Rozdział 3

Fizyka laboratorium 4

3.1. Punkt materialny

Każdy proces przygotowania symulacji komputerowej rozpoczynamy od zdefiniowania modelu fizycznego, który przekształcamy w model matematyczny i następnie w model numeryczny.

Ruch swobodnego punktu materialnego jest opisany zasadami dynamiki Newtona: pierwsza pochodna wektora pędu punktu materialnego względem czasu jest równa przyłożonej do niego sile $\mathbf{F} = \frac{d\mathbf{p}}{dt}$, lub w przypadku gdy $m = \text{const}$ w postaci: $\mathbf{F} = m\mathbf{a}$, czyli

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{\mathbf{F}}{m}\,. \tag{3.1}$$

Mamy do czynienia z równaniem różniczkowym zwyczajnym drugiego rzędu, które można rozwiązać numerycznie w prosty sposób stosując metodę Eulera, bądź bardziej dokładną metodę Rungego-Kutty.

Ponieważ zarówno metoda Eulera, MidPoint jak i Rungego-Kutty, służą do rozwiązywania równań różniczkowych pierwszego rzędu, musimy w pierwszym kroku sprowadzić równanie do układu równań pierwszego rzędu.

$$\begin{cases} \dfrac{d\mathbf{r}}{dt} = \mathbf{v}\,, \\ \dfrac{d\mathbf{v}}{dt} = \dfrac{\mathbf{F}}{m}\,. \end{cases} \tag{3.2}$$

naszym celem jest wyznaczenie wartości funkcji $\mathbf{r}$ oraz $\mathbf{v}$ w kolejnym kroku czasowym na podstawie znajomości ich wartości w chwili czasowej t oraz znajomości wartości ich pochodnych.

Stosując metodę `Eulera` rozwiązanie układu równań możemy zapisać w następujący sposób:

$$\begin{aligned} \mathbf{r}(t+h) &= \mathbf{r}(t) + \mathbf{v}(t) \cdot h \\ \mathbf{v}(t+h) &= \mathbf{v}(t) + \frac{\mathbf{F}(t)}{m} \cdot h. \end{aligned} \tag{3.3}$$

Przykładowa procedura rozwiązująca układ równań metodą `Eulera`

```
void solveEuler(Punkt *p, float dt){
    p->v = p->v + p->f * (1.0/p->m) * dt;
    p->r = p->r + p->v * dt;
}
```

Gdzie `Punkt` to pewna struktura zawierająca informacje o punkcie materialnym, znaczenie jej składowych jest następujące:

- **v** - składowa typu `Wektor` reprezentująca wektor prędkości,
- **r** - składowa typu `Wektor` reprezentująca wektor położenia,
- **f** - składowa typu `Wektor` reprezentująca wektor wypadkowych sił działających na punk,
- m - składowa typu `float` skalar reprezentujący masę punktu.

Dla wygody i ułatwienia dalszej rozbudowy modelu punktu materialnego możemy napisać procedurę wyznaczającą wartości pochodnych funkcji **r** oraz **v**

```
void derivatives(Wektor *in, Wektor *out, Punkt *p){
    // dr/dt =v
    // d2r/dt2=F/m
    out[0] = in[1];
    out[1] = p->f * (1.0/p->m);
}
```

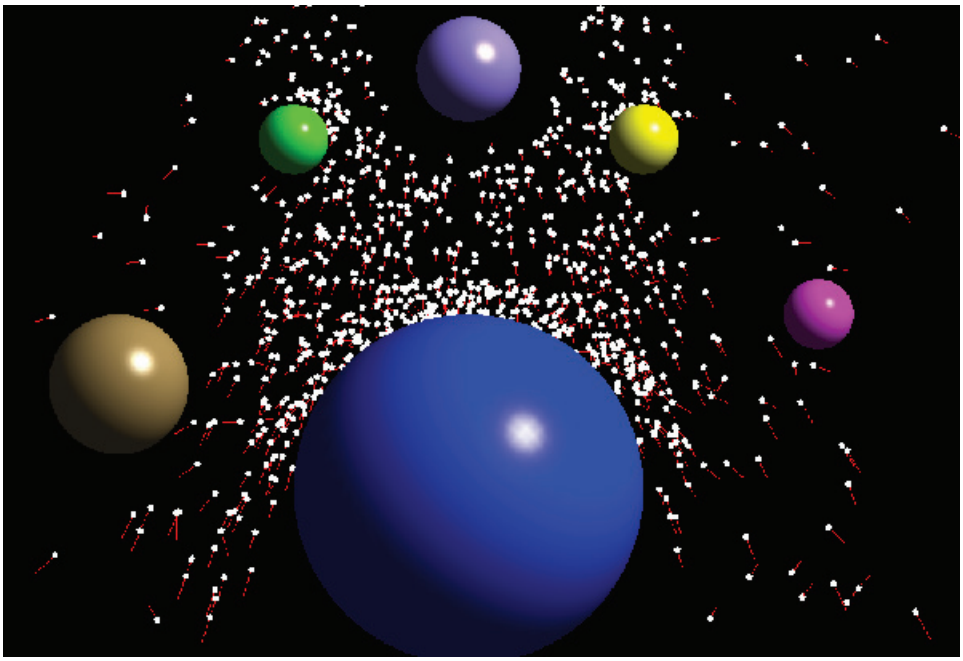
Jej działanie jest dość intuicyjne, na wejściu podajemy aktualny stan systemu a na wyjściu (zmienna out) dostajemy wartości pochodnych.

