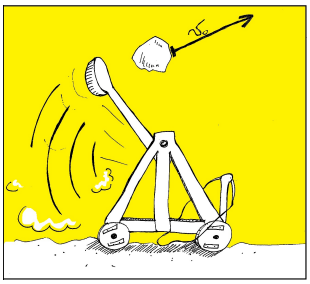
Rzut ukośny, czyli o prawie Newtona i równaniach różniczkowych

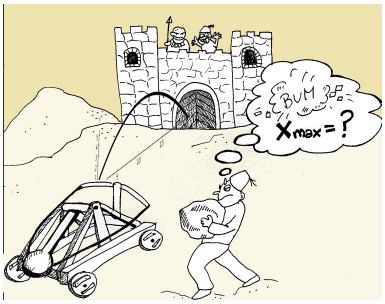
29 kwietnia 2014



Z lekcji fizyki wszyscy wiemy, że prędkość cząstki $v = \frac{\Delta x}{\Delta t}$ opisuje tempo zmiany położenia Δx w czasie Δt. Z kolei przyspieszenie $a = \frac{\Delta v}{\Delta t}$ opisuje tempo zmiany prędkości Δv w czasie Δt. Te pozornie mało spektakularne wzory nabierają Mocy dopiero w zetknięciu z komputerem. Czytelnika chcącego głębić tajniki metody rozwiązywania równań różniczkowych zapraszamy do *aktywnej* lektury tego artykułu.

Odkrywanie praw fizycznych przez "komputerowe eksperymentowanie" z równaniami jest celem projektu dydaktycznego iCSE prowadzonego na Uniwersytecie Śląskim. Narzędzie stanowi system Sage [1] będący otwartą implementacją systemu algebry komputerowej z językiem Python. Program Sage jest dostępny z poziomu przeglądarki internetowej poprzez usługę chmurową [2] lub serwer pojedyńczych obliczeń na którym bazuje interaktywna wersja tego artykułu [3].

1 Rzut ukośny



Rozpoczniemy od pozornie nudnego tematu jakim może się wydawać znany ze szkoły rzut ukośny. W fizyce pojęcie to oznacza ruch punktu materialnego o pewnej masie pod działaniem siły grawitacji (siły przyciągania ziemskiego). Wiedza na temat tego zagadnienia była kluczowa podczas konfliktów zbrojnych. Wysyłając bowiem w stronę wroga punkty materialne w postaci różnorakich pocisków, chcieliśmy wiedzieć czy aby na pewno dotrą one do celu. Z tego punktu widzenia interesujący jest zasięg posiadanego przez nas urządzenia miotającego, oczywiście im większy tym lepszy. Można się spodziewać, że ów zasięg zależy od siły miotającej naszego urządzenia, a co za tym idzie prędkości początkowej pocisku. Kolejną rzeczą jest uwzględnienie kąta wystrzału. Jeśli strzelimy pocisk pionowo to z pewnością upadnie na nas - zasięg będzie zerowy. Jeśli strzelimy poziomo to również pocisk nie poleci zbyt daleko. A zatem pod jakim kątem powinniśmy strzelać? Odpowiedzi dręczące artylerzystów dostarcza Fizyka, a mówiąc konkretnie jej gałąź zwana dynamiką. Rozwiązując więc odpowiednie zagadnienie dynamiki dostajemy tor ruchu pocisku.

Podstawowym prawem jest słynna druga zasada dynamiki Newtona, która wyraża się równaniem:

$$m\vec{a} = \vec{F}. \tag{1}$$

W równaniu tym m jest masą pocisku, $\vec{a}$ jest przyśpieszeniem (wektorem o składowej poziomej i składowej pionowej) i $\vec{F}$ jest siłą także o dwóch składowych: poziomej i pionowej. Podczas lekcji z reguły rozważa się przypadek, w którym pomijamy siły tarcia. Wówczas jedyna siła działająca na ciało to siła o składowych $(0, -mg)$, tzn, składowa pozioma wynosi 0, natomiast siła w kierunku pionowym wynosi mg i jest skierowana w dół (dlatego jest znak minus!). Możemy w takim przypadku rozważać ruch odbywający się w płaszczyźnie rozpiętej przez wektor siły i prędkości początkowej - czyli nasz pocisk będzie leciał w płaszczyźnie pionowej do Ziemi i nie będzie skręcał w lewo ani w prawo.

