =

   4   3  -2
   6  -5   3
   3   2   5
```

a matice sestavené z pravých stran

```
octave:2> B=[40;50;220]
B =

   40
   50
  220
```

podílem matic

```
octave:3> A\B
ans =

  10.000
  20.000
  30.000
```

určujícím řešení $x = 10$, $y = 20$ a $z = 30$. Na tomto lze demonstrovat dvojí typ dělení matic. Výše použitá operace `A\B`[5] matematicky odpovídá násobení inverzní maticí zleva $A^{-1}B$. Běžně užívaný znak dělení `A/B` pak reprezentuje výraz AB^{-1}, který ovšem pro naše konkrétní matice nelze vypočítat pro nekompatibilní počet řádků a sloupců matic A a B. Rozdíl mezi oběma děleními pak ukážeme na pomocí matice B, srovnáme-li výrazy[6]

```
octave:4> B\B, B/B
ans = 1.0000
ans =

  0.030476  0.038095  0.167619
  0.038095  0.047619  0.209524
  0.167619  0.209524  0.921905
```

Podobně jako při násobení musíme dávat pozor, chceme-li dělit jednotlivé prvky matice, kde je opět třeba vložit tečku

```
octave:5> B./B
ans =

  1
  1
  1
```

Na matici A si můžeme ukázat výběr jejích jednotlivých prvků. Např. prvek A_{23} vyvoláme příkazem,

```
octave:6> A(2,3)
ans = 3
```

3. řádek a 2. sloupec pak postupně

```
octave:7> A(3,:), A(:,2)
ans =

  3  2  5
```

[5] Znak \ používáme také v jiném významu – rozdělujeme jím příkaz na více řádků, pokud je z nějakých důvodů jejich délka omezena jako např. při sazbě tohoto textu. Potom za ním ale následuje přechod na nový řádek.

[6] Dodejme, že pojem inverzní matice zavádíme přesně vzato pouze u matic čtvercových, z ukázky vidíme, že dvojí typ dělení umožňuje *GNU Octave* i pro některé matice jiného typu.

```
ans =

   3
  -5
   2
```

Součet řádkové nebo sloupcové matice (vektoru) získáme jednoduše příkazem

```
octave:8> sum(B)
ans = 310
```

podobně determinant matice A

```
octave:9> det(A)
ans = -241
```

Na příkladu kvadratické $4x^2 + x - 3 = 0$ rovnice můžeme demonstrovat schopnost hledat kořeny polynomu. Z koeficientů polynomu sestavíme matici v pořadí od nejvyššího k nejnižšímu

```
octave:10> koeficienty=[4 1 -3]
koeficienty =

   4   1  -3
```

Kvadratický polynom na levé straně můžeme pro kontrolu nechat vypsat ve zvolené proměnné (v tomto případě x)

```
octave:11> polyout(koeficienty, 'x')
4*x^2 + 1*x^1 - 3
```

Hledané řešení – kořen polynomu – vrací funkce matice koeficientů

```
octave:12> roots(koeficienty)
ans =

  -1.00000
   0.75000
```

Získáváme tak řešení $x_1 = -1$ a $x_2 = 0{,}75$.

Řešit můžeme i úlohu opačnou – najít mnohočlen ke známým kořenům. Např. pro hodnoty $x_1 = -0{,}7$, $x_2 = 0{,}9$ jsou kořeny polynomu s koeficienty

```
octave:13> poly([-.7, .9])
ans =

   1.00000  -0.20000  -0.63000
```

tj. $x^2 - 0{,}2x - 0{,}63$.

Program umožňuje nalézt jak součin, tak podíl dvou polynomů opět pomocí matic jejich koeficientů. Např. pro polynomy $x^4 - 8x^3 + 16x^2 - 7x - 2$ a $x^2 - - 3x + 2$ vytvoříme matice `koef1=[1 -8 16 -7 -2]; koef2=[1 -3 2];`. Koeficienty součinu obou polynomů získáme příkazem

```
octave:15> conv(koef1,koef2)
ans =

    1  -11   42  -71   51   -8   -4
```

a koeficienty podílu

```
octave:16> podil=deconv(koef1,koef2)
podil =

   1  -5  -1
```

Poté můžeme vypsat i odpovídající polynom např. v proměnné y

```
octave:17>  polyout(podil,'y')
1*y^2 - 5*y^1 - 1
```

neboli $y^2 - 5y - 1$.

8.4 Práce se soubory

Nutnou podmínku práce s jakýmkoli programem představuje možnost ukládat informace do souborů na pevném disku, síti či přenosném médiu. Některé základní funkce jako zjištění aktuálního adresáře či výpis souborů v něm již byly zmíněny v části 8.1. Nyní zadefinujeme matici prvků po jedné od 1 do 10 (tj. s krokem rovným 1); protože pro nás bude výhodnější sloupcová matice namísto řádkové, použijeme operaci transponování matice `'`

```
octave:1> mereni=[1:1:10]'
mereni =

   1
```

```
   2
   3
   4
   5
   6
   7
   8
   9
  10
```

Nyní k téže matici přidejme sloupec s (pseudo)náhodnými hodnotami okolo 5. Použijeme k tomu generátor (pseudo)náhodných čísel `rand(i,j)`, který vrací matici $i \times j$ pseudonáhodných čísel v intervalu $\langle 0{,}1 \rangle$, vynásobením číslem $0{,}1$ docílíme toho, že čísla v druhém sloupci se budou od 5 lišit až na druhém desetinném čísle. Výsledná dvousloupcová matice tak připomíná výsledek deseti měření nějaké veličiny (tím jsme se inspirovali i pro název proměnné)

```
octave:2> mereni=[mereni [5+.1*rand(10,1)]]
mereni =

   1.0000   5.0124
   2.0000   5.0323
   3.0000   5.0559
   4.0000   5.0607
   5.0000   5.0124
   6.0000   5.0832
   7.0000   5.0341
   8.0000   5.0759
   9.0000   5.0659
  10.0000   5.0149
```

K téže matici můžeme přidat další sloupec (dvěma náhodnými čísly zvětšujeme náhodné „rozmazání“ hodnot ve třetím sloupci)

```
octave:3> mereni=[mereni [9.81*12.5+0.05*rand(10,1)-0.07*rand(10,1)]]

mereni =

    1.0000    5.0124  122.6048
    2.0000    5.0323  122.6408
    3.0000    5.0559  122.5813
    4.0000    5.0607  122.6648
    5.0000    5.0124  122.6527
```

```
    6.0000    5.0832  122.6507
    7.0000    5.0341  122.5876
    8.0000    5.0759  122.6040
    9.0000    5.0659  122.6187
   10.0000    5.0149  122.6318
```

Získanou matici `mereni` nyní uložíme do souboru `mereni.txt` jako čistý text (ne v binárním formátu), aby ji bylo možné načíst i jiným programem (např. tabulkovým kalkulátorem *Excel*)

```
octave:4> save -text mereni.txt mereni
```

Po vymazání proměnné z paměti

```
octave:5> clear mereni
```

ji můžeme načíst ze souboru (jméno načtené proměnné bude odpovídat jménu souboru bez přípony)

```
octave:6> load mereni.txt
```

Zkonstruovanou matici využijeme k ukázce možnosti, jak s pomocí *GNU Octave* zpracovat výsledky měření. Podíváme-li se na hodnoty v matici, vidíme, že první sloupec by mohl odpovídat číslu měření, druhý času $t \approx 5\,\mathrm{s}$ a třetí dráze volného pádu $s \approx gt^2/2$. Vybereme-li z načtené matice jednotlivé sloupce a uložíme je do proměnných

```
octave:7> pokus=mereni(:,1); cas=mereni(:,2); draha=mereni(:,3);
```

lze vypočítat aritmetické průměry

```
octave:8> [mean(cas) mean(draha)]
ans =

    5.0448  122.6237
```

odchylky

```
octave:9> [center(cas) center(draha)]
ans =

  -0.0323554  -0.0189170
  -0.0124431   0.0171056
   0.0111785  -0.0424311
   0.0159147   0.0410511
```

```
  -0.0323656   0.0289959
   0.0384046   0.0269348
  -0.0106577  -0.0361146
   0.0310966  -0.0197190
   0.0211222  -0.0050090
  -0.0298949   0.0081032
```

a standardní odchylky $\sqrt{\sum_{i=1}^{n}(x-\bar{x})^2/(n-1)}$

```
octave:10> [var(cas) var(draha)]
ans =

  0.00072476  0.00083179
```

S problematikou měření souvisí i regrese – hledání závislosti mezi veličinami. Zatímco např. *Excel* standardně umožňuje pouze regresi lineární, *GNU Octave* umožňuje veličinami proložit polynom požadovaného stupně, v naší ukázce zvolíme 2. Zadefinujme nejprve veličiny x a y jako jednořádkové matice (vektory)

```
octave:11> x=1:1:10, y=x.^2+x+3+0.5*rand(1,10)-0.3*rand(1,10)
x =

   1   2   3   4   5   6   7   8   9  10

y =

 Columns 1 through 8:

    5.1096    9.1916   15.0361   23.0593   33.1781
                                  44.8745   59.1035   75.3419

 Columns 9 and 10:

   92.9817  112.8734
```

Koeficienty kvadratického polynomu vyhovující podmínce nejmenších čtverců najdeme příkazem

```
octave:12> polyfit(x,y,2)
ans =

  0.99685  1.02112  3.08015
```

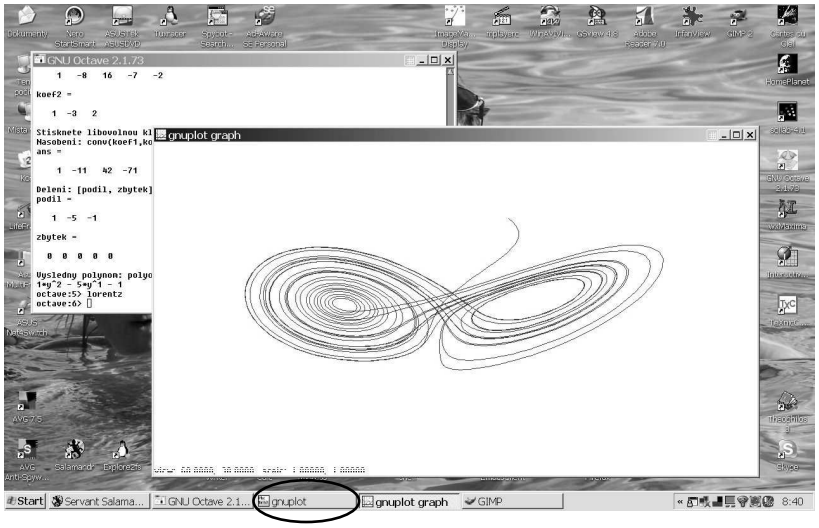
nejlepší aproximací 2. stupně je tedy polynom $y = 0{,}9968x^2 + 1{,}021x + 3{,}08$.

Na závěr uveďme, jak lze vypsat např. posledních 10 zadaných příkazů

```
octave:13> history 10
```

8.5 Kreslíme obrázky

Nezbytnou součástí dynamického modelování je i vykreslení vypočtených hodnot a funkčních závislostí. Věnujme se proto na závěr kapitoly základním příkazům dvourozměrné a třírozměrné grafiky.



Obr. 8.2: Okno s obrázkem vykresleným programem GNUplot

Jak již bylo řečeno, *GNU Octave* pro vykreslování grafů automaticky volá *GNUplot*, v OS Windows se na panelu ve spodní části obrazovky objeví dvě nové položky (viz. obr. 8.2). Zatímco okno s vlastním grafem můžeme bez problémů zavřít a při dalším požadavku se nám vykreslí nový graf (nebo překreslí stávající s novými parametry), druhé příkazové okno samotného *GNUplotu* (na obr. 8.2 je vyznačeno) *zavřít nesmíme*; pokud se to stane, přeruší se komunikace (pipeline)

mezi *GNU Octave* a *GNUplotem* a žádný další obrázek se již nevykreslí – je nutno *GNU Octave restartovat*.[7]

8.5.1 Dvourozměrná grafika

Jednoduchý graf funkce $y = \sin x$ pro $x \in \langle 0{,}10 \rangle$ vykreslíme pomocí příkazů[8]

```
octave:1>  x=0:pi/4:10; plot(x,sin(x))
```

Nejprve definujeme hodnoty nezávisle proměnné x od 0 do 10 s krokem $\pi/4$ a poté vykreslíme příslušnou závislost. Pokud se nám jako v tomto případě zná graf příliš hranatý (program spojuje jednotlivé body úsečkami, i když lze použít i splajny), musíme zjemnit dělení např. zjemněním kroku na $\pi/100$. Rozsah nezávisle proměnné můžeme zadat i volbou počáteční a konečné hodnoty spolu s počtem dělicích bodů. Chceme-li např. výše použitý interval rozdělit na sto intervalů (mezi 101 body), použijeme

```
octave:2>  x=linspace(0,10,101); plot(x,sin(x))
```

K příkazu plot můžeme zadat i další nepovinné parametry, např. vykreslit i body se spočítanými funkčními hodnotami (což může být užitečné při znázorňování diskrétních dat)

```
octave:3>  x=linspace(0,10,11); plot(x,sin(x),'-o')
```

nebo změnit barvu

```
octave:4>  x=linspace(0,10,11); plot(x,sin(x),'-b')
```

popř. vykreslit několik grafů najednou s různými parametry pro každý z nich

```
octave:5>  x=linspace(0,10,41);
octave:6>  plot(x,sin(x),'-*',x,cos(x),'ob')
```

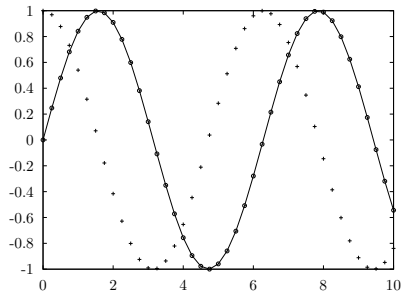
výsledek vidíme na obr. 8.3, jinou volbu

