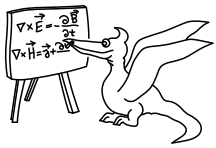


Seriál: Numerická kopaná

A je to tady, naučíme se dělat numerické simulace. Na příkladu rotujícího fotbaláku si ukážeme, jak numericky integrovat diferenciální rovnice v programu Octave. Plánoval jsem do tohoto dílu nacpat ještě nějaké historické povídání, ale možná více oceníte, pokud si opravdu důkladně projdeme krátký prográmek k výpočtu tohoto problému. Pro člověka neožehnutého programováním může následující text vypadat dost hrozivě, ale nestrachujte se, půjdeme krok po kroku a nemusíte všemu ihned rozumět. Pro řešení seriálových úloh stačí program opsat a měnit jen příslušné parametry.

Milión minikroků

Co to tedy znamená numericky integrovat? Jedná se fakticky o variaci na přiblížení derivace pomocí diskrétní změny. Například rychlost je (pro jednorozměrný případ)

$$v = \frac{\mathrm{d}x}{\mathrm{d}t} \approx \frac{\Delta x}{\Delta t}\,, \tag{1}$$

a zrychlení je

$$a = \frac{\mathrm{d}x}{\mathrm{d}t} \approx \frac{\Delta v}{\Delta t}\,, \tag{2}$$

kde Δt je nějaký malý časový úsek, během kterého se daná poloha a rychlost změnily o Δx, respektive Δv. S použitím $F(x) = ma$ a rovnic (1) a (2) tedy můžeme přibližně vypočítat, že

$$x(t+\Delta t) \approx x(t) + v(t)\Delta t \quad \text{a} \quad v(t+\Delta t) \approx v(t) + \Delta t\frac{F(x)}{m}\,. \tag{3}$$

Když pak použijeme hodně malé Δt a proceduru mnohokrát opakujeme, získáváme přibližný vývoj systému s integračním krokem Δt. Všimněme si, že bychom mohli v rovnici (3) použít bez velkých potíží $F(x+\Delta x)$ a tím získat přesnější aproximaci. Takovýchto triků a dalších metod existuje velmi mnoho – stačí si jen pamatovat, že numerická integrace přibližně řeší spojitou diferenciální rovnici pomocí jednoduchých diskrétních kroků. Pro zmenšující krok se pak jedná o přesnější a přesnější aproximaci opravdového řešení. Pro dostatečně malý krok dostaneme v řadě případů numerickým řešením fakticky přesný vývoj daného systému.

Nadatlování do Octave

Pojďme se teď podívat na Octave. Po spuštění by se vám měl objevit terminál s příkazovou řádkou. Sám jsem si nainstaloval poslední vydání *Octave Forge* pro Windows a nezapomněl zaškrtnout balík *odepkg* při instalaci. Poslední řádek hned po spuštění pro mě pak vypadá takto:

```
octave-3.6.4.exe:1>_
```

Ve Windows 8 se musí poupravit zástupce programu Octave kliknutím pravým tlačítkem myši, zvolením *Vlastnosti* a přidáním `-i --line-editing` do pole *Cíl* – jinak se vám tato příkazová řádka nezobrazí. Můžete si přečíst základní tutoriál na `http://octave.cz` pro obecnější informace, my se rovnou vrhneme na integraci rovnic pro letící míč.

Než začneme zapisovat rovnice do programu, musím se omluvit za lehkou botu z minulého dílu – napsal jsem špatně znaménko u Magnusovy síly. Správně je Magnusova síla

$$\boldsymbol{F}_{\mathrm{Magn}} = \alpha v(\boldsymbol{\Omega} \times \boldsymbol{v})\,.$$

Rovnice z posledního dílu zformulujeme pro speciální případ, kdy míč dostane počáteční rychlost v rovině xz, a v ní také přesně rotuje, takže jej Magnusova síla strhává jen do směrů xz, a trajektorie tedy nikdy z původní roviny nevybočí. Tj. $\vec{\Omega}$ ukazuje do směru osy y a

$$\vec{\Omega} \times \vec{v} = (0, \Omega, 0) \times (v_x, 0, v_z) = (\Omega v_z, 0, -\Omega v_x)\,,$$

kde pokud zvolíme kladné Ω, znamená to, že se míč otáčí „dolů za nosem“, neboli že Magnusova síla strhává míč dolů při kladné v_x. Dá to trochu úsilí točit s prsty, aby si člověk zkonstruoval pravotočivé pořadí os a dobře zadefinoval směr vektoru úhlové rychlosti, ale doporučuji si ověřit, že Magnusova síla má s tímto znaménkem přesně takovýto efekt.