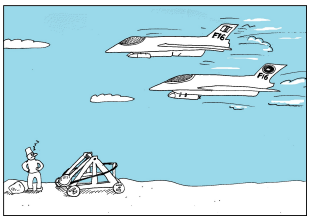
Klasyczne rozwiązanie tego problemu oparte jest o fakt, że ruch w poziomie x i ruch w pionie y są od siebie niezależne. W kierunku pionowym ruch jest jednostajny, a w pionie zachodzi ruch jednostajnie przyśpieszony. Można to zapisać w postaci [4]:

$$\begin{aligned} x(t) &= x_0 + v_{x0}t, \\ y(t) &= y_0 + v_{y0}t - \frac{1}{2}gt^2. \end{aligned} \tag{2}$$

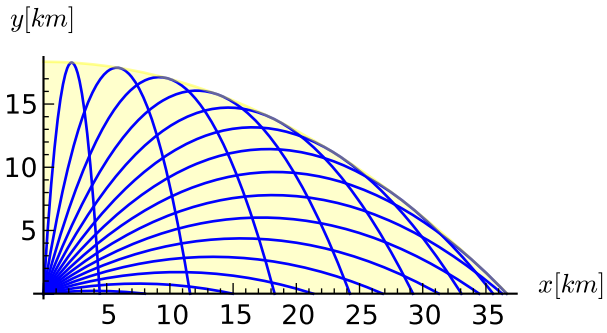
Równania (2) są tak zwanym parametrycznym przedstawieniem toru ruchu. Oznacza to, że znamy jak w zależności od parametru t (czyli czasu) zmienia się każda ze współrzędnych x i y. Wyliczając czas t z pierwszego równania i wstawiając go do drugiego równania można przekształcić te dwa równania do postaci funkcyjnej $y = f(x)$. Otrzymamy wtedy równanie paraboli o ujemnym współczynniku przy wyrazie x^2. Jednak w programie Sage mamy bardzo wygodną procedurę do rysowania właśnie takich krzywych parametrycznych - mianowicie funkcję `parametric_plot`.

```
var('t')
x0,y0,vx0,vy0 = 0,0,10,10
g = 9.81
t_end = 2*vy0/g
p1 = parametric_plot(\
 [x0+vx0*t,y0+vy0*t-g*t^2/2],\
 (t,0,t_end)  )
p1.show()
```

Rysownie tych krzywych jest fajną zabawą! Wyobraźmy sobie, że armata ma funkcję działa przeciwlotniczego i pytamy się w jakim obszarze przestrzeni jej pociski mogą dosięgnąć wrogi samolot?



Narysujmy w Sage na jednym wykresie tory wystrzałów z prędkościami $v_{x0} = v_0\cos(\alpha)$ i $v_{y0} = v_0\sin(\alpha)$, dla różnych kątów α (to jest kąt pomiędzy poziomem i wektorem prędkości początkowej pocisku). Zachęcamy do samodzielnego napisania kodu, który powinien generować coś zbliżonego do rysunku (1). Brzeg obszaru skutecznego ognia działa przeciwlotniczego, jest matematycznie rzecz biorąc obwiednią rodziny krzywych. Nasz rysunek sugeruje, że ma on kształt



Rysunek 1: Obszar rażenia działa o prędkości początkowej pocisku $v_0 = 600m/s$.

paraboli - i dokładny rachunek pokazuje, że tak jest rzeczywiście (zob. [3]).