```
octave:7> plot(x,-sin(x),'^',x,cos(x),'-ob;kosinus;', \
x,-sin(x).^2+.5,'mL;schody;')
```

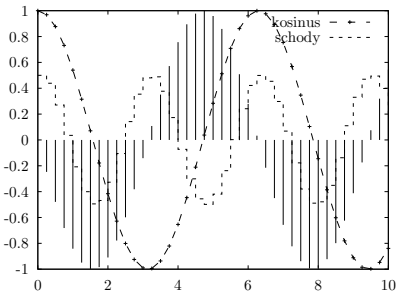
pak na obr. 8.4. Nezadáme-li parametry žádné, budou grafy vykresleny čarami

[7] V operačním systému Linux se toto druhé okno vůbec neotevírá, pouze samotný graf, takže se s tímto problémem nesetkáme.

[8] Z důvodu úspory místa zde nebudeme reprodukovat všechny grafy, předpokládáme, že čtenář má možnost si výstup jednotlivých příkazů bezprostředně ověřit přímým zadáním do programu.



Obr. 8.3: Dva grafy v jednom obrázku s různými parametry



Obr. 8.4: Totéž s jinou volbou parametrů příkazu plot

(úsečkami spojujícími jednotlivé body), jejichž barva, resp. typ přerušované čáry u černobílého výstupu, se budou automaticky pro jednotlivé čáry lišit v závislosti na typu výstupu (obrazovka, postscriptový soubor, obrázek png apod.)[9] Každému lze určitě doporučit, aby se nebál s volbami experimentovat a sám si našel kombinaci parametrů, která jemu nejvíce vyhovuje. Z předchozích ukázek vidíme, že jimi pro každý z grafů můžeme definovat:

- *typ grafu* čárový (výchozí, `-`), tečkovaný (`.`), schodovitý (`L`) nebo vynášecí (`^`).
- *barvu* a to buď písmenem `r`, `g`, `b`, `m`, `c` či `w` nebo čísly 1–6, tj. v uvedeném pořadí červená (red), zelená (green), modrá (blue), purpurová (magenta), azurová (cyan) a bílá (white). Vyzkoušením se přesvědčíme, že i číslům 7–9 jsou barvy přiřazeny ovšem bez komentáře v manuálu.
- *bodový graf* neboli vykreslení bodů buď přímo uvedením symbolu, který se má v daném bodě vykreslit jako `*`, `+`, `o`, `x`, nebo dvojciferným číslem, v němž první cifra určuje barvu a druhá symbol, opět lze experimentovat s čísly 1–9). Z uvedených ukázek vidíme, že vyznačení bodů lze kombinovat s vykreslením čar, např. `-*`.
- *popisek* uvádíme mezi středníky, např. `;kosinus;`, středník na konci popisku nesmí chybět. Výchozí popisky mají tvar „line 1“, „line 2“ atd.

Pokud je navíc zapnut mód `mouse` (v OS Windows automaticky ve výchozím nastavení), vidíme aktuální hodnoty souřadnic bodů, nad nimiž se pohybujeme myší, v levém dolním rohu obrazovky. Stisknutím prostředního tlačítka můžeme značku zvoleného bodu s těmito hodnotami přidat do grafu, tahem při stisk-

[9] Nastavení terminálu, tj. typu čar a bodů lze zjistit v programu *GNUplot* příkazem `test`.

nutém pravém tlačítku můžeme vybrat detailní výběr nějaké části grafu (pak ale před dalším obrázkem musíme anulovat veškerá nastavení obrázku). Tento doplněk, dostupný pouze v novějších verzích, lze kdykoli zapnout příkazem `__gnuplot_set__ mouse` a naopak inaktivovat `__gnuplot_set__ nomouse`.

Chceme-li vykreslené obrázky vytisknout popř. zařadit do textu, uvítáme možnost měnit vzhled os, jejich popisek, poměru stran obrázku apod. U řady obrázků může být kód (posloupnost příkazů) pro tyto změny stejně dlouhý jako kód pro samotný výpočet hodnot. Uživatelé *Matlabu*® mají k dispozici některé příkazy typu `axis`, nejvíce možností prozatím skýtá zadávání příkazu samotného *GNUplotu*, které bývají uvozeny `__gnuplot_set__` nebo `__gnuplot_raw__`. V našem textu se omezíme jen na několik nejdůležitějších možností, podrobný popis by nutně suploval manuál k programu GNUplot (viz např. [25]). Dodejme, že ve starších verzích programu se namísto uvedených příkazů používal tvar `gset` resp. `graw`, které sice zůstávají stále implementovány, ale při jejich prvním volání nás varovná hláška upozorní, že se s nimi v dalším vývoji již nepočítá a pro začínajícího uživatele nemá tudíž smysl si na ně zvykat.

Ukažme si je na příkladu funkce přirozeného logaritmu $\ln x$.[10] Graf logaritmické funkce pro $x \in \langle 0{,}10 \rangle$ vykreslíme příkazem

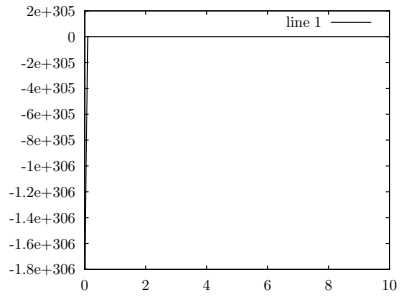
```
octave:8> x=linspace(0,10,101); plot(x,log(x));
```

jehož výsledek na obr 8.5 nás určitě neuspokojí. Zavřeme proto okno s obrázkem (ale ne s programem GNUplot!) a změníme rozsah vykreslených hodnot na ose y na hodnoty $y \geqq -10$ (to musíme udělat většinou, pokud funkce má ve zobrazovaném intervalu singulární bod, v němž hodnoty rostou nebo klesají k $\pm\infty$) nastavením parametru `yrange`. Dále přidáme souřadnicovou síť (`grid`), vypneme popisek v pravém horním rohu (`nokey`) a změníme směr značek na osách směrem ven z obrázku (`tics out`). Nakonec obrázek překreslíme s novým nastavením pomocí příkazu `replot`. Celá posloupnost příkazů pak vypadá následovně

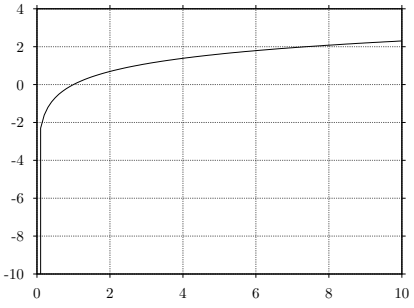
```
octave:9> __gnuplot_set__ yrange [-10:]
octave:10> __gnuplot_set__ grid
octave:11> __gnuplot_set__ nokey
octave:12> __gnuplot_set__ tics out
octave:13> replot
```

její výstup je na obr. 8.6. K dispozici jsou i logaritmická měřítka na osách, buď na ose x nebo y příp. na osách obou

[10] Musíme dát pozor, že pro přirozený logaritmus o základu e čísla b, tj. pro $\ln b$ použijeme příkaz `log(b)`, zatímco pro dekadický logaritmus o základu 10 $\log b$ příkaz `log10(b)`. Logaritmus o libovolném základě pak získáme podílem logaritmů jednoho z těchto typů podle známých vztahů $\log_a b = \ln b / \ln a = \log b / \log a$.



Obr. 8.5: Graf logaritmické funkce v základním nastavení



Obr. 8.6: Upravený graf logaritmické funkce

```
octave:14>  semilogx(x,log(x))
octave:15>  semilogy(x,log(x))
octave:16>  loglog(x,log(x))
```

Dále můžeme pochopitelně upravit rozsah hodnot zobrazených na ose x[11] nastavením parametru `xrange` a použít desetinné čárky namísto tečky parametrem `decimalsign`

```
octave:17>  __gnuplot_set__ xrange [0.1:]
octave:18>  __gnuplot_set__ decimalsign ','
```

Všechna nastavení anulujeme na výchozí příkazem `__gnuplot_raw__ ("reset;")`. Ve většině případů se tím aktivuje vykreslování pouhých bodů (ne čar), které musíme opět zapnout. Celá dvojice příkazů pak vypadá

```
octave:19> __gnuplot_raw__ ("reset;")
octave:20> __gnuplot_set__ data style lines;
```

Vykreslit lze i parametricky definované křivky, *musíme však dát pozor, aby matice definující parametry měly stejný rozměr, tj. stejný počet prvků,* v našem případě 361. Pro $\varphi \in \langle 0{,}8\pi \rangle$ a $r \in \langle 0{,}2 \rangle$ nakreslíme spirálu s rostoucím poloměrem o rovnicích $x = r\cos\varphi$, $y = r\sin\varphi$ a také křivku popsanou v polárních souřadnicích rovnicí $r = 2\sin(2\varphi)$, tj. $x = 2\sin(2\varphi)\cos\varphi$, $y = 2\sin(2\varphi)\sin\varphi$. Použijeme postupně příkazy

[11] Označení `xrange` resp. `yrange` či v trojrozměrném případě `zrange` se nezmění, ani když pracujeme s jinými proměnnými než x,y,z.

```
octave:21> phi=linspace(0,8*pi,361);
octave:22> r=linspace(0,2,361);
octave:23> plot(r.*cos(phi),r.*sin(phi),\
               2*sin(2*phi).*cos(phi),2*sin(2*phi).*sin(phi));
```

Správný tvar křivek vyžaduje stejné měřítko na obou osách

```
octave:24> axis('equal'), replot
```

zobrazení os můžeme i vypnout

```
octave:25> axis('off'), replot
```

nebo zapnout zobrazení os procházejících počátkem souřadnic příslušným typem čáry (v našem případě plná tenká čára linetype -1[12])

```
octave:26> __gnuplot_set__ xzeroaxis lt -1
octave:27> __gnuplot_set__ yzeroaxis lt -1
octave:28> replot
```

a nevykreslovat rám okolo grafu

```
octave:29> __gnuplot_set__ noborder
octave:30> replot
```

V poslední ukázce vyplníme barvou oblast mezi grafem funkce a osou x o rovnici $y = 0$. Nejprve opět pro jistotu vymažeme předchozí nastavení grafů

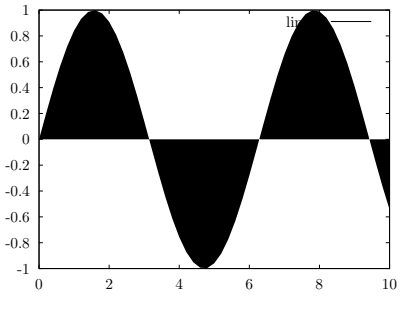
```
octave:31> __gnuplot_raw__ ("reset;")
octave:32> __gnuplot_set__ data style lines;
```

Pro hodnoty $x \in \langle 0{,}10 \rangle$ opět vykreslíme graf funkce $\sin x$. Změníme styl vykreslování na vyplňování ohraničené osou x

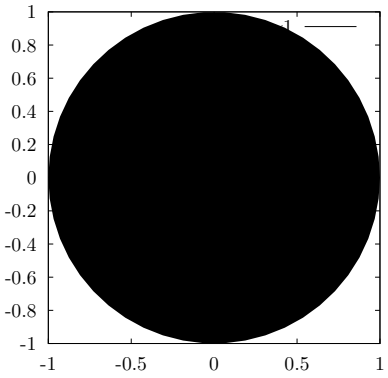
```
octave:33> __gnuplot_set__ style data filledcurves y1=0
octave:34> x=linspace(0,10,51)'; plot(x,sin(x))
```

Vyplnit můžeme i uzavřené křivky, pro jednoduchost zvolíme jednotkový kruh s hranicí $x = \cos\varphi$, $y = \sin\varphi$, přitom musíme opět nastavit stejné měřítko na obou osách

```
octave:35> phi=linspace(0,2*pi,51);
octave:36> __gnuplot_set__ style data filledcurves closed
octave:37> axis('equal')
octave:38> plot(cos(phi)', sin(phi)');
```



Obr. 8.7: Vyplněná oblast ohraničená grafem funkce



Obr. 8.8: Vyplněný kruh

Výsledky pak vidíme na obr. 8.7 a 8.8. Je zřejmé, že implementace této možnosti je prozatím spíše v počátcích (více je rozvinuta již v beta-verzi *GNUplotu*). Uživatelé, kteří by chtěli vyplňovat oblasti zvolenými barvami a kombinovat vyplňování s kreslením křivek mohou doinstalovat balík *EpsTk* (`http://www.epstk.de/`). Ten využívá možností postscriptu a je poměrně dobře a snadno použitelný v operačním systému Linux.

Před dalším kreslením nesmíme zapomenout vrátit se k výchozímu nastavení

```
octave:39> __gnuplot_raw__ ("reset;")
octave:40> __gnuplot_set__ style data lines;
octave:41> __gnuplot_set__ data style lines;
```

nebo restartovat *GNU Octave*.

8.5.2 3D grafika

K snadné ukázce trojrozměrných grafů můžeme použít demonstračních funkcí `sombrero(n)` a `peaks(n)`, kde n určuje hustotu sítě bodů, v nichž budou spočítány funkční hodnoty (množství intervalů podél os x a y, tj. poslední argument

[12] Číslování čar a dalších parametrů nastíněno spolu s parametry příkazu `plot` na str. 109, hodnotě -1 odpovídá tenká plná čára.

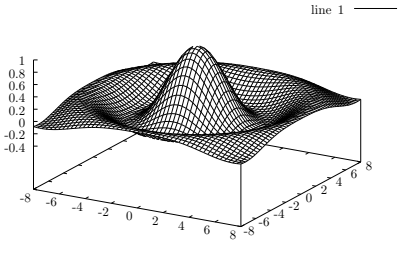
funkce `linspace`). Funkce odpovídají závislostem

$$\begin{aligned}
\texttt{sombrero}\,(x,y) &= \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}, \quad x,y \in \langle -8,8 \rangle \\
\texttt{peaks}\,(x,y) &= 3\,(1-x)^2 \exp\left[-x^2-(y+1)^2\right] - \\
&\quad -10\left(\frac{x}{5}-x^3-y^5\right)\exp\left(-x^2-y^2\right) - \\
&\quad -\frac{1}{3}\exp\left[-(x+1)^2-y^2\right], \quad x,y \in \langle -3,3 \rangle
\end{aligned}$$

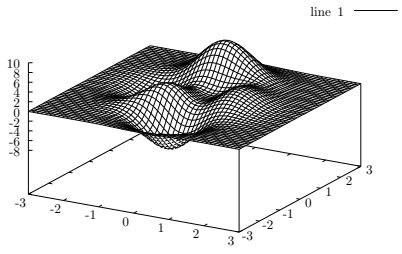
Výstup příkazů

```
octave:1> sombrero(51)
octave:2> peaks(51)
```

vidíme na obrázcích 8.9 a 8.10.



Obr. 8.9: `sombrero(51)`



Obr. 8.10: `peaks(51)`

Zadefinujme si nyní postupně vlastní funkci dvou proměnných. Jak již bylo naznačeno, pro výpočet jejích funkčních hodnot musíme vytvořit „síť“ bodů (meshgrid) se všemi možnými kombinacemi hodnot nezávisle proměnných, v našem případě x a y. Pro $x,y \in \langle -2,2 \rangle$ k tomu použijeme příkaz `meshgrid` s tím, že dělení na osách samotných definujeme podobně jako u dvourozměrných grafů pomocí `linspace` (nebo pomocí kroku ve zvoleném intervalu, např. `-2:.1:2` – snadno ověříme, v čem nejsou definice stejné). Pro funkci

$$z = \left[(x-0{,}3)^2+y^2\right]\exp\left[-x^2-(y-0{,}2)^2\right]$$

zadáme

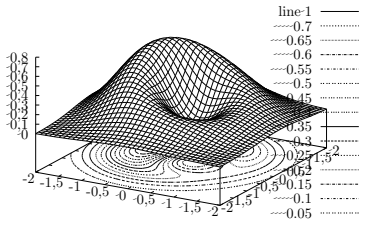
```
octave:3> [x,y]=meshgrid(linspace(-2,2,41),linspace(-2,2,41));
octave:4> z=((x-.3).^2+y.^2).*exp(-x.^2-(y-.2).^2);
octave:5> mesh(x,y,z);
```

Připomeňme, že středníkem na konci řádku jsme zakázali výpis delších matic na obrazovku a vlastní vykreslení provedl příkaz `mesh`, jehož argumentem jsou matice jednotlivých proměnných (x a y přitom musí tvořit síť „meshgrid“). Pro zachování kompatibility s *Matlabem*® je téhož výsledku dosáhnout také příkazem `surf(x,y,z)`, ale plocha se – na rozdíl od *Matlabu*® – většinou nevyplní barvami (viz níže).

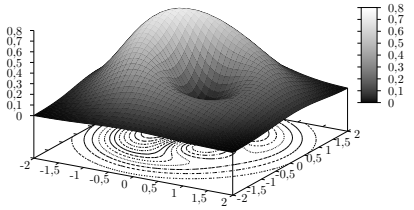
Nyní můžeme opět graf upravovat změnou patřičných parametrů. Zobrazení „vrstevnic“ tj. čar spojujících body se stejnou hodnotou z aktivujeme parametrem `contour`, jejich hustotu zvýšíme na 30 vrstevnic mezi minimem a maximem parametrem `cntrparam`, poté již výše zmíněným způsobem nahradíme desetinnou tečku čárkou parametrem `decimalsign` a nakonec obrázek překreslíme v novém nastavení pomocí `replot`

```
octave:6> __gnuplot_set__ decimalsign ","
octave:7> __gnuplot_set__ contour
octave:8> __gnuplot_set__ cntrparam level auto 30
octave:9> replot
```

Výsledek vidíme na 8.11. Pokud bychom chtěli vykreslit vrstevnice odpovídající konkrétním hodnotám, použijeme např. `__gnuplot_set__ cntrparam levels discrete .1, .5, .9` pro hodnoty $z = 0{,}1$, $z = 0{,}5$ a $z = 0{,}9$. Před dalšími úpravami zopakujeme nastavení vyzkoušená již u dvourozměrných



Obr. 8.11: Graf spolu s vrstevnicemi



Obr. 8.12: Vrstevnice spolu s barevným stínováním plochy

obrázků – deaktivujeme popisky parametrem `nokey` a změníme směr značek na

osách `tics out`. Potom naopak zapneme barevné stínování plochy parametrem `pm3d` a změníme použité barvy pomocí `palette`.[13]

```
octave:10> __gnuplot_set__ nokey
octave:11> __gnuplot_set__ tics out
octave:12> __gnuplot_set__ pm3d
octave:13> __gnuplot_set__ palette defined \
             ( 0 "blue", 3 "green", 6 "yellow", 10 "red" )
octave:14> replot
```

výsledek znázorňuje obr. 8.12. Pokud nechceme vykreslovat barevný obdélník v pravé části znázorňující přiřazení barev hodnotám z, stačí zadat

```
octave:15> __gnuplot_set__ nocolorbox
octave:16> replot
```

K dispozici je mnoho barevných „palet“, jež si může uživatel nadefinovat použitím nejrůznějších kombinací, z ukázky vidíme, že přiřazujeme barvy relativním pozicím hodnot z mezi minimem (0) a maximem (10) na desetistupňové škále.[14] Spektrum od modré po červenou lze získat např. příkazy

```
octave:17> __gnuplot_set__ palette defined \
             ( 0 "dark-blue", 1 "blue", 2 "cyan", \
               3 "yellow", 4 "red" , 5 "dark-red" )
octave:18> replot
```

nebo

```
octave:19> __gnuplot_set__ palette rgbformulae 33,13,10;
octave:20> replot
```

Pro černobílé obrázky vystačíme pouze se stupni šedé, např.

```
octave:21> __gnuplot_set__ palette defined \
                 ( 0 "dark-grey", 1 "white")
octave:22> replot
```

nebo

```
octave:23> __gnuplot_set__ palette defined \
                  ( 0.5 0.5 0.5 0.5, 1 1 1 1)
octave:24> replot
```

[13] Doporučujeme za každou změnou provést překreslení příkazem `replot`, aby se čtenář sám přesvědčil, jaké změny jednotlivé kroky s grafem způsobí. Funkce spojené se stínováním jsou dostupné až novějších verzích *GNUplotu*.

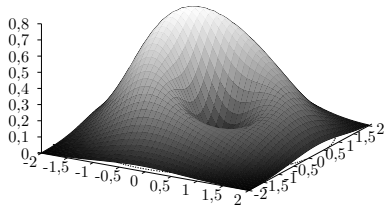
[14] Hodnoty 0 a 10 neodpovídají skutečným hodnotám z, jde o relativní stupnici.

K dalším na internetu doporučovaným paletám patří také kombinace `rgbformulae 21,22,23` (odpovídá „hot“ přechodu černá-červená-žlutá-bílá).

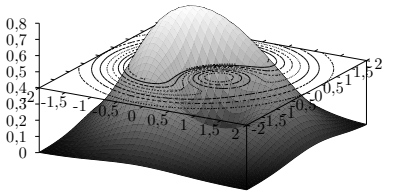
Pokud chceme změnit polohu plochy vůči základní rovině xy, musíme změnit parametr `ticslevel`, udávající polohu nejnižších bodů plochy relativně k celému rozsahu osy z. Volbou

```
octave:25> __gnuplot_set__ ticslevel 0
```

„přilepíme“ plochu na základnu (obr. 8.13). Záporná čísla mají za následek vyzdvižení základny nad graf (nebo jeho část), jak vidíme z obr. 8.14.



Obr. 8.13: Volba `ticslevel -0.5`



Obr. 8.14: Volba `ticslevel -0.5`

Další proměnnou, kterou lze nastavit, je pohled na vykreslenou plochu prostřednictvím parametru `view`. Má celkem čtyři argumenty – úhel $\vartheta \in \langle 0{,}180^\circ \rangle$ rotace okolo osy x, úhel $\varphi \in \langle 0{,}360^\circ \rangle$ rotace okolo osy z, měřítko os v základní rovině a konečně měřítko svislé osy z. Přednastavené hodnoty jsou `60, 30, 1, 1`. Na náš obrázek zkusme aplikovat např.