Zápis pohybových rovnic do Octave začneme napsáním příkazu

```
pkg load odepkg
```

čímž se načte balík na integraci diferenciálních rovnic. Pak pokračujeme definicí funkce časové derivace našich proměnných:

```
function xidot = f(t,xi)
  alfa = 0.1;
  beta = 0.4;
  gamma = 0.0001;
  g = 9.81;
  m = 1;
  r2 = 0.01;
  vx = xi(3);
  vz = xi(4);
  Omega = xi(5);
  v = sqrt(vx^2+vz^2);
  ax = v/m*(alfa*Omega*vz-beta*vx);
  az = -g+v/m*(-alfa*Omega*vx-beta*vz);
  Omegadot = -3*gamma*v^3/(2*m*r2)*Omega;
  xidot = [vx;vz;ax;az;Omegadot];
endfunction
```

Proměnné `alfa`, `beta`, `gamma`, `g`, `m`, `r2` odpovídají parametrům α, β, γ, g, m, r^2 z posledního dílu. Používáme zde trik, ve kterém ukládáme polohu, rychlost a úhlovou rychlost otáčení míče do jednoho vektoru

$$\xi = (x, z, v_x, v_z, \Omega)\,,$$

kde v_x odpovídá `vx`, v_z odpovídá `vz` a Ω odpovídá `Omega`. Definujeme pak v tomto bloku funkci `f(t,xi)`, která definuje první časovou derivaci vektoru ξ, tj.

$$\dot{\xi} = (v_x, v_z, a_x, a_z, \dot{\Omega})\,.$$

Když se pak někde dále v programu objeví funkce `f`, bude navracet vektor `xidot` tak, jak je sestavený v předposledním řádku.

K tomu, aby se nám integrace včas zastavila, musíme zadefinovat funkci, která integrátoru ohlásí okamžik, kdy míč dopadl na zem. Zvolíme si počátek souřadnic tak, že $x = 0$, $z = 0$ je startovací poloha míče na zemi a tudíž $z = 0$ je okamžik dopadu. Funkce `dopad` tedy bude vypadat takto:

```
function [hodnota,ukoncuje,smer] = dopad(t,xi)
  hodnota = xi(2);
  ukoncuje = 1;
  smer = 0;
endfunction
```

Tato funkce vrátí trojici čísel `[hodnota,ukoncuje,smer]`. Když ji vložíme do numerického integrátoru, při každém kroku integrace se podívá na `hodnota`, pokud je nulová, podívá se, jestli má běh ukončit na `ukoncuje` (1 znamená ano) a případně se dokáže rozhodnout podle `smer`, jestli ukončit podle toho, jestli k protnutí došlo z kladné nebo záporné `hodnota`. V našem případě je `hodnota` $\xi_2 = z$.

Třidvajedna integrace

Teď už můžeme řešit rovnici ve třech řádcích:

```
nastaveni = odeset('Events',@dopad,'InitialStep', 0.00001,'MaxStep',0.0001);
pocPodminka = [0,0.0001,10,3,1];
reseni = ode45(@f,[0,10],pocPodminka,nastaveni);
```

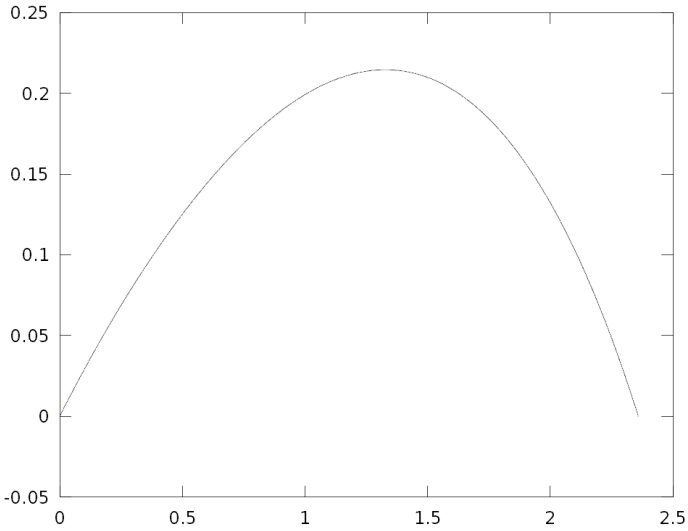
kde `nastaveni` je pomocná proměnná, do které si ukládáme nastavení integrátoru `odeset`. Část `'Events',@dopad` určuje, že integrátor čeká zmíněným způsobem na funkci `dopad`. Možnosti `'InitialStep',0.00001,'MaxStep',0.0001` pak určují, v jakých rozmezích se pohybuje diskrétní integrační krok Δt.