2 Rzut ukośny - inaczej

Czytelnik pewnie zastanawia się, czy przypadkiem nie chcemy wrobić go w przerobienie lekcji fizyki pod pretekstem rysowania wykresów funkcji na komputerze. Otóż nie! Rozwiązanie matematyczne przyda się nam do weryfikacji wyników rozważań nieco odmiennych niż te prezentowane zazwyczaj w szkole. Rozważania te umożliwią nam analizę toru lotu pocisku z uwzględnieniem bardziej realistycznych czynników takich jak tarcie, wiatr czy nawet rotacja pocisku.

W równaniu Newtona dla ruchu pocisku występuje przyśpieszenie. Oznaczmy przez t pewną chwilę czasu, a przez Δt mały przyrost czasu. Dla składowej poziomej przyspieszenia otrzymujemy relację

$$a_x = \frac{\Delta v_x}{\Delta t} = \frac{v_x(t+\Delta t) - v_x(t)}{\Delta t}. \tag{3}$$

Z prawa Newtona dla składowej x-owej mamy:

$$a_x = \frac{F_x}{m} \tag{4}$$

więc możemy napisać

$$\frac{v_x(t+\Delta t) - v_x(t)}{\Delta t} = \frac{F_x}{m}. \tag{5}$$

Załóżmy, że znamy prędkość $v_x(t)$ w chwili t. Chcielibyśmy obliczyć prędkość $v_x(t+\Delta t)$ w chwili $t+\Delta t$. Z równania (5) wynika, że

$$v_x(t+\Delta t) = v_x(t) + \Delta t \cdot \frac{F_x}{m}. \tag{6}$$

Świetnie! Zapiszmy podobne wzory dla składowej pionowej i będziemy mogli obliczać prędkość w dowolnej chwili czasu. A co z położeniem? Możemy postąpić podobnie, stosując wzór na prędkość. I tak np. dla x:

$$v_x = \frac{x(t+\Delta t) - x(t)}{\Delta t} \tag{7}$$

czyli

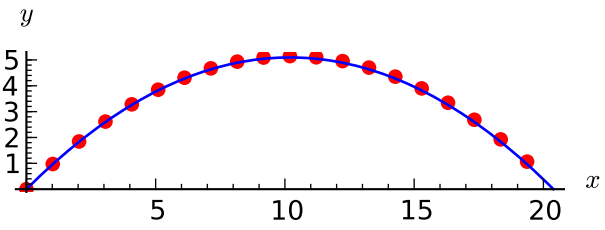
$$x(t+\Delta t) = x(t) + \Delta t \cdot v_x \tag{8}$$

Wzory (6) oraz (8) oraz odpowiedniki dla składowych pionowych y układają się w algorytm, który możemy zaprogramować na komputerze. Ale chwileczkę...

Czy te wzory są poprawne? Niestety nie! W rów. (6) założyliśmy prawo dla ruchu jednostajnie przyśpieszonego, a siła w ogólności nie musi być stała. W rów. (8) założyliśmy, że prędkość jest stała, a nie musi wcale tak być. No chyba, że Δt jest odpowiednio małe. Wtedy można by się spodziewać, że siła i prędkość w czasie między t a $t+\Delta t$ nie zmienią się. Wówczas wzory byłyby przybliżeniem prawdy. Sprawdźmy to eksperymentalnie wykonując poniższy kod w systemie Sage:

```
x0,y0,vx0,vy0 = 0,0,10,10
g = 9.81
t_end = 2*vy0/g
n = 200
tab = [[x0,y0]]
delta_t = t_end/n
for i in range(1,n):
    x  = x0 + vx0*delta_t
    y  = y0 + vy0*delta_t
    vx = vx0
    vy = vy0 - g*delta_t
    x0, y0, vx0, vy0 = [ x, y, vx, vy]
    tab.append([x0,y0])
p2 = list_plot(tab,figsize=4)
p2.show()
```