```
octave:26> __gnuplot_set__ view 15, 135, 1, 4
```

Pokud je zapnut mód `mouse` (v OS Windows ve výchozím nastavení), vidíme aktuální hodnoty parametru view v levém dolním rohu obrazovky a měnit je můžeme pomocí myši – při stisknutém levém tlačítku obrázkem otáčíme, při stisknutém prostředním měníme měřítko ve směru osy z. Jak bylo řečeno, tento doplněk, dostupný podobně jako barevné stínovaní v novějších verzích, lze kdykoli zapnout příkazem `__gnuplot_set__ mouse`.

Speciálním a hodně užívaným je pohled „ze shora“ s vrstevnicemi vykreslenými přes barevné stínování. Dosáhneme toho nastavením

```
octave:27> __gnuplot_set__ view map
```

Protože hodnoty x i y leží ve stejném intervalu, nastavíme opět stejné měřítko na osách `axis('equal')`. Ukažme si také změnu popisu os, k čemuž slouží postupně parametry `xtics`, `mxtics` a `xlabel` pro osu x a obdobné parametry (např. `mytics` nebo `zlabel` pro osy zbývající)[15]. V naší ukázce umístíme podél osy x hlavní značky automaticky od -2 s krokem $0{,}5$ a na ose y konkretní popisky do zvolených bodů

```
octave:28>  __gnuplot_set__ xtics -2, 0.5
octave:29> __gnuplot_set__ ytics \
                 ("1. bod" -2, "2. bod" 0, "3. bod" 1)
```

Dále aktivujeme menší značky tak, že určíme jejich počet mezi značkami hlavními (což jde, pokud se opakují podle nějakého algoritmu, jako v případě naší osy x)

```
octave:30>  __gnuplot_set__ mxtics 10
```

Nyní ještě doplníme označení os, u něhož můžeme zvolit i jiný font. V našem případě použijeme pro osu y standardní font Symbol pro psaní symbolů a řeckých písmen, pro osu x skloněné písmo Times-Italic o velikost 30 bodů.[16]

```
octave:31>  __gnuplot_set__ xlabel "{/Times-Italic=30 osa x}"
octave:32>  __gnuplot_set__ ylabel "{/Symbol abc}"
octave:33>  replot
```

Výsledek posloupnosti příkazů je uveden na obr. 8.15.

Chceme-li podobně jako v *Matlabu*® vykreslit stínovanou plochu spolu s čarami spojujícími body na ní, použijeme posloupnost příkazů

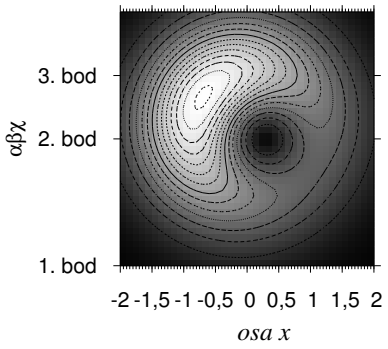
```
octave:1> [x,y]=meshgrid(linspace(-2,2,41),linspace(-2,2,41));
octave:2>  z=((x-.3).^2+y.^2).*exp(-x.^2-(y-.2).^2);
octave:3> global gnuplot_has_pm3d=1;
octave:4> gnuplot_has_pm3d=1;
octave:5> surf(x,y,z)
```

Protože funkce `surf` nastavuje příslušné parametry pro `mesh`, budou se poté opět obě funkce chovat stejně. Stejného efektu docílíme i postupnými kroky

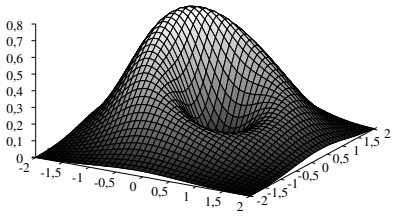
```
octave:1> [x,y]=meshgrid(linspace(-2,2,41),linspace(-2,2,41));
octave:2>  z=((x-.3).^2+y.^2).*exp(-x.^2-(y-.2).^2);
```

[15] Opět platí, že názvy se nezmění, i když pracujeme s jinými proměnnými.

[16] Změna fontů se nemusí projevit v každém prostředí, např. na obrazovce počítače nebo v obrázku png. V postscriptových souborech ji ale lze použít téměř vždy, užití speciálních, třeba TEX-ovských fontů (většinou v tomto textu), závisí na konkrétní instalaci prohlížeče postscriptu.



Obr. 8.15: Pohled ze shora s upravenými popisky



Obr. 8.16: Barevné stínování spolu s vykreslením čar.

```
octave:3> __gnuplot_set__ pm3d at s hidden3d 1
octave:4> __gnuplot_set__ nohidden3d
octave:5> __gnuplot_set__ nosurf
octave:6> __gnuplot_set__ line style 1 lt 1 lw 1
octave:7> mesh(x,y,z)
```

Výsledek po úpravě dalších známých parametrů

```
octave:8> __gnuplot_set__ nokey
octave:9> __gnuplot_set__ nocolorbox
octave:10> __gnuplot_set__ tics out
octave:11> __gnuplot_set__ decimalsign ","
octave:12> __gnuplot_set__ palette rgbformulae 33,13,10
octave:13> __gnuplot_set__ ticslevel 0
```

vidíme na obr. 8.16.

Zbývá uvést, jak vykreslené grafy uložit pro další použití. Z mnoha dostupných formátů, z nichž mnohé jsou velmi speciální se zaměříme na tři. Jako první uveďme PNG (Portable Network Graphics), který postupně i na internetu nahrazuje formát GIF dostupný ve starších verzích při zachování jeho vlastností. K exportu našeho obrázku do souboru `pokus.png` postupně zadáme:

```
octave:14> __gnuplot_set__ term png large
octave:15> __gnuplot_set__ output 'pokus.png'
octave:16> replot
```

přičemž nepovinný údaj `large` v prvním řádku požaduje větší písmo popisek os. Obr. 8.16 byl získán exportem do postscriptového souboru `pokus.eps` (přesněji jde Encapsulated PostScript vyvinutý firmou Adobe Systems), u něhož navíc požadujeme pro všechny popisky font „Times-Roman“ o velikosti 25 bodů

```
octave:17> __gnuplot_set__ term post eps enh "Times-Roman" 25
octave:18> __gnuplot_set__ output 'pokus.eps'
octave:19> replot
```

Z dalších možností uveďme formát SVG (Scalable Vector Graphics – škálovatelná vektorová grafika neboli značkovací jazyk popisující dvojrozměrnou vektorovou grafiku pomocí XML), který by se měl v budoucnu stát základním otevřeným formátem pro vektorovou grafiku na internetu (většina používaných formátů jako GIF, JPG nebo výše zmíněné PNG jsou formáty pro rastrovou grafiku). Čtenář jistě sám doplní, že v takovém případě by předcházející příkazy obsahovaly `term svg` a např. `output "pokus.svg"`. Tento formát je také doporučován pro náčrtky a grafy v příspěvcích do Wikipedie [1].

Protože nastavení pro jeden obrázek mohou být zcela nežádoucí pro obrázky jiné, bude v ukázkách v příští kapitole vždy na počátku vymazání veškerých proměnných příkazem `clear`, vymazán terminál příkazem `clc` a zrušeno nastavení grafického výstupu

```
octave:1> clear;
octave:2> clc;
octave:3> __gnuplot_raw__ ("reset;")
octave:4> __gnuplot_set__ data style lines;
```

Náš stručný přehled základních funkcí není zdaleka vyčerpávající a slouží především k lepšímu pochopení programů a procedur uvedených v následujících kapitolách. Více informací najde čtenář buď v [11, 15] popř. v jiných internetových zdrojích.

Část 9

Pokročilejší funkce

Ke studiu a sestavení dynamických modelů potřebujeme ještě cykly a logické proměnné (rozhodování), jež patří k nezbytnému arzenálu každého programovacího jazyka. Jejich použití si postupně vysvětlíme na algoritmech pro numerické řešení algebraických rovnic v této kapitole a diferenciálních rovnic v kapitole následující. Popsané skripty budou jednoduchou alternativou k předdefinovaným procedurám, které k těmto účelům program přímo nabízí a jež by také rozhodně neměly uniknout naší pozornosti.

9.1 Numerické řešení algebraických rovnic

Pod numerickým řešením algebraických rovnic rozumíme přibližný výpočet reálného kořene (popř. kořenů) rovnice

$$f(x) = 0$$

na určitém intervalu (a,b). Základem všech metod je *Bolzanova věta*[1], zaručující existenci hledaného řešení:

Bolzanova věta: Nechť f je funkce spojitá na intervalu $\langle a,b\rangle$. Mají-li čísla $f(a)$ a $f(b)$ různá znaménka, tj. $f(a)\cdot f(b) < 0$, pak existuje alespoň jeden bod $x^* \in (a,b)$ v němž platí $f(x^*) = 0$.

Pokud bude zaručeno, že daná funkce je na intervalu $\langle a,b\rangle$ rostoucí nebo klesající, existuje takový bod x^* právě jeden. Používané metody se liší způsobem sestavení tzv. iterační posloupnosti $\{x_n\}$ takové, že $\lim x_n = x^*$. Absolutně přesný výpočet této limity však většinou není možný a proto se spokojujeme s k-tým členem iterační posloupnosti $|x^* - x_k| < \varepsilon$, kde ε je předem dané kladné číslo – požadovaná přesnost řešení rovnice. Výběr konkrétní metody závisí na vlastnostech funkce f na intervalu (a,b). Jedním z důležitých kriterií je i rychlost konvergence – kolik členů posloupnosti je zapotřebí k dosažení požadované přesnosti.

K nejpoužívanějším metodám patří:

1. *Metoda půlení intervalu (metoda bisekce)*, jejíž výhodou je jednoduchá konstrukce zmíněné posloupnosti (středy intervalů) a minimální požadavky na funkci f, nevýhodou je v některých případech pomalá konvergence.

[1] Zatímco na internetu najdeme Bolzanovu větu poměrně snadno (viz např. [13]), klasické učebnice matematické analýzy jako [10] popř. známá přehledná publikace [17] ji uvádí jen jako jednu z bezejmenných vět o vlastnostech spojitých funkcí.

2. *Metoda tětiv (regula falsi, lineární interpolace)*, kde princip vytváření iterační posloupnosti spočívá postupném nahrazování funkce $f(x)$ na příslušném intervalu úsečkami a hledání průsečíků těchto úseček s osou x.
3. *Metoda tečen (Newtonova metoda)*, v níž jsou kladeny větší požadavky na funkci f (existence spojité první i druhé derivace), ale její výhodou je větší rychlost konvergence iterační posloupnosti, důležité je však správná volba intervalu (a,b), jinak se dokonce může stát, že posloupnost k hledané limitě x^* nebude vůbec konvergovat.

Mohli bychom zmínit i např. *metodu prosté iterace*, z hlediska studentů středních škol nám však připadají nejschůdnější metoda půlení intervalu a metoda regula falsi. Program *GNU Octave* má k tomuto účelu i nadefinovanou proceduru `fsolve` využívající modifikovanou Newtonovu iterační metodu. Podrobnější informace včetně odvození potřebných vztahů lze najít např. v [3, 9, 14, 19]. Zde si ukažme jejich použití na konkrétní úloze převzaté z [21], kde je použit program *FAMULUS*.

9.1.1 Úloha: kometa Hale-Bopp

Kometa Hale-Bopp objevená 23. července 1995 prolétla 1. dubna 1997 periheliem ve vzdálenosti $r_\mathrm{p} = 0{,}9141\,\mathrm{AU}$ od Slunce. Hlavní poloosa její trajektorie měří $a = 187{,}8\,\mathrm{AU}$. Naším úkolem je určit, v jaké vzdálenosti od Slunce se nacházela v době objevení a jakou rychlostí se přitom pohybovala?

Určování poloh astronomických těles v konkrétním čase patří v astronomii k základním úlohám. Pro pohyb po eliptické trajektorii lze odvodit *Keplerovu rovnici* (viz. např. [9, 21])

$$E - \frac{e}{a}\sin E - \sqrt{\frac{\varkappa M}{a^3}}\,t = 0, \tag{9.1}$$

kde E tzv. je excentrická anomálie udávaná v radiánech, $\varkappa = 6{,}67{\cdot}10^{-11}\,\mathrm{m^3{\cdot}s^{-2}{\cdot}kg^{-1}}$ gravitační konstanta, $M \approx 1{,}989{\cdot}10^{30}\,\mathrm{kg}$ hmotnost Slunce, $a = 2{,}809\,5{\cdot}10^{13}\,\mathrm{m}$ a e hlavní poloosa a excentricita eliptické trajektorie. Snadno vypočítáme relativní excentricitu $\varepsilon = e/a = (a - r_\mathrm{p})/a = 0{,}995\,13$ a délku vedlejší poloosy trajektorie $b = \sqrt{a^2 - e^2} = 18{,}51\,\mathrm{AU}$. Pro kartézské souřadnice polohy komety s počátkem ve středu Slunce pak platí

$$x = a\cos E - e, \qquad y = \frac{b}{a}\,a\sin E = b\sin E, \tag{9.2}$$

pro vzdálenost komety od Slunce $r = \sqrt{x^2 + y^2}$. Podle zadání od objevení komety do průletu periheliem uplynulo $618\,\mathrm{dnů}$ neboli $618{\cdot}24{\cdot}3\,600 =$

= $53\,395\,200\,\mathrm{s}$. Tento čas a zbývající hodnoty dosadíme do Keplerovy rovnice (9.1) a upravíme ji na tvar

$$E - 0{,}99513 \sin E - 0{,}00413 = 0 \tag{9.3}$$

Jde o transcendentní rovnici, kterou neumíme vyřešit elementárními prostředky (analyticky), Keplerovou rovnicí (9.3) je excentrická anomálie určena implicitně. Jedná se o rovnici typu $f(t,E) = 0$ a její řešení pro zvolené t musíme provést některou z přibližných numerických metod. Pro $t \in \langle 0,T \rangle$, kde T odpovídá periodě oběhu komety, je výraz $\sqrt{\varkappa M/a^3}\, t$ z intervalu $\langle 0,2\pi \rangle$ a také E je v intervalu $\langle 0,2\pi \rangle$. Jakmile známe E, vypočítáme souřadnice bodů trajektorie v čase t podle vztahů (9.2).

Jak již bylo řešeno, k získání numerického řešení se nabízí několik způsobů.

Řešení pomocí předdefinované funkce `fsolve`

Nejprve musíme zadefinovat vlastní funkci (označme ji `f`) odpovídající levé straně rovnice (9.3).[2] Definice bude ohraničena příkazy `function` a `endfunction`. Poté do argumentu příkazu `fsolve` uvedeme název funkce a počáteční bod, tj. hodnotu, od níž má algoritmus řešení hledat; v našem případě 0. Celý výpis použitých kroků pak vypadá následujícím způsobem

```
1 octave:1> function y=f(E)
2 > y=E-0.99513*sin(E)-0.0041307;
3 > endfunction
4 octave:2> E=fsolve("f",0)
```

Předcházející řádky lze samozřejmě uložit, např. do souboru `halebp.m` a posloupnost příkazů volat příkazem `halebp`. Volaná funkce `fsolve` vypíše hledané řešení

```
5 E = 0.25891
```

Hledaná hodnota je $E = 0{,}25891\,\mathrm{rad}$. Uvedený postup je sice velmi jednoduchý (zadání zabírá pouhé 4 řádky), ale neumožňuje nám zcela nahlédnout do algoritmu, jakým *GNU Octave* k tomuto výsledku došel a mít pod kontrolou všechny parametry.[3] Chceme-li použít konkrétní metodu, musíme zadat přesný postup výpočtu. Máme-li zájem o výpis většího počtu desetinných míst, zapneme

[2] Na rozdíl od *Matlabu*® nemusí být definice uživatelské funkce uložena v samostatném souboru.

[3] K dispozici je ještě analogická funkce `fzero`, která má více parametrů a umožňuje zadat i požadovanou přesnost.

`format long`, `fsolve` pak vrací `E = 0.258914801661345`. Abychom nemuseli vypisovat celý algoritmus pokaždé znova, je vhodné si na každou metodu vytvořit samostatný soubor. Připomeňme, že ho stačí napsat v libovolném textovém editoru a uložit s příponou `*.m` do adresáře v níž, budeme s *GNU Octave* pracovat (přepneme se do něj pomocí příkazu `cd`). Celý program pak voláme jménem souboru bez přípony.

Řešení metodou půlení intervalu

Možná podoba programu uložená ve zvláštním souboru (např. `halboppi.m`) je následující