`pocPodminka` určuje počáteční polohu, rychlost a rotaci pomocí pětisložkového vektoru ξ. Pro jednoduchost jsme nastavili počáteční $\xi_2 = z = 0{,}000\,1$, aby se nám okamžitě nesepnula podmínka dopadu.

Řádek `reseni=ode45(@f,[0,10],pocPodminka,nastaveni);` už pak jen říká integrátoru `ode45`, aby zintegroval rovnice s danou počáteční podmínkou a nastavením, derivací `f(t,xi)` a s maximálním rozsahem času `[0,10]` (to můžete prodloužit, ale typicky míč spadne mnohem dříve). Výsledek se pak uloží pod proměnnou `reseni`. Pokud vše předcházející správně napíšete do příkazového řádku, spustí se integrace, která by na normálním počítači měla trvat pár desítek sekund. Po jejím skončení si můžete vykreslit výsledky příkazem:

```
plot(reseni.y(:,1),reseni.y(:,2))
```

Výsledný objekt `reseni` totiž obsahuje sloupec čísel s časy t_i pod `reseni.x` a sloupec vektorů $\xi(t_i)$ v odpovídajících časech (zkuste napsat daná klíčová slova do příkazové řádky a Octave vám ukáže obsah těchto sloupců). Pod `reseni.y(i,j)` najdete j-tou složku vektoru ξ v čase t_i, tj. $\xi_j(t_i)$. `reseni.y(:,1)` pak znamená $\xi_1 = x$ ve všech časech a `reseni.y(:,2)` znamená $\xi_2 = z$ ve všech časech.



Obr. 1: Trajektorie míče s danými parametry a počátečními podmínkami. Všimněte si, že osa z má úplně jinou škálu než osa x. Doopravdy je trajektorie mnohem „plošší“.

Dohromady tedy `plot(reseni.y(:,1),reseni.y(:,2))` říká, aby Octave načrtlo polohy x, z skrze všechny časové kroky t_i. Na obrazovce by vám po tomto příkazu měl vyskočit graf s trajektorií míče v rovině xz od výkopu až po dopad na zem – docela jako na obrázku 1.

Řekneme si už jen poslední tip, v úlohách si budete muset trochu hrát s hodnotami různých konstant a s počátečními podmínkami, a proto by bylo dobré se naučit, jak toto vše Octave zadat pomocí jednoduchého skriptu. To uděláte tak, že vše předchozí sepíšete do jednoho textového souboru s příponou `.m` (ve Windows například v programu Poznámkový blok) a uložíte do nějaké složky. Mějme například skript s cestou `C:/cesta/ksouboru/mujskript.m`. Pustíte si jej pak v Octave takto:

```
chdir('C:/cesta/ksouboru');
mujskript;
```

Doufám, že tento díl seriálu nebyl příliš vyčerpávající a že se vám program na integraci podařilo spustit. V případě obtíží mi napište na e-mail `witzanyv@fykos.cz`.

Příští díl bude oddechovější, dozvíte se o tom, jak James Clerk Maxwell, jeden z nejvýznamnějších teoretických fyziků vůbec, došel k závěru, že je příroda nevypočitatelná, ale že i přesto *žádný levhart nemůže změnit svoje skvrny!* Co tím myslel? A jak to souvisí se slibovaným chaosem? Nechte se překvapit.

Fyzikální korespondenční seminář je organizován studenty MFF UK. Je zastřešen Oddělením pro vnější vztahy a propagaci MFF UK a podporován Ústavem teoretické fyziky MFF UK, jeho zaměstnanci a Jednotou českých matematiků a fyziků.