Podobnie jak w przypadku rysowania rozwiązania dokładnego startujemy z zadania warunków początkowych i parametrów układu (linie 1-3). Następnie zakładamy, że chcemy zastosować przybliżoną procedurę $n = 200$ razy w ciągu całego lotu pocisku. Wyliczamy $\Delta t = t_{end}/n$ (linia 6). Wykonujemy n razy pętlę w której korzystamy z czterech przybliżonych wzorów (6),(8) i analogicznych dla komponentu pionowego. Linia 12 jest niezwykle ważna bowiem "ustawia" wyliczone nowe wartości prędkości i położenia jako warunki początkowe dla kolejnego kroku. W linii 13 dołączamy wybrane parametry - akurat interesuje nas położenie - do listy punktów potrzebnych do późniejszego narysowania (dwie ostatnie linie) toru pocisku. Jeżeli w tej samej sesji Sage wykonaliśmy już pierwszy program, to możemy łatwo narysować rozwiązanie dokładne i przybliżone na jednym wykresie komendą: `(p1+p2).show()`.



Rysunek 2: Porównanie wyniku dokładnego i metody numerycznej.

Na rysunku 2 widzimy, że otrzymaliśmy tor ruchu zbliżony do dokładnego. Zachęcamy Czytelnika do samodzielnych eksperymentów i zbadania jak np. ilość iteracji - czyli krok czasowy - wpływa na wynik. Ciekawostką jest, że nasz program nigdzie nie zawierał funkcji kwadratowej, a pomimo tego narysował jej wykres - parabolę!

Po co robiliśmy tyle szumu i zaprzęgali komputer do obliczania tego co było i tak znane? Przecież w przypadku rzutu ukośnego metoda numeryczna z mozołem odtwarza wynik analityczny. Okazuje się, że nasz algorytm może być użyty do rozwiązywania praktycznie KAŻDEGO zagadnienia opisanego równaniem Newtona! Wystarczy zmodyfikować wyrażenia dla sił. Co więcej, siły mogą zależeć w najdziwniejszy sposób od każdej ze zmiennych. Wystarczy w naszym algorytmie zmienić zaledwie dwie linie:

```
vx = vx0 + Fx/m*delta_t
vy = vy0 + Fy/m*delta_t
```

gdzie za F_x, F_y wstawiamy odpowiednie wyrażenia na siły w kierunku poziomym i pionowym, które chcemy modelować. Na przykład możemy rozwiązać układ, w którym mamy realistyczną siłę oporu, która zależy kwadratowo od prędkości. Możemy dodać wiatr i to nawet taki, który wieje inaczej na wysokości 100m nad Ziemią, a inaczej na wysokości 10km. Możemy uwzględnić zmianę gęstości powietrza na dużych wysokościach. W takich przypadkach nie jest łatwo lub wręcz się nie da otrzymać rozwiązania metodami analitycznymi.

Cierpliwych Czytelników zapraszamy do lektury następnej części tego artykułu. Niecierpliwych zachęcamy do samodzielnego eksperymentowania od zaraz.

3 A równania różniczkowe?

Czyżby dopadła nas cyfrowa demencja - przecież mieliśmy się dowiedzieć czegoś o równaniach różniczkowych! Okazuje się, że nasz drugi program tak na prawdę był schematem Eulera rozwiązującym układ czterech równań różniczkowych zwyczajnych! Skoro potrafimy już je rozwiązywać, to może dowiedzmy się co to jest? Mówiąc mniej precyzyjnie, jest to równanie podobne do (5), ale w granicy $\Delta t \to 0$. Lewa strona przechodzi w wielkość zwaną pochodną (w tym przypadku pochodną prędkości). Równania różniczkowe to właśnie takie równania, które zawierają funkcje (w naszym przypadku $x(t), v_x(t), y(t), v_y(t)$, i ich pochodne. Przybliżając granicę przez wzięcie pewnego małego, ale skończonego Δt otrzymaliśmy właśnie schemat numeryczny rozwiązujący nasze równania różniczkowe.

Literatura

[1] http://sagemath.org

[2] http://cloud.sagemath.com

[3] http://visual.icse.us.edu.pl/Warsztaty

[4] http://pl.wikipedia.org/wiki/Rzut_ukośny