```
# --- Metoda puleni intervalu pro reseni Keplerovy rovnice
# --- Vymazeme hodnoty promennych z~predchazejicich vypoctu
clear;
# --- a obsah  terminalu
clc;
# --- zadani konstant
a=187.8*1.4959787e11;
rp=0.9141*1.4959787e11;
kappa=6.672e-11;
M=1.9891e30;
t=618*24*3600;
# --- Definice funkce
function y=f(E)
   y=E-0.99513*sin(E)-0.00413;
endfunction
# --- Pozadovana presnost
presnost=1e-6;
# --- Zaciname merit dobu vypoctu
t0=clock();
# --- Krajni body a stred intervalu
x1=0;           # levy okraj
x2=pi;          # pravy okraj
f1=f(x1);       # leva funkcni hodnota
f2=f(x2);       # prava funkcni hodnota
k=1;            # krok vypoctu
xk=0.5*(x1+x2); # stred intervalu
fk=f(xk);       # hodnota funkce ve stredu intervalu
# --- Formatovany vypis hodnot
printf("k=%f xk=%f f(xk)=%.20f\n", [k,xk,fk]);
# --- Cyklus, ktery se opakuje do dosazeni pozadovane presnosti
while abs(x1-xk)>2*presnost
if (f1*fk<0)
  x2=xk;
else
```

```
35   x1=xk;
36 endif
37 k=k+1;
38 xk=0.5*(x1+x2);
39 fk=f(xk);
40 # --- Formatovany vypis hodnot
41 printf("k=%f xk=%f f(xk)=%.20f\n", [k,xk,fk]);
42 endwhile
43 # --- Ukoncime mereni trvani vypoctu a vypiseme cas potrebny na beh cyklu
44 casvypoctu=etime(clock(),t0)
45 # --- Vypocteme polohu a rychlost v~pozadovanem okamziku
46 r=sqrt((a*cos(xk)-(a-rp))^2+(sqrt(a^2-(a-rp)^2)*sin(xk))^2)/1.5e11
47 v=sqrt(kappa*M*(2/(r*1.5e11)-1/a))/1e3
```

Protože jednotlivé části jsou okomentovány a mělo by být zřejmé, jaký účel jednotlivé části mají, doplňme jen pár poznámek. Komentáře, tj. řádky, které nebude *GNU Octave* provádět a interpretovat, mohou být uvozeny buď znakem `#`, nebo `%` (z důvodu kompatibility s *Matlabem*®). Psaní komentářů lze podobně jako u jiných programovacích jazyků vřele doporučit. Činí kód přehlednějším a srozumitelnějším jak pro jiné uživatele, tak pro samotného autora, pokud se k programu vrací po delší době. Otázkou volby je použití diakritiky v komentářích, zde se jí z důvodu přenositelnosti mezi různými operačními systémy s různým kódováním češtiny vyhýbáme.

Zadáváme-li vstupní hodnoty (řádky 7–11) většinou nepožadujeme jejich vypsání na obrazovku, proto jsou ukončeny znakem `;`. Naopak poslední řádky tento ukončovací znak nemají, neboť chceme vypočtené hodnoty

$$\begin{aligned} r &= \sqrt{\left[a\cos x_\mathrm{k} - (a - r_\mathrm{p})\right]^2 + \left[\sqrt{a^2 - (a - r_\mathrm{p})^2}\,\sin x_\mathrm{k}\right]^2} \\ v &= \sqrt{\varkappa M\left(\frac{2}{r} - \frac{1}{a}\right)} \end{aligned}$$

vypsat, přitom je přímo převádíme na astronomické jednotky AU resp. na $\mathrm{km{\cdot}s^{-1}}$. Při běhu cyklu chceme, aby program postupně vypisoval jak číslo kroku k iterační posloupnosti, střed intervalu v tomto kroku x_k (momentální aproximaci hledaného řešení) a hodnotu funkce v tomto bodě $f(x_\mathrm{k})$ (posledně jmenovanou na 20 desetinných míst) příkazem `printf`.

V programu se setkáváme s třemi novými prvky, o nichž ještě nebyla v textu řeč. Jde jednak o měření času, který počítač spotřebuje na provedení výpočtu nebo některé jeho části. Slouží k tomu příkaz `etime`, který spočte rozdíl dvou časových údajů zadaných jako jeho argumenty. Systémový čas pak vrací funkce `clock()` jako vektor se složkami odpovídajícími roku, měsíci, dni, hodině, minutám a

sekundám. Čas potřebný na provedení výpočtu je orientační, neboť závisí nejen na hardwarové konfiguraci (procesor, operační paměť apod.), ale i na procesech běžících momentálně v počítači na pozadí. Proto se údaj i u dvou bezprostředně následujících stejných výpočtech může o něco lišit (až v řádu sekund).

Dalším novým prvkem je rozhodovací podmínka `if` (32.–36. řádek) větvící výpočet na základě vyhodnocení nějakého logického výrazu, větvení je uzavřeno výrazem `endif`, obě možnosti odděluje `else`. První část se provede, pokud je podmínka větvení splněna, druhá pokud splněna není. V našem případě jde o rozhodutí, v kterém z intervalů $\langle x_1,x_\mathrm{k}\rangle$ nebo $\langle x_\mathrm{k},x_2\rangle$ bude výpočet pokračovat. Vybrat musíme ten, v němž leží nulový bod; v něm musí mít funkční hodnoty opačná znaménka, tj. jejich součin bude záporný. Leží-li v $\langle x_1,x_\mathrm{k}\rangle$ bude v dalším kroku x_k horní, v opačném případě dolní hranicí studovaného intervalu.

Posledním prvkem, o němž se zmíníme, je cyklus typu `while` (31.–42. řádek) probíhající tak dlouho, dokud je splněna zadaná podmínka, a ukončený výrazem `endwhile`. V našem případě jde o dosažení přesnosti výsledku – chceme, aby se poslední nalezené x_k od skutečného řešení lišilo méně než o zvolené ε. Protože při posledním kroku najdeme ještě střed intervalu, stačí testovat, zda je x_k od jednoho z krajních bodů (zde x_1) vzdáleno více než 2ε.

Po zadání příkazu `halboppi` se na obrazovce objeví výpis

```
k=1.000000 xk=1.570796 f(xk)=0.57153632679489663193
k=2.000000 xk=0.785398 f(xk)=0.07760499223527936308
k=3.000000 xk=0.392699 f(xk)=0.00774931764925237444
k=4.000000 xk=0.196350 f(xk)=-0.00192069129854763841
k=5.000000 xk=0.294524 f(xk)=0.00152332039781002439
k=6.000000 xk=0.245437 f(xk)=-0.00048994036543237827
k=7.000000 xk=0.269981 f(xk)=0.00043675232187719883
k=8.000000 xk=0.257709 f(xk)=-0.00004569147745284684
k=9.000000 xk=0.263845 f(xk)=0.00019064495530215532
k=10.000000 xk=0.260777 f(xk)=0.00007126924592972851
k=11.000000 xk=0.259243 f(xk)=0.00001248874645650745
k=12.000000 xk=0.258476 f(xk)=-0.00001667618293280625
k=13.000000 xk=0.258859 f(xk)=-0.00000211244972662082
k=14.000000 xk=0.259051 f(xk)=0.00000518346210189903
k=15.000000 xk=0.258955 f(xk)=0.00000153433504570567
k=16.000000 xk=0.258907 f(xk)=-0.00000028935007293472
k=17.000000 xk=0.258931 f(xk)=0.00000062241929662567
k=18.000000 xk=0.258919 f(xk)=0.00000016651631525900
k=19.000000 xk=0.258913 f(xk)=-0.00000006142145290121
k=20.000000 xk=0.258916 f(xk)=0.00000005254628767694
k=21.000000 xk=0.258915 f(xk)=-0.00000000443786849456
casvypoctu = 0.012676
r = 7.1242
v = 15.610
```

Vidíme, že funkční hodnota $f(x_\mathrm{k})$ v nalezených bodech se ve 3. sloupci stále více blíží 0, hledanou číselnou aproximací řešení je poslední hodnota $E = x_k = $ $= (0{,}258\,915 \pm 10^{-6})\,\mathrm{rad}$. K dosažení požadované přesnosti jsme potřebovali $k = 21$ kroků.

Řešení metodou tětiv

Výpis programu pro řešení Keplerovy rovnice metodou tětiv může být třeba v souboru `halbopte.m` následující

```
# --- Metoda tetiv pro reseni Keplerovy rovnice
# --- Vymazeme hodnoty promennych z~predchazejicich vypoctu
clear;
# --- a obsah terminalu
clc;
# --- Definice funkce
function y=f(E)
   y=E-0.99513*sin(E)-0.00413;
endfunction
# --- Pozadovana presnost
presnost=1e-8;
# --- Zaciname merit dobu vypoctu
t0=clock();
# --- Krajní body
a=0;            # levy okraj
b=pi;           # pravy okraj
fa=f(a);        # leva funkcni hodnota
fb=f(b);        # prava funkcni hodnota
k=1;            # krok vypoctu
x(k)=(a*fb-b*fa)/(fb-fa); # aproximace řešení
fk=f(x(k));       # hodnota funkce ve stredu intervalu
# --- Formátovaný výstup hodnot
printf("k=%f x(k)=%f f(xk)=%.20f\n", [k,x(k),fk]);
# --- Cyklus
while abs(f(x(k)))>presnost
k=k+1;
if (f(a)*f(x(k-1))<0)
  x(k)=(a*f(x(k-1))-x(k-1)*fa)/(f(x(k-1))-fa);
else
  x(k)=(x(k-1)*fb-b*f(x(k-1)))/(fb-f(x(k-1)));
endif
fk=f(x(k));
# --- Formátovaný výstup hodnot
printf("k=%f xk=%f f(xk)=%.20f\n", [k,x(k),fk]);
endwhile
casvypoctu=etime(clock(),t0)
```

Zde v testovací podmínce cyklu `while` na 25. řádku testujeme, zda se hodnota funkce v hledaném bodě $f(x_\mathrm{k})$ liší od 0 více než je požadovaná přesnost. Vidíme, že od předcházející metody se program skutečně liší jen konstrukcí bodu x_k.

Příkaz `halbopte` pak vypíše

```
k=1.000000 x(k)=0.004130 f(xk)=-0.00410987521635498669
k=2.000000 xk=0.008234 f(xk)=-0.00408980538600391998
...
k=416.000000 xk=0.258915 f(xk)=-0.00000001033584787723
k=417.000000 xk=0.258915 f(xk)=-0.00000000997460833894
casvypoctu = 0.26952
```

Pro danou rovnici je metoda regula falsi evidentně méně efektivní, neboť při stejné přesnosti potřebujeme přes 400 kroků.

Vidíme, že všemi způsoby dostáváme $E \approx (0{,}258\,915 \pm 10^{-6})\,\mathrm{rad}$. Dosazením do vztahů (9.2) dostaneme vzdálenost komety od Slunce v době jejího objevení e vzdálenost komety od Slunce $r = 7{,}12\,\mathrm{AU}$ a rychlost v tomto okamžiku $v = 15{,}6\,\mathrm{km{\cdot}s^{-1}}$.

Zájemce o další středoškolské úlohy řešené pomocí *GNU Octave* lze odkázat na práci [9], podobné úlohy řešené programem *FAMULUS* najde v textu [19].

9.2 Numerické řešení diferenciálních rovnic

Numerické řešení soustavy diferenciálních rovnic je jádrem dynamického modelování. V programu *GNU Octave* máme na výběr, zda sestavíme cyklus odpovídající některé z Eulerových metod či metod Rungova-Kuttova typu nebo zda využijeme předdefinované funkce `lsode`.[4] Její použití ilustrujeme na dvou konkrétních příkladech.

[4] Jde také o algoritmus patřící mezi Rungovy-Kuttovy metody nazvané původně po německých matematicích Carlu Davidu Tolmé Rungovi (1856–1927) a Martinu Wilhelmu Kuttovi (1867-1944). Na nějaké variantě Rungových-Kutových metod jsou vesměs založeny procedury pro řešení obyčejných diferenciálních rovnic ve většině matematických programů *FAMULUS* nevyjímaje. Pro uživatele znalé Matlabu jsou v *GNU Octave* navíc funkce `ode23`, `ode45`, `ode78` a `rk2fixed`, `rk4fixed`, `rk8fixed` pro Rungovy-Kutovy metody naznačeného řádu s proměnným resp. fixovaným (pevným) krokem, jejichž autorem je Marc Compere. Zde se přidržíme pouze funkce `lsode`. Název samotný je akronymem k „Livermore Solver for ODEs“ a vychází z fortranovského balíčku Alana Hindmarshe (`http://www.netlib.org/odepack/`). V podstatě jde opět o variantu Rungovy-Kuttovy metody 4. řádu s tím, že konkrétní podmetodu (Adamsovu apod.) lze volit parametrem funkce `lsode_options`.

9.2.1 Van der Polův oscilátor

Van der Polův oscilátor sehrál významnou úlohu ve vývoji nelineární dynamiky. Z matematického hlediska je popsán van der Polovou rovnicí

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + \mu \left(y^2 - 1\right) \frac{\mathrm{d}y}{\mathrm{d}t} + y = 0, \tag{9.4}$$

kde $\mu \geqq 0$ je konstantní parametr. Rovnice tohoto typu se objevila při studiu nelineárních elektrických obvodů v prvních radiopřijímačích v roce 1926 holandským inženýrem B. van der Polem, podobnou rovnicí se zabýval už lord Rayleigh okolo roku 1880 v souvislosti s nelineárními vibracemi [7, 18]. Rovnice (9.4) se od rovnice harmonického oscilátoru liší prostředním členem odpovídajícím nelineárnímu tlumení $\mu \left(y^2 - 1\right) \mathrm{d}y/\mathrm{d}t$. Pro $|y| > 1$ vede ke skutečnému kladnému tlumení, naopak pro $|y| < 1$ představuje negativní tlumení (buzení) systému. Jinými slovy, tlumí velké a budí malé výchylky. Lze odhadnout, že systém může přejít do stavu, kdy se oba procesy vyrovnají a z obecnějších vět lze ukázat [18], že pro každé $\mu > 0$ má rovnice jednoznačně určený limitní periodický režim. Jak ukážeme, s rostoucím μ se tyto limitní kmity víc a více liší od kmitů harmonických.

Pro použití funkce `lsode` rozdělíme diferenciální rovnici 2. řádu na soustavu rovnic 1. řádu

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= x_2, \\ \frac{\mathrm{d}^2 y}{\mathrm{d}t^2} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right) &= -\mu \left(y^2 - 1\right) \frac{\mathrm{d}y}{\mathrm{d}t} - y = -\mu \left(x_1^2 - 1\right) x_2 - x_1. \end{aligned}$$

Takto zapsaná nám umožňuje zavést dvě matice (vektory). Jedna, již označíme `x` obsahuje v prvním řádku `x(1)` hodnoty y a ve druhém `x(2)` hodnoty 1. derivace $\mathrm{d}y/\mathrm{d}t$. Druhá matice `xdot` se skládá z derivací v matice první, tj. z první a druhé derivace y. Soustavu proto lze přepsat ve tvaru

$$\begin{aligned} \mathrm{xdot}(1) &= x(2), \\ \mathrm{xdot}(2) &= -\mu \left(x(1)^2 - 1\right) x(2) - x(1). \end{aligned}$$

Soubor `vanderpol.m` pak může mít následující podobu

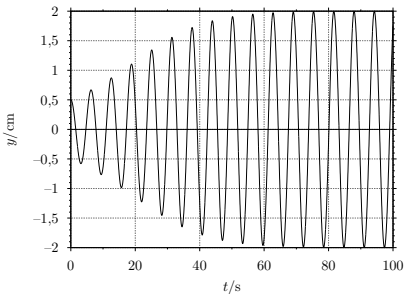
```
1 # --- van der Poluv oscillator
2 # --- Vymazeme hodnoty promennych z~predchazejicich vypoctu
3 clear
4 # --- a obsah terminalu
5 clc;
```

```
6  global mu
7  mu=.1;
8  # --- pocatecni podminky
9  x0=[0.5; 0];
10 # --- definice diferencialni rovnice
11 function xdot = vdpol(x,t)
12 global mu;
13   xdot = [x(2); -mu*(x(1).^2-1).*x(2)-x(1)];
14 endfunction
15 # --- oblast integrace
16 t=linspace(0,100,5000);
17 # --- reseni soustavy rovnic
18 vdp=lsode( "vdpol",x0, t);
19 % vykresleni zavislosti
20 plot(t,vdp(:,1))
21 % vykresleni rychlosti v zavislosti na poloze ("fazovy prostor")
22 plot(vdp(:,1),vdp(:,2))
```

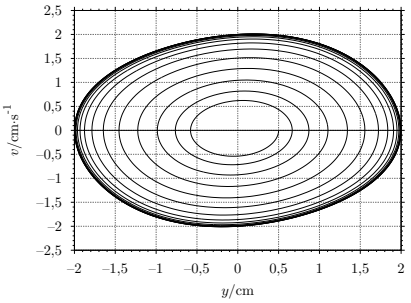
Všimněme si, že pokud přiřazujeme hodnotu nějaké konstantě, kterou potom využíváme v definici funkce (v našem případě jde o parametr $\mu = 0{,}1$), musíme ji deklarovat jako globální konstantu příkazem `global` na počátku před přiřazením hodnoty (6. řádek) i v těle funkce (12. řádek). Pokud se vyskytuje pouze v definici funkce, mohli bychom se tomu vyhnout tak, že bychom pouze v definici funkce nahradili 12. řádek za `mu=0.1`.

Vzhledem k tomu, že jde o rovnici 2. řádu (nebo po provedené úpravě soustavu dvou rovnic 1. řádu), zadáváme na 9. řádku počáteční podmínky jako vektor `x0`, jehož složky odpovídají volbě $y_0 = 0{,}5$ a $\mathrm{d}y/\mathrm{d}t|_0 = 0$. Časový interval, v němž bude provedena integrace pak zadáváme na 16. řádku pomocí nám již známého příkazu `linspace`; zde konkrétně $t \in \langle 0{,}100 \rangle$ s a interval je rozdělen na 5000 bodů. Tímto dělením určujeme integrační krok a tím i přesnost výsledku popř. hladkost vykreslených křivek – pokud je dělení jemné, trvá výpočet déle, pokud je ale příliš hrubé, namísto křivek dostaneme lomené čáry a výsledek nebude dostatečně přesný.

Samotná procedura `lsode`, která provádí vlastní numerickou integraci, vrací matici (zde pojmenovanou `vdp`), jejíž sloupce postupně odpovídají veličinám `x(1)` a `x(2)`, tj. v našem případě hodnotám y a první derivace $\mathrm{d}y/\mathrm{d}t$. Chceme-li potom pouze hodnoty y, vybíráme na 20. řádku pouze první sloupec této matice `vdp(:,1)`, chceme-li druhý, použijeme `vdp(:,2)`. Závislosti $y = y(t)$ a $v = \mathrm{d}y/\mathrm{d}t = v(y)$ jsou pro $\mu = 0{,}1$ a uvedené počáteční podmínky znázorněny na obr. 9.1 a 9.2. Dodejme, že ve výpisu programu nejsou uvedeny příkazy ovlivňující vlastnosti grafu popsané v části 8.5. Vidíme, že průběh kmitů připomíná harmonický pohyb. Z trajektorie pohybu ve fázovém prostoru $y - v$ je zřejmá odlišnost, neboť harmonickým kmitům by odpovídala elipsa, zatímco v našem případě limitní fázová trajektorie připomíná elipsu poněkud deformovanou.

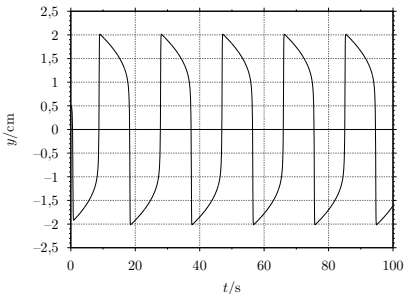


Obr. 9.1: Závislost $y = y(t)$ pro $\mu = 0{,}1$

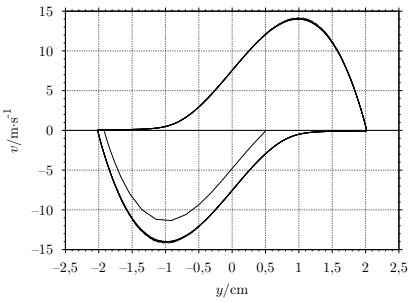


Obr. 9.2: Závislost $v = v(y)$ pro $\mu = 0{,}1$

Pro vyšší hodnoty parametru μ vynikne odlišnost od harmonického pohybu ještě více, což obrážejí obr. 9.3 a 9.4 vykreslené pro stejné počáteční podmínky. Čtenář si může sám ověřit, že limitní kmity oscilátoru závisejí pouze na parametru



Obr. 9.3: Závislost $y = y(t)$ pro $\mu = 10$



Obr. 9.4: Závislost $v = v(y)$ pro $\mu = 10$

μ a nikoli na zvolených počátečních podmínkách (s výjimkou případů, kdy ke kmitům vůbec nedojde jako u triviální volby $y_0 = v_0 = 0$).

9.2.2 Lorenzův atraktor

Na závěr kapitoly si ukažme použití procedury `\lsode` u soustavy 3 obyčejných diferenciálních rovnic. Jako model takové soustavy použijeme známý

Lorenzův atraktor, který se stal jedním ze symbolů nelineární dynamiky a teorie chaosu.

Je nazván po meteorologovi Edwardu N. Lorentzovi, jenž v roce 1963 z hydrodynamických rovnic vynucené konvekce v atmosféře za silně zjednodušených předpokladů odvodil systém tří provázaných nelineárních diferenciálních rovnic. Numerické řešení pak odhalilo typicky chaotické chování – silnou závislost na počátečních podmínkách, která komplikuje předpověď vývoje systému. Není jistě překvapením, že příklad takových systémů poskytuje právě meteorologie a ne náhodou je možné počasí spolehlivě předpovídat jen na několik následujících dnů. Samotnému Lorentzovi se podařilo zpopularizovat podobné jevy pomocí „efektu mávnutí motýlího křídla“.[5]

Lorenzovy rovnice lze zapsat ve tvaru [18]

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma\,(y - x)\,, \tag{9.5a}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x\,(r - z) - y, \tag{9.5b}$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - bz, \tag{9.5c}$$

kde σ, r a b jsou nezáporné hydrodynamické parametry – σ tzv. Prandtlovo číslo, r tzv. Rayleighovo číslo (b pojmenování nemá).

Chceme-li vykreslit známý obrázek odpovídající jednomu z řešení Lorentzových rovnic, může soubor `lorentz.m` obsahovat následující kód

```
# --- Lorentz atractor
# --- Vymazeme hodnoty promennych z~predchazejicich vypoctu
clear
# --- a obsah terminalu
clc;
x=zeros(3,1);
# --- definice diferencialni rovnice
function xdot = loren(x,t)
sigma=10;
r=28;
b=8/3;
xdot = [sigma * (x(2) - x(1));
        x(1) * (r - x(3))-x(2);
        x(1) * x(2) - b * x(3)];
```

[5] Citlivost na změnu počátečních podmínek je ilustrována tvrzeními typu, že mávnutí motýlích křídel v Brazílii může rozhodnout o tom, zda texaské pláně budou zasaženy tornádem či nikoli. Dodejme, že Lorenzova práce zůstala až do počátku 70.-ých let 20. století bez větší odezvy [7, 18].

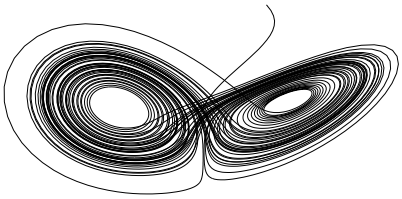
```
15 endfunction
16 # --- oblast integrace
17 t=(0:0.01:60)'
18 # --- reseni soustavy rovnic
19 lr = lsode("loren", [3;14;50], t);
20 # --- vykresleni reseni - trojrozmerna krivka
21 __gnuplot_set__ noborder
22 __gnuplot_set__ noxtics
23 __gnuplot_set__ noytics
24 __gnuplot_set__ noztics
25 __gnuplot_set__ nokey
26 plot3(lr(:,1),lr(:,2),lr(:,3));
```

Upozorněme opět na dosud nezmíněné příkazy a odlišnosti od programu pro van der Polův oscilátor. Na první pohled vidíme, že zde chybí globální deklarace konstant příkazem `global`, jejich hodnoty jsou uvedeny přímo v definici funkce definující soustavu Lorentzových diferenciálních rovnic na 9.–11. řádku. Počáteční podmínky tentokrát nevypisujeme na zvláštní řádek, ale zapisujeme přímo jako druhý argument procedury `lsode`. V tomto případě jde o počáteční podmínky $x_0 = 3$, $y_0 = 14$ a $z_0 = 50$ (matice `[3;14;50]`). Oblast integrace je dána na 17. řádku intervalem $t \in \langle 0{,}60 \rangle$ s, hodnoty v něm měníme s krokem $\Delta t = 0{,}01\,\mathrm{s}$.

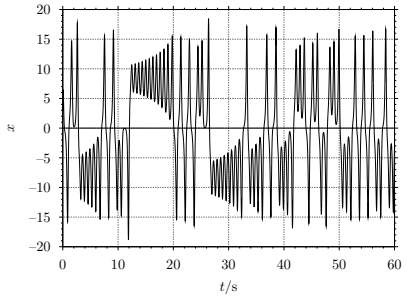
Upozorněme také, že označení funkce (zde `loren`) musí být odlišné od jména celého souboru (bez přípony, zde `lorentz`). Protože řešíme soustavu rovnic prvního řádu, není nutné ji nijak přeskupovat, ale lze ji přímo přepsat do definice funkce na 12.–14. řádku. Pracujeme s maticemi `x` o složkách $x_1 = x$, $x_2 = y$, $x_3 = z$ a `xdot` obsahující derivace `x` podle času, tj. $\mathrm{xdot}_1 = \mathrm{d}x/\mathrm{d}t$, $\mathrm{xdot}_2 =$ $= \mathrm{d}y/\mathrm{d}t$ a $\mathrm{xdot}_3 = \mathrm{d}z/\mathrm{d}t$. Procedura `lsode` pak vrací matici (zde označenou `lr`) se sloupci odpovídajícími složkám matice `x`, tj. souřadnicím x, y, z.

Chceme-li vykreslit prostorovou křivku, jejíž body musí být určeny třemi souřadnicemi, použijeme příkaz `plot3`, jehož argumentem jsou právě hodnoty jednotlivých souřadnic. Nepovinnou, ale často užívanou částí je deklarace nulové matice `x` na 6. řádku pomocí funkce `zeros`. Konkrétně `zeros(3,1)` vytvoří matici 3×1 obsahující samé 0, je ekvivalentní zápisu `[0;0;0]`. Konečně příkazy na řádcích 21–25 ovlivňují vzhled obrázku – postupně zakazují vykreslování os, značek na nich a popisu obrázku. Výsledek je pak znázorněn na obr. 9.5a. Trajektorie se protíná jen zdánlivě (jde o důsledek projekce na rovinu) a překlápí se zprava doleva a nazpátek vždy po několika oběžích na každé straně, přičemž počet oběhů na každé straně odpovídá náhodné posloupnosti. V prostoru je však trajektorie lokalizována v poměrně úzké oblasti nazývané podle Ruelleho a Takense od roku 1971 *podivným atraktorem*. S podobnými rovnicemi se setkáme u laserů, dynam a některých typů vodních kol.

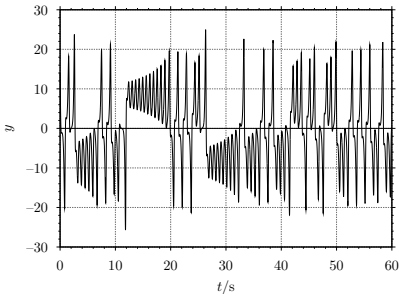
Poučné je i vykreslení závislostí jednotlivých souřadnic na čase příkazy typu `plot(t,lr(:,1))` pro $x = x(t)$ a analogicky pro y i z; vidíme je na obr. 9.5b–d. Řešení odpovídá aperiodickým oscilacím, jež se nikdy zcela přesně neopakují [18]. Čtenáři doporučujeme vyzkoušet i jiné počáteční hodnoty popř. hodnoty hydrodynamických parametrů (např. hodnotu $r = 99{,}96$).



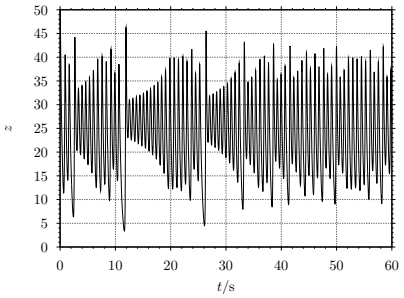
(a) Prostorová křivka



(b) Závislost $x = x(t)$



(c) Závislost $y = y(t)$



(d) Závislost $z = z(t)$

Obr. 9.5: Řešení Lorentzových rovnic pro $\sigma = 10$, $r = 28$, $b = 8/3$ a počáteční podmínky $x_0 = 3$, $y_0 = 14$ a $z_0 = 50$

Část 10

Příklady jednoduchých dynamických modelů

V poslední kapitole stručně uvedeme příklady skriptů pro *GNU Octave*, jimiž lze řešit diferenciální rovnice popisující konkrétní fyzikální systémy a jevy. Převážná většina z nich byla navíc teoreticky popsána v první části textu a modelována pomocí programu *Coach*. Nebudeme proto opakovat odvození či sestavení pohybových rovnic, zaměříme se za na naprogramování v *GNU Octave* a prezentaci grafických výsledků. Význam téměř všech použitých příkazů byl popsán v předcházejících kapitolách (k rychlé orientaci lze využít i rejstřík použitých příkazů na s. 158). Pro přehlednost nebudeme uvádět příkazy definující pouze vzhled obrázku, popisky os apod., jež byly shrnuty v části 8.5.

Modely jsou rozděleny podle použité integrační metody, tj. Eulerovy, Rungovy-Kuttovy popř. pomocí předdefinované funkce `lsode`. Dodejme, že další zajímavé úlohy řešené pomocí *GNU Octave* najde čtenář v diplomové práci [9], řešení pomocí programu *FAMULUS* v [12, 19].

10.1 Modely využívající Eulerovu metodu

10.1.1 Balistická křivka

Modelujeme šikmý vrh v odporujícím prostředí popsaný v části 1.4, odpovídající program pro *FAMULUS* najde čtenář v [24]. Uvažujeme těleso tvaru koule o hmotnosti m a poloměru R vržené počáteční rychlostí v_0 pod úhlem α k horizontu. Při pohybu na něj působí konstantní tíhová síla $m\boldsymbol{g}$ svisle dolů a proti pohybu síla odporu prostředí úměrná druhé mocnině okamžité rychlosti podle Newtonova vztahu $\boldsymbol{F}_\mathrm{o} = -kv\boldsymbol{v} = -C\varrho Sv\boldsymbol{v}/2$, kde C je součinitel odporu závisející na tvaru tělesa, ϱ hustota prostředí (uvažujeme vzduch) a $S = \pi R^2$ plocha příčného průřezu střely. Zvolíme-li standardně osu x vodorovně a osu y svisle vzhůru, musíme řešit soustavu diferenciálních rovnic

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t} = -kvv_x \tag{10.1a}$$

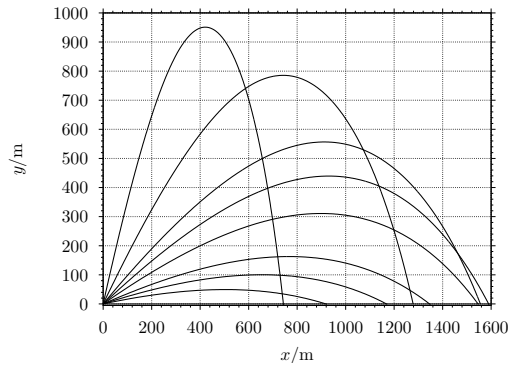
$$m\frac{\mathrm{d}^2y}{\mathrm{d}t} = -mg - kvv_y, \tag{10.1b}$$

kde parametr k postupně vypočteme uvedeným způsobem. Výsledné trajektorie – balistické křivky – pro několik různých elevačních úhlů lze vykreslit např. pomocí souboru `baliste.m` s obsahem

```
# --- numericke reseni - balisticka krivka
# --- pro nekolik elevacnich uhlu
clear;
clc;
# --- definice pocatecnich podminek (v zakladnich jednotkach SI)
t0=0;
x0=0;
y0=0;
v0=200         # pocatecni rychlost
eluhel=[10 15 20 30 37.9 45 60 75]     % elevacni uhly ve stupnich
alpha=eluhel*pi/180; # prevedeni na radiany
# --- krok integrace
dt = 0.05;
# --- parametry prostredi
R=0.04;         # polomer strely
m=2.5;          # hmotnost strely
ro=1.3;         # hustota vzduchu
C=0.48;         # koeficient odporu
g=9.81;         # tihove zrychleni
S=pi*R^2;
k=C*ro*S/2;
hold on # pro vice obrazku do jednoho grafu
# --- cyklus pro hodnoty elevacnich uhlu
for j=1:columns(alpha)
clear t
clear x
clear y
vx=v0*cos(alpha(j));
vy=v0*sin(alpha(j));
v=sqrt(vx^2+vy^2);
# --- vlastni numericka integrace - klasicka Eulerova metoda
t(1)=t0;
x(1)=x0;
y(1)=y0;
i=1;
 while (y(i)>=0)  # testujeme dopad na rovinu y=0
   i=i+1;
   x(i)=x(i-1)+vx*dt;
   y(i)=y(i-1)+vy*dt;
   ax=-k*v*vx/m;
   ay=-g-k*v*vy/m;
   vx=vx+ax*dt;
   vy=vy+ay*dt;
   v=sqrt(vx^2+vy^2);
   t(i)=t(i-1)+dt;
 endwhile
# --- vykresleni zavislosti
```

```
48 plot(x,y);
49 endfor
50 hold off
```

Ve výpisu vidíme několik dosud nezmíněných příkazů a postupů. V testovací podmínce cyklu `while` na 36. řádku nyní ověřujeme, zda těleso nedopadlo na rovinu $y = 0$, po dopadu se výpočet ukončí. V rámci jednoho běhu programu vykreslujeme balistické křivky pro několik hodnot elevačních úhlů zadaných ve stupních maticí `eluhel`(10. řádek), která se hned na následujícím řádku přepočítává na matici `alpha` v radiánech.



Obr. 10.1: Balistické křivky pro různé hodnoty elevačních úhlů

K využití všech hodnot pak používáme cyklus `for` mezi 24–49 řádkem ukončený příkazem `endfor`. Podobně jako v jiných programovacích jazycích, mění se v průběhu cyklu celočíselná proměnná (zde j) od jedničky do množství prvků matice `alpha`. Počet sloupců resp. řádků matice zjistíme snadno pomocí funkce `columns` (24. řádek) resp. `rows`. Takto můžeme hodnoty elevačních úhlů libovolně dopisovat nebo umazávat a program si vždy sám zjistí jejich počet.

V každém běhu cyklu, tj, pro každou hodnotu elevačního úhlu se vypočtené `t`, `x` a `y` ukládají do matic toho jména a pro jiný úhel budou jiné. Nechceme-li vytvářet vícerozměrné matice, musíme vždy na konci cyklu hodnoty vykreslit (48. řádek) a na začátku vynulovat příkazem `clear` (25.–27. řádek). Takto `t`, `x` a `y` zůstávají vektory, jejichž prvky voláme jednoduše `x(0)`,`y(j)` apod.

Uvedený postup vyžaduje také použití příkazů `hold on` a `hold off` (22. a 50. řádek). Protože hodnoty x a y jsou počítány postupně vždy v každém cyklu, nelze počkat na konec a použít několik dvojic argumentů funkce `plot` jako v části 8.5.1. Příkaz `hold on` zajistí, že další vykreslování bude provedeno

do již otevřeného obrázku a to tak dlouho, dokud volbu nevypneme voláním `hold off`. Výstup z popsaného programu je znázorněn na obr. 10.1, zvolené počáteční hodnoty jsou uvedeny a okomentovány ve výpisu programu. Vidíme, že za daných podmínek nedosáhneme největší délky vrhu pro elevační úhel $45°$, ale o něco menší (asi $37{,}9°$).

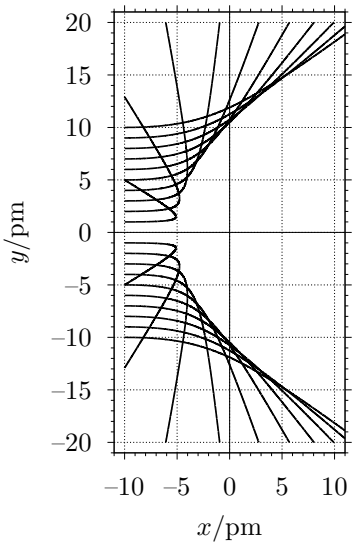
10.1.2 Rutherfordův rozptyl

Ostřelování zlaté fólie o tloušťce několika atomů jádry helia patří k nejznámějším historickým experimentům. Experiment samotný byl poprvé uskutečněn Hansem Geigerem a Ernestem Marsdenem roku 1909 pod vedením Ernesta Rutherforda, který jej v roce 1911 vysvětlil a zformuloval známý planetární model atomu. Fyzikální rozbor problému je nastíněn v části 5.2.

Na nabité α-částice s kladným nábojem $Q_\mathrm{a} = 2e$ a hmotností $m_\mathrm{a} \approx 4u$ působí jádra atomů zlata s nábojem $Q_\mathrm{j} = 79e$ odpudivou coulombovskou silou. Pohybové rovnice mají tvar

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t} = \frac{1}{4\pi\varepsilon_0}\,\frac{Q_\mathrm{a}Q_\mathrm{j}}{r^3}\,x \quad (10.2\mathrm{a})$$

$$m\frac{\mathrm{d}^2y}{\mathrm{d}t} = \frac{1}{4\pi\varepsilon_0}\,\frac{Q_\mathrm{a}Q_\mathrm{j}}{r^3}\,y, \quad (10.2\mathrm{b})$$



Obr. 10.2: Trajektorie částic při Rutherfordově rozptylu

kde $\varepsilon_0 = 8{,}854{\cdot}10^{-12}\,\mathrm{F{\cdot}m^{-1}}$ je permitivita vakua, ve vztazích pro náboje a hmotnost vystupují elementární náboj $e = 1{,}602{\cdot}10^{-19}\,\mathrm{C}$ a hmotnostní atomová jednotka $u = 1{,}66{\cdot}10^{-27}\,\mathrm{kg}$. Trajektoriemi částic budou hyperboly, jejichž příklad je znázorněn na obr. 10.2. Vykresleny byly pomocí souboru `ruther.m` s obsahem

```
1 # --- numericke reseni pro rozptyl na odpuzujícím centru
2 clear;
3 clc;
4 # --- definice pocatecnich podminek a konstant
5 u=1.66e-27;     # hmotnostni atomova jednotka
6 q=1.602e-19;    # elementarni naboj
7 eps0=8.854e-12; # premitivita vakua
```

```
ma=4*u;         # hmotnost alpha-castice
Qa=2*q;         # naboj alpha-castice
Qj=79*q;        # naboj jadra Au
kQQ=1/4/pi/eps0*Qa*Qj; # koeficient v Coulombove zakone
ymax=2e-11;     # maximalni hodnota y pro vykreslovani
t0=0;
x0=-1e-11;
y0=[-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9 10]*1e-12;
vx0=1e6;
vy0=0;
# ------------------------------------------------
dt=1e-20;       # integracni krok
# --- nastavenni os
axis([-11 11 -21 21],'equal');
hold on         # pro vice kreivek v jednom obrazku
# --- cyklus pro nekolik ruznych trajektorii
cas_spusteni = clock();
for j=1:columns(y0)
# --- vlastni numericka integrace - Eulerova metoda
clear x
clear y
t(1)=t0;
x(1)=x0;
y(1)=y0(j);
vx=vx0;
vy=vy0;
i=1;
# --- cyklus while s integraci
 while ((abs(y(i))<=ymax) & (x(i)>=x0))
   i=i+1;
   x(i)=x(i-1)+vx*dt;
   y(i)=y(i-1)+vy*dt;
   r=sqrt(x(i)^2+y(i)^2);
   ax=kQQ/r^3*x(i-1)/ma;
   ay=kQQ/r^3*y(i-1)/ma;
   vx=vx+ax*dt;
   vy=vy+ay*dt;
   t(i)=t(i-1)+dt;
 endwhile
# --- vykresleni zavislosti
plot(x/1e-12,y/1e-12,'1');
endfor
hold off
# --- udaje o case spotrebovanem na vypocet
celkovy_cas = etime(clock(), cas_spusteni);
disp(['Program bezel ',num2str(celkovy_cas),' sekund.']);
```

Opět zadáváme jedním příkazem na 15. řádku více počátečních podmínek maticí, neboť nás zajímají trajektorie s různým momentem hybnosti α-částic, který se při pohybu zachovává a který při počátečním pohybu ve směru osy x závisí na hodnotě y_0. Různé počáteční podmínky opět procházíme v cyklu `for...endfor` mezi 25.–49. řádkem, opakované vynášení do stejného obrázku aktivujeme a deaktivujeme pomocí `hold on...` `hold off`. Vypočítané hodnoty souřadnic vynášíme přímo v pikometrech, proto je v rámci příkazu `plot` na 48. řádku příslušně přeškálujeme.

Z příkazů, s nimiž jsme se v uvedené podobě v textu dosud nesetkali, upozorněme na `axis` (21. řádek), který nemá pouze argument `'equal'` požadující stejné měřítko na obou osách, ale také nastavení rozsahu vykreslovaných hodnot parametrem `[-11 11 -21 21]`, jež je ekvivalentní dvojici `__gnuplot_set__ xrange [-11:11]` a `__gnuplot_set__ yrange [-21:21]` ; omezujeme tím vykreslování na hodnoty $x \in \langle -11{,}11\rangle$, $y \in \langle -21{,}21\rangle$. V podmínce cyklu `while` na 36. řádku pak testujeme, zda hodnota x nesklesla pod x_0 a hodnota y neleží mimo interval $\langle -y_{\max}, y_{\max}\rangle$. Ke zobrazení času spotřebovaného na výpočet je pak na posledním řádku využita funkce `disp`, díky níž se nevypíše pouze hlášení `celkovycas=7.963`, ale celé hlášení `Program bezel 7.963 sekund`. Protože argumentem `disp` mohou být pouze textové řetězce, převádíme číselný údaj o čase na řetězec („string“) pomocí funkce `num2str`.

10.1.3 Pohyb lyžaře s odporem prostředí

Uveďme jednu z úloh zpracovaných pro systém *FAMULUS* v textu [23] se zadáním: Lyžař o hmotnosti $m = 90\,\mathrm{kg}$ získá po určité době jízdy na velmi dlouhém rovném svahu se sklonem $\alpha = 15^\circ$ stálou rychlost $v_\mathrm{m} = 18\,\mathrm{m{\cdot}s^{-1}}$. Součinitel smykového tření mezi lyžemi a sněhem je $f = 0{,}055$, tíhové zrychlení $g = 9{,}81\,\mathrm{m{\cdot}m^{-2}}$. Velikost síly odporu vzduchu je přímo úměrná druhé mocnině rychlosti: $F_\mathrm{o} = Kv^2$. Úkolem je určit velikost koeficientu K a modelovat závislost rychlosti a dráhy lyžaře v závislosti na čase.

Při pohybu lyžaře se uplatní pohybová složka tíhové síly, smykové tření a odpor vzduchu. Výslednice těchto sil má velikost

$$F = mg\,(\sin\alpha - f\cos\alpha) - Kv^2.$$

Na velmi dlouhém svahu dosáhne rychlost lyžaře mezní hodnoty v_m, při které je výslednice všech sil nulová a platí

$$\begin{aligned} mg\,(\sin\alpha - f\cos\alpha) - Kv_\mathrm{m}^2 &= 0 \\ K = \frac{mg\,(\sin\alpha - f\cos\alpha)}{v_\mathrm{m}^2} &= 0{,}5605\,\mathrm{N{\cdot}m^{-2}{\cdot}s^{-2}}. \end{aligned}$$

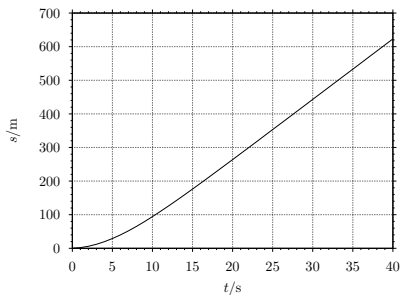
Okamžité zrychlení lyžaře v čase t potom bude

$$a = \frac{F}{m} = g\,(\sin\alpha - f\cos\alpha) - \frac{K}{m}\,v^2.$$

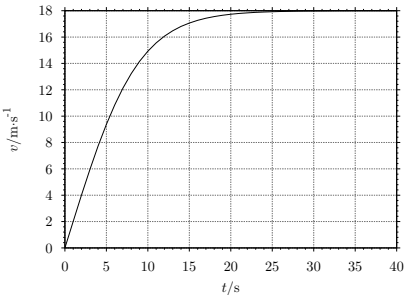
a program pro modelování závislosti rychlosti lyžaře na čase může vypadat následovně:

```
1  # --- numericke reseni rychlosti lyzare na svahu s odporem prostredi
2  clear;
3  clc;
4  # --- definice pocatecních podminek
5  t(1)=0;
6  s(1)=0;
7  v(1)=0;          # pocatecni rychlost
8  m=90;            # hmotnost lyzare
9  uhel=15;         # uhel svahu ve stupnich
10 alpha=uhel*pi/180;
11 f=0.055;         # soucinitel smykoveho treni
12 vm=18;           # mezni rychlost
13 g=9.81;          # tihove zrychleni
14 K=m*g*(sin(alpha)-f*cos(alpha))/vm^2;
15 dt = 1;          # krok integrace
16 # --- vlastní numerická integrace - metoda Eulerova
17    i=1;
18    a=g*(sin(alpha)-f*cos(alpha))-K*v^2/m;
19 # --- vlastni cyklus
20 while (v<=0.9999*vm)  %
21    i=i+1;
22    a=g*(sin(alpha)-f*cos(alpha))-K*v(i-1)^2/m;
23    v(i)=v(i-1)+a*dt;
24    s(i)=s(i-1)+v(i)*dt;
25    t(i)=t(i-1)+dt;
26 endwhile
27 # --- vykreslení závislosti v na t
28 plot(t,v);
29 pause
30 # --- vykreslení závislosti s na t
31 plot(t,s);
```

Výsledný graf je na obrázcích 10.1.3a,b. Připomeňme, že rychlost lyžaře se mezní rychlosti v_{m} sice blíží, ale z matematického hlediska ji přesně nikdy nedosáhne. Chceme-li na vyhnout nekonečnému cyklu `while`, musíme do podmínky zadat rychlost o něco nižší (na 20. řádku volíme konkrétně $0{,}9999\,v_{\mathrm{m}}$).



(a) Závislost $s = s(t)$ pro pohyb lyžaře



(b) Závislost $v = v(t)$ pro pohyb lyžaře

10.2 Modely využívající Rungovy-Kuttovy metody

I když ve většině matematických programů se používají Rungovy-Kuttovy metody 4. řádu, pro naše účely je pro svou jednoduchost zcela postačující metoda 2. řádu s pevným (fixním) krokem, použitá v následujícím příkladu.

10.2.1 Matematické kyvadlo s libovolnou počáteční výchylkou

Matematické kyvadlo patří k nejznámějším modelům. Zde se podobně jako v části 2.5 budeme zabývat pohybem s libovolně velkou počáteční výchylkou. Pohybová rovnice má známý tvar (viz např. [2, 4])

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2} = -\frac{g}{l}\,\sin\varphi,$$

kde g je tíhové zrychlení l délka závěsu. Pro jednoduchost uvažujme sekundové kyvadlo s dobou kmitu $1\,\mathrm{s}$, pro jeho délku musí platit

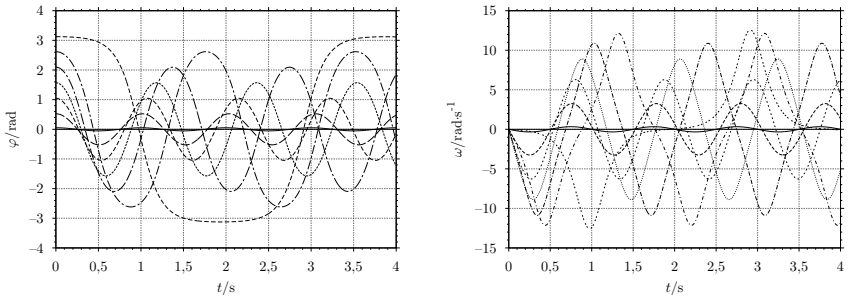
$$l = \frac{g}{4\pi^2} \approx 0{,}25\,\mathrm{m}.$$

Lépe tak uvidíme prodloužení dobu kmitu při větších počátečních výchylkách. Výpis programu `kyvrk2.m` může mít podobu

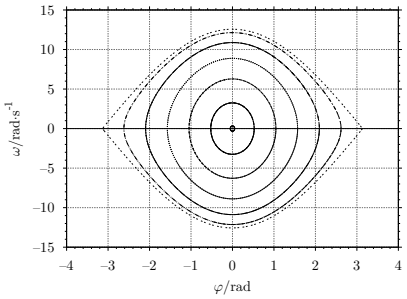
```
1 # --- matematicke kyvadlo s libovolnou pocatecni vychylkou
2 clear;
3 clc;
4 # --- definice počátečních podmínek (v základních jednotkách SI)
```

```
5  phi0=[3 30 60 90 120 150 179]*pi/180;
6  # --- vlastni cyklus
7  x=v=[];
8  hold on
9  for j=1:columns(phi0)
10 clear t;
11 clear ph;
12 t(1)=0;
13 ph(1)=phi0(j);
14 dph(1)=0;      # počateční derivace (úhlová rychlost)
15 mez=4;         # mez integrace - 4 male kmity
16 h = mez/500;   # krok integrace
17 g=9.81;
18 l=g/(4*pi^2);  # delka sekundoveho kyvadla
19 # --- vlastní numerická integrace - RK metoda 2. řádu
20 i=2;
21   while t<mez
22     k1=-g/l*sin(ph(i-1));
23     php=ph(i-1)+dph(i-1)*2*h/3+k1*2*h^2/9;
24     k2=-g/l*sin(php);
25     ph(i)=ph(i-1)+dph(i-1)*h+(k1+k2)*h^2/4;
26     dph(i)=dph(i-1)+(k1+3*k2)*h/4;
27     t(i)=t(i-1)+h;
28     i=i+1;
29   endwhile
30 # --- ukladame vychylku a rychlost do matice pro pozdejsi pouziti...
31 x=[x;ph];
32 v=[v;dph];
33 # --- Vykresleni zavislosti
34 plot(t,ph);
35 endfor
36 hold off
37 # --- vykresleni zavislosti uhlove rychlosti na case
38 for j=1:columns(phi0)
39 hold on
40 plot(t,v(j,:))
41 hold off
42 endfor
43 # --- vykresleni zavislosti uhlove rychlosti na vychylce
44 for j=1:columns(phi0)
45 hold on
46 plot(x(j,:),v(j,:))
47 hold off
48 endfor
```

Podobně jako v části 10.1.1 zde zadáváme několik různých počátečních výchylek v jediné matici `phi0` (5. řádek), počet jejích prvků (tj. zadaných hodnot)



(c) $\varphi = \varphi(t)$ (d) $\omega = \omega(t)$



(e) $\omega = \omega(\varphi)$

Obr. 10.3: Kmity matematického kyvadla s počátečními výchylkami $\varphi_0 = 3°$, $30°$, $60°$, $90°$, $120°$, $150°$ a $179°$

opět zjišťujme pomocí funkce `columns`. Podobně jako v části 10.1.1 užíváme i cyklu `for`...`endfor` a opakovaného vykreslování do stejného obrázku pomocí `hold on`... `hold off`. Chceme-li vykreslit kromě závislosti výchylky na čase $\varphi = \varphi(t)$ také závislosti úhlové frekvence na čase $\omega = \omega(t)$ a úhlové frekvence na výchylce $\omega = \mathrm{d}\varphi/\mathrm{d}t = \omega(\varphi)$ (tj. znázornění kmitů ve „fázovém prostoru“), musíme cyklus s vykreslováním opakovat třikrát (9.–35. řádek včetně výpočtu, 38.–42. řádek a 44.–48. řádek pro vykreslení). Abychom třikrát neopakovali celou integraci, ukládáme hodnoty postupně do matic `x` a `v`, které již nejsou vektory – každý jejich řádek odpovídá řešení pro jednu počáteční výchylku. Proto tyto matice na počátku zavádíme jako prázdné (`x=v=[];` na 7. řádku), abychom k nim mohli přidávat další řádky z hodnot `ph` a `dph` příkazy `x=[x;ph]` resp. `v=[v;dph]`

(31.–32. řádek). Integrační krok na 16. řádku je pak volen tak, aby uvažovaný časový interval $\langle 0{,}4\rangle$ byl rozdělen na 500 dílků.

Dodejme, že není možné zadat limitní hodnotu počáteční výchylky $\varphi_0 = 180^\circ$, neboť v tomto případě při nulové počáteční rychlosti těleso začne padat volným pádem a rovnice matematického kyvadla pro něj nebude platit. Všechny tři zmíněné závislosti jsou postupně vykresleny na obrázcích 10.3a–c. Vidíme, že s rostoucí počáteční výchylkou se pohyb více a více liší od harmonického, pro $\varphi_0 = 179^\circ$ je doba kmitu téměř $4\,\mathrm{s}$, tj. asi $4\times$ větší než pro malé kmity kyvadla.

10.2.2 Pohyb družice v gravitačním poli Země a Měsíce

Problém je po fyzikální stránce popsán v části 1.8, jde o příklad tzv. omezeného problému tří těles, kdy předpokládáme, že třetí těleso (v našem případě družice) neovlivňuje svou gravitací zbývající dvě tělesa (Zemi a Měsíc), která se pohybují po kruhových trajektoriích okolo společného hmotného středu. Pokud úlohu navíc řešíme v soustavě, která se otáčí spolu se Zemí a Měsícem, můžeme obě tělesa považovat za nehybná, počátek vztažné soustavy volíme ve středu Země. Úlohu budeme řešit v přiblížení, kdy zanedbáváme setrvačné síly spojené s neinerciálností rotující soustavy oproti gravitačnímu působení. Dodejme, že obecný problém tří těles je úloha, která v důsledku nelineárnosti rovnic nemá obecné analytické řešení a zabývala se jím řada význačných matematiků (Euler, Lagrange, Jacobi, Hill, Poincaré, Levi-Civita, Birkhoff).

V naší úloze použijeme několik astronomických konstant – hmotnost Země $M_\mathrm{Z} = 5{,}983{\cdot}10^{24}\,\mathrm{kg}$, hmotnost Měsíce $M_\mathrm{M} = 7{,}374{\cdot}10^{22}\,\mathrm{kg}$, jež se nacházejí ve vzájemné vzdálenosti $x_\mathrm{M} = 60{,}13\,R_\mathrm{Z}$. Poloměr Země $R_\mathrm{Z} = 6{,}371{\cdot}10^{6}\,\mathrm{m}$ použijeme jako jednotku vzdálenosti, souřadnice polohy družice budeme uvádět v násobcích R_Z. Dále budeme předpokládat, že družice se pohybuje výhradně pod vlivem gravitačního pole s vypnutými motory.

Pro určení gravitačních sil, které působí na kosmickou sondu, musíme určit nejen souřadnice polohy vzhledem k Zemi (x_SZ, y_SZ), ale i relativní souřadnice sondy vzhledem k Měsíci $x_\mathrm{SM} = x_\mathrm{SZ} - x_\mathrm{M}$, $y_\mathrm{SM} = y_\mathrm{SZ} - y_\mathrm{M}$, kde x_M, y_M jsou souřadnice Měsíce ve vztažné soustavě spojené se Zemí. Měsíc umístíme na osu x naší soustavy, takže klademe $x_\mathrm{M} = 60{,}13 R_\mathrm{Z}$, $y_\mathrm{M} = 0$. Současně je také zřejmé, že $y_\mathrm{SM} = y_\mathrm{SZ}$. Pro vzdálenost družice od Země platí $r = \sqrt{x_\mathrm{SZ}^2 + y_\mathrm{SZ}^2}$, pro její vzdálenost od Měsíce $r_\mathrm{M} = \sqrt{x_\mathrm{SM}^2 + y_\mathrm{SM}^2} = \sqrt{\left(x_\mathrm{SZ} - x_\mathrm{M}\right)^2 + y_\mathrm{SZ}^2}$.

Složky zrychlení sondy budou výslednicí gravitačního působení Země a Měsíce, můžeme pro ně proto psát

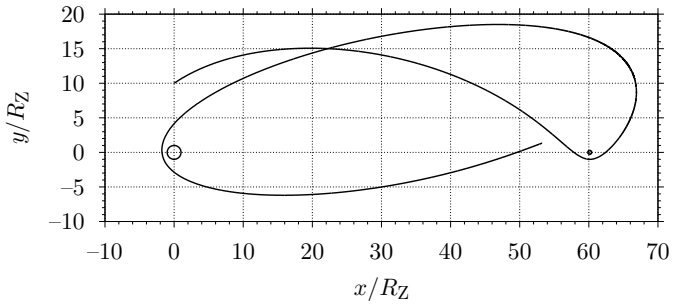
$$\begin{aligned} a_x &= -\varkappa\frac{x_{\mathrm{SZ}}M_{\mathrm{Z}}}{r^3} - \varkappa\frac{x_{\mathrm{SM}}M_{\mathrm{M}}}{r_{\mathrm{M}}^3} = -\varkappa\frac{x_{\mathrm{SZ}}M_{\mathrm{Z}}}{r^3} - \varkappa\frac{(x_{\mathrm{SZ}}-x_{\mathrm{M}})\,M_{\mathrm{M}}}{r_{\mathrm{M}}^3}, \\ a_y &= -\varkappa\frac{y_{\mathrm{SZ}}M_{\mathrm{Z}}}{r^3} - \varkappa\frac{y_{\mathrm{SM}}M_{\mathrm{M}}}{r_{\mathrm{M}}^3}. = -\varkappa\frac{y_{\mathrm{SZ}}M_{\mathrm{Z}}}{r^3} - \varkappa\frac{y_{\mathrm{SZ}}M_{\mathrm{M}}}{r_{\mathrm{M}}^3}. \end{aligned}$$

Program `sondark2` pak může vypadat následovně

```
# --- numericke reseni pohybu druzice
clear;
clc;
# --- pocatecni podminky a konstanty ---
RZ=6.378e6;     % polomer Zeme
t(1)=0;
xsZ(1)=0;
ysZ(1)=10*RZ;
vx=2.7e3;       % x-ova slozka startovni rychlosti
vy=1.84e3;      % y-ova slozka startovni rychlosti
xM=60.13*RZ;
yM=0;
mez= 1.2e6;     % mez integrace
h = mez/6000;   % krok integrace
# --- parametry prostredi, konstanty ---
k=6.67259e-11;  % gravitacni konstanta
MZ=5.98e24;     % hmotnost Zeme
MM=7.374e22;    % hmotnost Mesice-
# --- vlastni numericka integrace - RG metoda 2. radu
i=2;
while (t<mez & xsZ(i-1)^2+ysZ(i-1)^2>RZ^2)
   k1x=-k*xsZ(i-1)*MZ/(xsZ(i-1)^2+ysZ(i-1)^2)^1.5 \
    -k*MM*(xsZ(i-1)-xM)/((xsZ(i-1)-xM)^2+ysZ(i-1)^2)^1.5;
   k1y=-k*ysZ(i-1)*MZ/(xsZ(i-1)^2+ysZ(i-1)^2)^1.5 \
    -k*MM*ysZ(i-1)/((xsZ(i-1)-xM)^2+ysZ(i-1)^2)^1.5;
   xp=xsZ(i-1)+vx*2*h/3+k1x*2*h^2/9;
   yp=ysZ(i-1)+vy*2*h/3+k1y*2*h^2/9;
   k2x=-k*xp*MZ/(xp^2+yp^2)^1.5-k*MM*(xp-xM)/((xp-xM)^2+yp^2)^1.5;
   k2y=-k*yp*MZ/(xp^2+yp^2)^1.5-k*MM*yp/((xp-xM)^2+yp^2)^1.5;
   xsZ(i)=xsZ(i-1)+vx*h+(k1x+k2x)*h^2/4;
   ysZ(i)=ysZ(i-1)+vy*h+(k1y+k2y)*h^2/4;
   vx=vx+(k1x+3*k2x)*h/4;
   vy=vy+(k1y+3*k2y)*h/4;
   t(i)=t(i-1)+h;
   i=i+1;
endwhile
# --- vykreslení zavislosti
```

```
38 axis('equal');
39 hold on
40 # --- Vykresleni Zeme
41 u=linspace(0,2*pi,100);
42 plot(cos(u),sin(u),'b')
43 # --- Vykresleni Mesice
44 plot(1738000/RZ*cos(u)+xM/RZ,1738000/RZ*sin(u),'g')
45 # --- vykresleni trajektorie
46 plot(xsZ/RZ,ysZ/RZ,'r');
47 hold off
```

Počáteční podmínky odpovídají hodnotám $x_{\mathrm{SZ}0} = 0$, $y_{SZ0} = 10R_{\mathrm{Z}}$, $v_{x0} =$ $= 2{,}7\,\mathrm{km{\cdot}s^{-1}}$ a $v_{y0} = 1{,}84\,\mathrm{km{\cdot}s^{-1}}$. Výsledná podoba trajektorie je přitom na změnu počátečních podmínek velmi citlivá. V podmínce cyklu `while` na 21. řádku testujeme nejenom zda čas t nepřekročil stanovenou horní mez, ale také dopad na zemský povrch (tj. zda $r > R_{\mathrm{Z}}$). Kromě samotné trajektorie (příkazem `plot` na 46. řádku) jsou vykresleny i kružnice znázorňující povrch Země i Měsíce (42. resp. 44. řádek), opět nastavujeme stejné měřítko na obou osách příkazem `axis('equal')`. Výsledek znázorňuje obr. 10.4. Dodejme, že pokud bychom použili Eulerovu namísto Rungovy-Kuttovy metody, k dosažení stejné přesnosti by se citelně prodloužila doba výpočtu, zejména na pomalejších počítačích.



Obr. 10.4: Příklad pohybu družice v soustavě Země–Měsíc

10.2.3 Pružné kyvadlo

Model popsaný již v části 2.3 je tvořen kyvadlem, u něhož je závěs realizován pružinou o tuhosti k a délce l (v nezatíženém stavu). Jde o poměrně jednoduchý a zároveň velmi zajímavý model, na němž lze studovat řadu jevů spojených s oscilacemi. Svědčí o tom mimo jiné dva články z nedávné doby [5, 8], jež můžeme

vřele doporučit zájemcům o hlubší seznámení s problematikou i o odkazy na další zdroje informací.

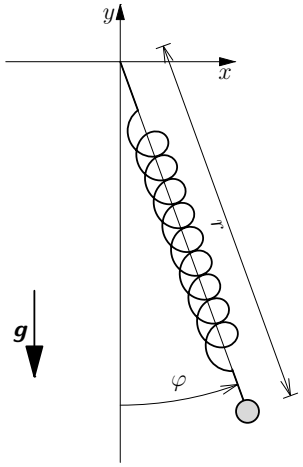
Pro naše numerické řešení vystačíme s pohybovou rovnicí

$$m\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{G} - k\,(r-l)\;\frac{\boldsymbol{r}}{r},$$

kterou lze v kartézských souřadicích $x = r\sin\varphi$, $y = r\cos\varphi$ rozepsat do složek

$$\begin{aligned} m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} &= -k\,(r-l)\;\frac{x}{r}, \\ m\frac{\mathrm{d}^2y}{\mathrm{d}t^2} &= -g - k\,(r-l)\;\frac{y}{r}. \end{aligned}$$

Počátek souřadnic přitom volíme v místě závěsu podle obr. 10.5.



Obr. 10.5: K zavedení souřadnic pro pružné kyvadlo

Jedná se o systém s dvěma význačnými módy (kmity kyvadla spojené se změnou souřadnice φ a kmity pružiny spojené se změnou r), mezi nimiž je nelineární vazba, která je zodpovědná za specifické vlastnosti pohybu. Rovnovážná poloha, kdy je závaží po zavěšení na pružinu v klidu ve svislé ose y je určena vzdáleností od bodu závěsu $r = l_{\mathrm{g}} = l + mg/k$, vůči této hodnotě budeme v souladu s [5] vykreslovat změny vzdálenosti závaží od bodu závěsu. Pro snadnější porovnání s uvedenými prameny [5, 8] volíme tytéž parametry – nezatíženou délku pružiny $l = 0{,}28\,\mathrm{m}$, tuhost pružiny $k = 12{,}5\,\mathrm{N{\cdot}m}$.

Jak je odvozeno v [5, 8], předávání energie mezi zmíněnými módy je nejvýraznější při rezonanci, kdy platí

$$\omega_r = \sqrt{\frac{k}{m}} = 2\omega_\varphi = \sqrt{\frac{g}{l_{\mathrm{g}}}}.$$

Po dosazení za l_{g} dostaneme hmotnost závaží, které musíme zavěsit, aby rezonance nastala

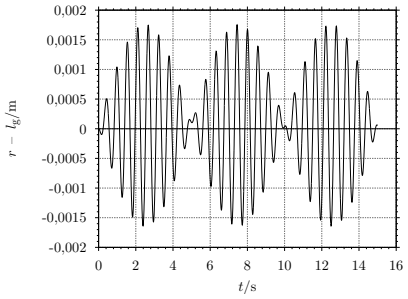
$$m = \frac{lk}{3g} \approx 0{,}12\,\mathrm{kg}. \tag{10.3}$$

Ukažme nejprve případ dominantních kyvů, kdy kývání kyvadla bude minimálně ovlivněno kmity pružiny. Předpokládáme, že na počátku závaží o hmotnosti

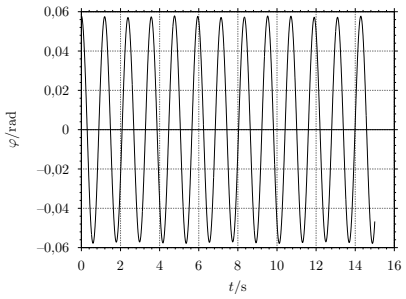
$m = 0{,}09\,\mathrm{kg}$ vychýlíme o úhel $\varphi_0 = 0{,}05785\,\mathrm{rad}$, aniž by se pružina prodloužila, tj. $r_0 = l_\mathrm{g}$. Výpis programu `pruzkrk.m` pak může být následující:

```
# --- numericke reseni pruzneho kyvadla
clear;
clc;
# --- konstanty
l=0.28;       # delka nezatizene pruziny
k=12.5;       # tuhost pruziny
g=9.81;       # tihove zrychleni
m=0.09;       # hmotnost zavazi
lg=l+m*g/k;   # rovnovazna delka pruziny
# --- pocatecni podminky
d0=0.00;     # pocatecni prodlouzeni pruziny
phi0=0,05785; # pocatecni vychylka od svisleho smeru
x0=(lg+d0)*sin(phi0);    # prepoctena soradnice x0
y0=-(lg+d0)*cos(phi0);   # prepoctena soradnice y0
vx0=0;        # pocatecni rychlost ve smeru x
vy0=0;        # pocatecni rychlost ve smeru x
# --- oblast integrace
t0=0;         # cas merime od 0
mez=15;       # horni mez integrace
h=5e-3;       # integracni krok
# --- vlastni numericka integrace - RK metoda 2. radu
t(1)=t0;
x(1)=x0;
y(1)=y0;
vx=vx0;
vy=vy0;
i=1;
# --- cyklus probiha pro t<= horni mez
 while (t(i)<=mez)
   i=i+1;
   r=sqrt(x(i-1)^2+y(i-1)^2);
   k1x=-k*(r-l)/r*x(i-1)/m;
   k1y=-g-k*(r-l)/r*y(i-1)/m;
   xp=x(i-1)+vx*2*h/3+k1x*2*h^2/9;
   yp=y(i-1)+vy*2*h/3+k1x*2*h^2/9;
   rp=sqrt(xp^2+yp^2);
   k2x=-k*(rp-l)/rp*xp/m;
   k2y=-g-k*(rp-l)/rp*yp/m;
   x(i)=x(i-1)+vx*h+(k1x+k2x)*h^2/4;
   y(i)=y(i-1)+vy*h+(k1y+k2y)*h^2/4;
   vx=vx+(k1x+3*k2x)*h/4;
   vy=vy+(k1y+3*k2y)*h/4;
   t(i)=t(i-1)+h;
 endwhile
# --- vykresleni trajektorie
```

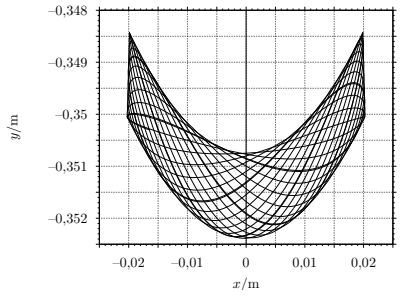
```
46 plot(x,y);
47 # --- vykresleni závislosti r-lg na case
48 plot(t,sqrt(x.^2+y.^2)-lg);
49 # --- vykresleni závislosti phi na case
50 plot(t,atan2(x,abs(y)))
```



(a) Závislost prodloužení $r(t) - l_\mathrm{g}$ na čase



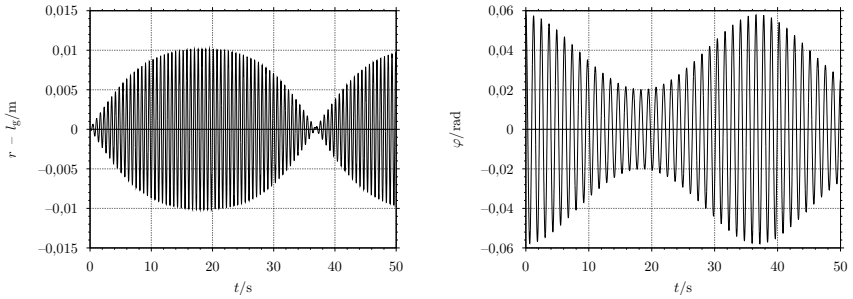
(b) Závislost výchylky φ na čase



(c) Výsledný pohyb v rovině xy

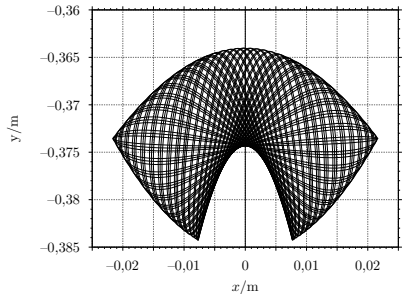
Obr. 10.6: Kmity pružného kyvadla v režimu dominantních kyvů

Vidíme, že počáteční délku pružiny zadáváme v programu na 11. řádku rozdílem $d_0 = r_0 - l_\mathrm{g}$, jež je v tomto případě nulový, počáteční hodnotu kartézských souřadnic x_0 a y_0 pak dopočteme podle již uvedených převodních vztahů. Naopak, z hodnot x a y získaných integrací získáme $r = \sqrt{x^2 + y^2}$ a $\varphi = \operatorname{arctg} x/|y|$, neboť v naší volbě osa y směřuje nahoru a pod závěsem je y záporné. Použili jsme tu dvě dosud nezmíněné funkce programu `atan2(a,b)` pro výpočet $\operatorname{arctg}(a/b)$ a



(a) Závislost prodloužení $r(t) - l_\mathrm{g}$ na čase

(b) Závislost výchylky φ na čase



(c) Výsledný pohyb v rovině xy

Obr. 10.7: Kmity pružného kyvadla v blízkosti rezonance

`abs` pro výpočet absolutní hodnoty. Výsledné kmity jsou znázorněny na obr. 10.6. Je zřejmé, že změny prodloužení pružiny jsou relativně velmi malé a výslednému pohybu skutečně dominují kyvy v souřadnici φ. Toto tvrzení je jen zdánlivě v rozporu s výslednou trajektorií na obr 10.6c, neboť změny v souřadnici y jsou mnohem menší než v souřadnici x.

Změníme-li hmotnost závaží na $m = 0{,}12\,\mathrm{kg}$ a prodloužíme časový interval pro integraci, abychom zachytili delší průběh pohybu, mělo by podle rovnice (10.3) docházet k rezonanci. Na obr. 10.7 je skutečně patrné postupné předávání energie mezi kyvy kyvadla a kmity pružiny. I zde platí, že tvar trajektorie na obr. 10.7 je deformován nestejným měřítkem, změny ve svislém směru jsou ve skutečnosti mnohem menší než ve směru vodorovném. Pokud bychom ale na-

stavili na obou osách stejné měřítko, trajektorie by se nám redukovala ve „tlustě rozmazanou čáru“ kyvů.

10.3 Modely využívající funkce `lsode`

Modely využívající procedury `lsode` jsou vesměs stručnější a kompaktnější, naopak je u nich obtížnější testovat případnou dodatečnou podmínku (např. dopad na rovinu $y = 0$ jako v části 10.1.1) a při jejím nesplnění výpočet zastavit. Mezi tyto modely patří pochopitelně i příklady uvedené v částech 9.2.1 a 9.2.2.

10.3.1 Balistická křivka podruhé

Vraťme se ještě jednou k balistické křivce popsané v části 10.1.1. Pokud vyjdeme z rovnic (10.1a) a rozhodneme se použít funkci `lsode`, může soubor `balist.m` obsahovat příkazy

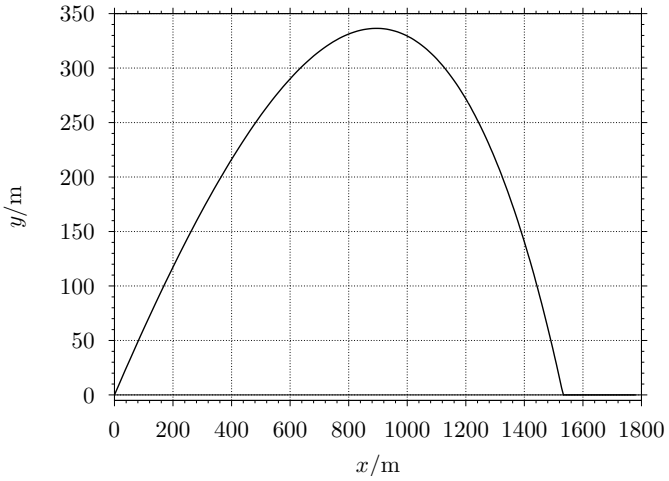
```
# --- numericke reseni balistickeho problemu
clear;
clc;
# --- definice pocatecnich podminek
t0=0;
x0=0;
y0=0;
v=input('zadej pocatecni rychlost v m/s: ');
alpha=input('elevacni uhel ve stupnich: ')*pi/180;
vx=v*cos(alpha);
vy=v*sin(alpha);
# --- parametry prostredi a strely
R=0.04;                 # polomer strely
global m=2.5;           # hmotnost strely
ro=1.3;                 # hustota vzduchu
                        # koeficient odporu
C=input('zadej koeficient odporu C (<0,1>): ');
global g=9.81;          # tihove zrychleni
S=pi*R^2;
global k=C*ro*S/2;
# --- definice funkce pro integraci
function xdot=fce(xp,t)
 global k m g # prebira globalne definovane konstanty
  xdot=zeros(4,1);
  xdot(1)=xp(2);
  xdot(2)=-k/m*sqrt(xp(2).^2+xp(4).^2).*xp(2);
  xdot(3)=xp(4);
  xdot(4)=-g-k/m*sqrt(xp(2).^2+xp(4).^2).*xp(4);
endfunction
# --- meze integrace a cas jako nezavisle promenna
```

```
31 dm=0;
32 hm=2*vy/g;
33 t=linspace(dm,hm,200);
34 # --- vlastni numericka integrace pomoci lsode
35 yp=lsode("fce",[x0,vx,y0,vy],t);
36 x=yp(:,1);
37 y=yp(:,3).*(yp(:,3)>=0);
38 # --- vykresleni reseni
39 plot(x,y);
```

Novým, v tomto textu dosud nepoužitým příkazem je ze `input` na řádcích 8, 9 a 17. Poslouží nám tehdy, pokud chceme vytvořit interaktivní program, který po spuštění požádá uživatele (našeho studenta) o vložení hodnot příslušných proměnných. Konkrétně na základě 8. řádku vypíše program

```
zadej pocatecni rychlost v m/s:
```

a čeká na vložení číselné hodnoty ukončené klávesou Enter. Jak vidíme, máme možnost zadat text výzvy, která se objeví na obrazovce. V sofistikovanějších programech bychom mohli ošetřit, zda zadané hodnoty splňují další dodatečné podmínky (nezápornost apod.).



Obr. 10.8: Balistická křivka pro $v_0 = 200\,\mathrm{m{\cdot}s^{-1}}$, $\alpha = 32°$ a $C = 0{,}48$

Jak již bylo uvedeno výše, na rozdíl od Eulerovy metody s cyklem `while` použité v části 10.1.1 zde neukončíme integraci po dopadu na rovinu $y = 0$,

proto musíme záporné hodnoty souřadnice y odfiltrovat jinak. Nechceme-li nějakým vedlejším výpočtem omezovat horní mez integrace (zde, jak je zřejmé ze 32. řádku, je zvolena jako doba šikmého vrhu v neodporujícím prostředí), využijeme jedné z dalších metod efektivní práce s maticemi. Na 37. řádku se setkáváme se sloupcovou maticí `yp(:,3)>=0`, kterou vytváříme ze 3. sloupce matice `yp` pomocí logické podmínky `>=0`. Výsledkem bude vektor (sloupcová matice), v níž budou buď jedničky (na místech, kde prvky `yp` podmínky splňují, tj. zde jsou nezáporné) nebo nuly (na těch řádcích, kde hodnoty `yp` podmínku nesplňují). Vynásobíme-li poté prvky této matice původní vektor `yp(:,3)` (pozor, ne matice jako celek, ale odpovídající prvky mezi sebou, takže musíme použít `.*`), násobíme její nezáporné prvky 1 a záporné 0, takže získáme matici obsahující nezáporné hodnoty `yp(:,3)` a na místech záporných hodnot 0. Výsledná křivka pro interaktivně zadané hodnoty $v_0 = 200\,\mathrm{m{\cdot}s^{-1}}$, $\alpha = 32°$ a $C = 0{,}48$ je znázorněna na obr. 10.8.

10.3.2 Foucaultovo kyvadlo

Stáčení roviny kmitu Foucaultova kyvadla vlivem Coriolisovy síly patří od svého prvního předvedení Jeanem Bernardem Léonem Foucaultem v pařížském Pantheonu v roce 1851 k základním důkazům rotace Země. Foucaultův model prý nesl hmotnost $28\,\mathrm{kg}$ na laně dlouhém $68\,\mathrm{m}$ [1].

Zvolíme-li vztažnou soustavu spojenou s rotujícím povrchem Země tak, že osa x míří v daném místě ve směru poledníku na jih, osa y ve směru rovnoběžky na východ a osa z svisle vzhůru, omezíme-li se na malé kmity vzhledem k délce kyvadla, lze pro průmět kmitů do roviny xy v místě se zeměpisnou šířkou λ odvodit přibližné rovnice (viz např. [22])

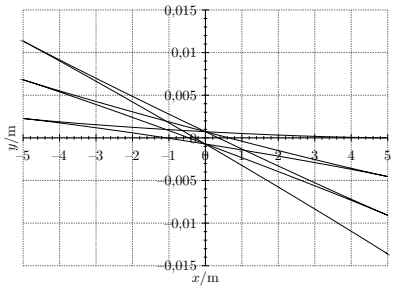
$$\begin{aligned} \ddot{x} &= -\omega^2 x + 2\dot{y}\Omega\sin\lambda, \\ \ddot{y} &= -\omega^2 y - 2\dot{x}\Omega\sin\lambda, \end{aligned}$$

v nichž $\omega = \sqrt{g/l}$ je úhlová frekvence matematického kyvadla téže délky a $\Omega \approx 7{,}3{\cdot}10^{-5}\,\mathrm{s^{-1}}$ úhlová frekvence otáčení Země okolo vlastní osy. Do souboru `foucaoult.m` můžeme napsat

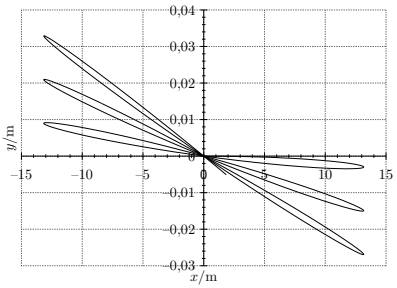
```
1 # --- numericke reseni Foucaultovo kyvadlo
2 clear;
3 clc;
4 # --- definice pocatecnich podminek
5 t0=0;
6 x0=5;
7 y0=0;
8 vx0=0;
```

```
vy0=0;
global phi=50/180*pi;       # zemepisna sirka
global om=2*pi/(86400);     # uhlova rychlost rotace Zeme
global l=68;                # delka kyvadla
global g=9.81;
# --- vlastni numericka integrace - preddefinovana procedura lsode
function xdot=fce(xp,t)
global phi om g l           # prebira globalne definovane konstanty
  xdot=zeros(4,1);
  xdot(1)=xp(2);
  xdot(2)=-g/l*xp(1)+2*om*sin(phi)*xp(4);
  xdot(3)=xp(4);
  xdot(4)=-g/l*xp(3)-2*om*sin(phi)*xp(2);
endfunction
# --- meze integrace
dm=0;
hm=50;
# --- cas jako nezavisla promenna, oblast integrace
t=linspace(dm,hm,500);
# --- reseni soustavy diferencialnich rovnic
yp=lsode("fce",[x0,vx0,y0,vy0],t);
# --- vykresleni
plot(yp(:,1),yp(:,3));
```

Řešení vidíme na obr. 10.9a. Kyvadlo bylo v tomto případě uvedeno do pohybu puštěním z maximální výchylky. Pokud změníme počáteční podmínky tak, že kyvadlo rozhoupeme udělením počáteční rychlosti z rovnovážné polohy, bude mít průmět trajektorie do roviny xy podobu znázorněnou na obr. 10.9b. Připomeňme, že měřítko na osách není stejné, za uvažovaných $50\,\mathrm{s}$ nemůže být výchylka ve směru osy y velká (naopak je řádově skoro 1000-krát menší).



(a) Počáteční podmínky $x_0 = 5\,\mathrm{m}$, $v_{x0} = 0\,\mathrm{m{\cdot}s^{-1}}$



(b) Počáteční podmínky $x_0 = 0\,\mathrm{m}$, $v_{x0} = 5\,\mathrm{m{\cdot}s^{-1}}$

Obr. 10.9: Kmity Foucaultova kyvadla

Literatura

[1] Wikipedie, otevřená encyklopedie, URL: `http://cs.wikipedia.org/`.

[2] *Bajer J.*: Mechanika 3. PřF UP Olomouc 2006.

[3] *Bartsch H.J.*: Matematické vzorce. SNTL, Praha 1984.

[4] *Brdička M. – Hladík A.*: Teoretická mechanika. Academia, Praha 1987.

[5] *Dvořák L.*: Pružné kyvadlo: od teoretické mechaniky k pokusům a zase zpátky. PMFA 51 (2006), č. 4, s. 312–327.

[6] *GNU project*: GNU General Public License v. 2 (český překlad), URL: `http://gnu.cz/article/30/pdf/gpl-cz.pdf`.

[7] *Greiner W.*: Classical mechanics: systems od particles and Hamiltonian dynamics. Springer-Verlag, New York 2003.

[8] *Havránek A. – Čertík O.*: Pružné kyvadlo. PMFA 51 (2006), č. 3, s. 198–216.

[9] *Holíková L.*: Použití numerických metod v úlohách středoškolské fyziky. Dipl. práce, UP Olomouc 2006. URL: `http://optics.upol.cz/~richterek/files.html`.

[10] *Jarník V.*: Diferenciální počet I. Academia, Praha 1984.

[11] *Just M.*: Octave – český průvodce programem, URL: `http://www.octave.cz/`.

[12] *Lepil O.*: Dynamické modelování. Studijní text, UP, Olomouc 2001.

[13] *Mařík R.*: Průběh funkce (Matematika (nejen pro) krajináře a nábytkáře), URL: `http://user.mendelu.cz/marik/mat-web/mat-webse4.html`.

[14] *Míka S.*: Numerické metody algebry. Matematika pro vysoké školy technické sešit IV, SNTL, Praha 1982.

[15] *Poláček J.*: GNU Octave, URL: `http://www.abclinuxu.cz/software/veda/gnu-octave`.

[16] *Přikryl P.*: Numerické metody matematické analýzy. Matematika pro vysoké školy technické sešit XXIV, SNTL, Praha 1985.

[17] *Rektorys K. a kol.*: Přehled užité matematiky I, II. SNTL, Praha 1988.

[18] *Strogatz S.H.*: Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering. Westview Press 2000.

[19] *Šedivý P.*: Užití numerických metod při řešení rovnic ve fyzikálních úlohách. Studijní text pro řešitele 35. ročníku FO, MAFY, Hradec Králové 1996.

[20] *Šedivý P.*: Modelování pohybů numerickými metodami. Knihovnička Fyzikální olympiády č. 38, Gaudeamus, Hradec Králové 1999.

URL: http://fo.cuni.cz/texty/modelov.pdf.
[21] *Šedivý P. – Volf I.*: Pohyb tělesa po eliptické trajektorii v radiálním gravitačním poli. Knihovnička FO č. 43, MAFY, Hradec Králové 2000. URL: http://fo.cuni.cz/texty/druzice.pdf.
[22] *Trkal V.*: Mechanika hmotných bodů a tuhého tělesa. ČSAV, Praha 1956.
[23] *Volf I. – Šedivý P.*: Pohyby tělesa v odporujícím prostředí. Hradec Králové 1992. URL: http://fo.cuni.cz/texty/odpor.pdf.
[24] *Vybíral B. – Zdeborová L.*: Pohyb těles s vlivem odporových sil. Knihovnička FO č. 55, MAFY, Hradec Králové 2002. URL: http://fo.cuni.cz/texty/odpory.pdf.
[25] *Williams T. Kelley C. – others.*: Gnuplot. An Interactive Plotting Program, URL: http://www.gnuplot.info/docs/gnuplot.html.

Seznam příkazů použitých v textu

Dynamické modelování

Seminární materiál projektu Učíme fyziku moderně
Další vzdělávání učitelů fyziky Olomouckého kraje
Slovanské gymnázium Olomouc

Autoři: doc. RNDr. Oldřich Lepil, CSc.
Mgr. Lukáš Richterek, Ph.D.

Vydal Repronis v roce 2007 jako svou 261. publikaci.

Počet stran: 160
Náklad: 120 výtisků
Vydání: první
Tisk: Repronis s. r. o., Ostrava

ISBN 978-80-7329-156-3