$$\begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix}' \rightarrow \begin{bmatrix} \mathbf{v} \\ \frac{\mathbf{F}}{m} \end{bmatrix}$$

Możemy procedurę `solveEuler` zapisać wykorzystując procedurę `derivatives` w następujący sposób:

```
void solveEuler(Punkt *p, float dt){
    Wektor in[2], out[2];
    in[0] = p->r;
    in[1] = p->v;
    derivatives(in,out,p);
    p->v = p->v + out[1] * dt;
    p->r = p->r + out[0] * dt;
}
```

Możemy zaimplementować przedstawiony model, przeprowadzając symulację ruchu większej ilości punktów materialnych.



Rysunek 3.1. Symulacja ruchu punktów materialnych

Przedstawiony model można wzbogacić o kolejne siły, dodajmy opór ośrodka. Siła oporu jest proporcjonalna do prędkości z jaką porusza się obiekt.

$$\mathbf{F} = \mathbf{F}_g + \mathbf{F}_t \tag{3.4}$$

gdzie $\mathbf{F}_g = m\mathbf{g}$, natomiast $\mathbf{F}_t = -c\mathbf{v}$, gdzie c to stała oporu ośrodka.

Należy dodać do kodu programu procedurę wyznaczającą wypadkową siłę działającą na punkt w chwili czasowej t.

```
void calcForce(Punkt *p, Wektor v){
    p->f = p->m * g - C * v;
}
```

Modyfikacji wymaga również procedura `solveEuler`, należy umieścić wywołanie procedur `calcForce`.

```
void solveEuler(Punkt *p, float dt){
    Wektor in[2], out[2];
    in[0] = p->r;
    in[1] = p->v;
    calcForce(p, p->v);
    derivatives(in,out,p);
    p->v = p->v + out[1] * dt;
    p->r = p->r + out[0] * dt;
}
```

3.2. Rzut ukośny

Ciału umieszczonemu w jednorodnym polu grawitacyjnym o przyspieszeniu g nadajemy pewną prędkość początkowa V_0, wyrzucając je pod kątem α do poziomu. Równania opisujące współrzędne ciała w kartezjańskim układzie po upływie czasu t:

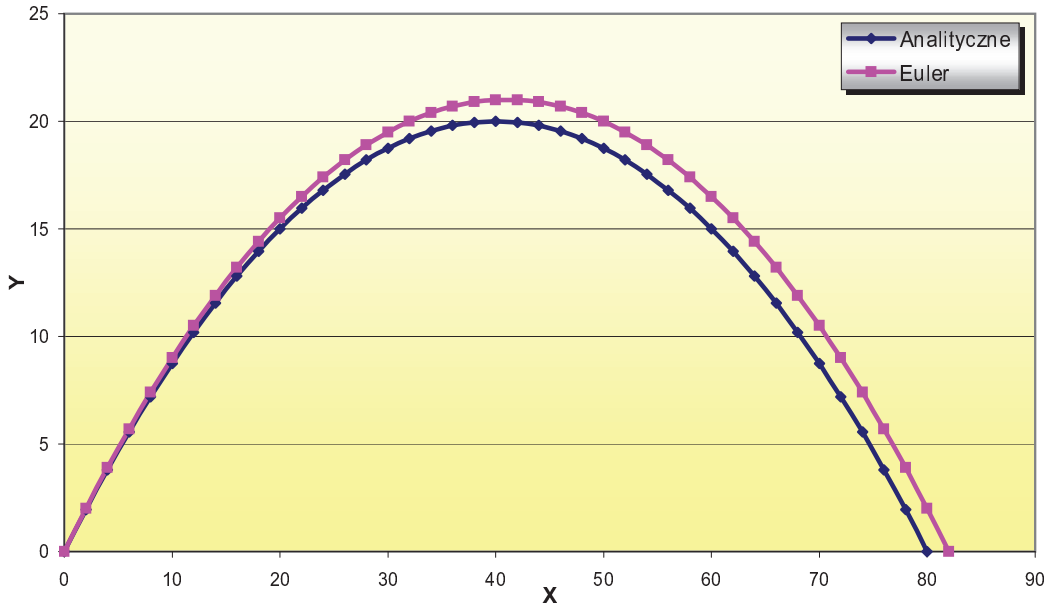
$$x = V_{0x} \cdot t \tag{3.5}$$

$$y = V_{0y} \cdot t - \frac{gt^2}{2} \tag{3.6}$$

gdzie, $V_{0x} = V_0 \cos(\alpha)$, oraz $V_{0y} = V_0 \sin(\alpha)$

Położenie i prędkość ciała możemy obliczyć rozwiązując numerycznie równanie ruchu dla punktu materialnego, umieszczając punkt w początku układu współrzędnych w chwili t_0 i nadając mu prędkość początkową.

Na wykresie przedstawiono rozwiązanie analityczne w porównaniu z rozwiązaniem numerycznym z wykorzystaniem najprostrzej metody `Eulera`.



Rysunek 3.2. Rzut ukośny

Oczywiście zastosowanie lepszej metody numerycznej powinno dać lepsze rozwiązanie. Zobaczmy jak wyglądałyby obliczenia dla metody `MidPoint`.

Rozpoczynamy w chwili t_0 dla której znamy położenie naszego obiektu $\mathbf{r}(t_0)$, oraz prędkość $\mathbf{v}(t_0)$. Korzystając z metody `MidPoint` w pierwszym kroku wyznaczamy $k1$ dla funkcji położenia $\mathbf{r}$ oraz prędkości $\mathbf{v}$:

$$k1 = \begin{bmatrix} \mathbf{r}(t_0) \\ \mathbf{v}(t_0) \end{bmatrix}' \rightarrow \begin{bmatrix} \mathbf{v}(t_0) \\ \mathbf{g}(t_0) \end{bmatrix}$$

Następnie obliczamy $k2$ zgodnie ze wzorem (2.20).

$$k2 = \begin{bmatrix} \mathbf{r}(t_0) + \frac{h}{2}k1[0] \\ \mathbf{v}(t_0) + \frac{h}{2}k1[1] \end{bmatrix}' \rightarrow \begin{bmatrix} \mathbf{v}(t_0) + \frac{h}{2}k1[1] \\ \mathbf{g}(t_0) \end{bmatrix}$$

Dalej wyznaczamy wartości funkcji $\mathbf{r}(t_0 + h)$ oraz $\mathbf{v}(t_0 + h)$, korzystając z formuły (2.21):

$$\begin{bmatrix} \mathbf{r}(t_0 + h) \\ \mathbf{v}(t_0 + h) \end{bmatrix} = \begin{bmatrix} \mathbf{r}(t_0) + \left(\mathbf{v}(t_0) + \frac{h}{2}\mathbf{g}(t_0)\right) h \\ \mathbf{v}(t_0) + \mathbf{g}(t_0)h \end{bmatrix}$$

Po przekształceniach otrzymamy następującą formułę opisującą wartość funkcji $\mathbf{r}$ w chwil $t_0 + h$:

$$\mathbf{r}(t_0 + h) = \mathbf{r}(t_0) + \mathbf{v}(t_0)h + \frac{\mathbf{g}h^2}{2}$$

Pamiętając, że $\mathbf{r}$ jest wektorem, oraz, że $\mathbf{g}$ wektor przyspieszenia grawitacyjnego jest stały oraz skierowany pionowo w dół możemy uzyskać położenie punktu względem osi x oraz y:

$$r_x(t_0 + h) = r_x(t_0) + v_x(t_0)h \tag{3.7}$$

$$r_y(t_0 + h) = r_y(t_0) + v_y(t_0)h + \frac{g_y h^2}{2} \tag{3.8}$$

Widać, że otrzymane równania są równoważne rozwiązaniu analitycznemu (3.5) oraz (3.6).

Możemy łatwo wyprowadzić wzory opisujące położenie punktu w chwili t rozpoczynając od punktu startowego $(0, 0)$, rozwijając funkcję $\mathbf{r}$ opisującą położenie punktu w przestrzeni w szereg Taylora:

$$\mathbf{r}(t_0 + h) = \mathbf{r}(t_0) + \mathbf{r}'(t_0)h + \frac{\mathbf{r}''(t_0)}{2!}h^2 + \dots \tag{3.9}$$

Ponieważ: $\mathbf{r}'(t_0) = \mathbf{v}(t_0)$ a $\mathbf{r}''(t_0) = \mathbf{v}'(t_0) = \mathbf{g}(t_0)$ oraz korzystając z faktu, że przyspieszenie $\mathbf{g}(x) = const$ czyli $\mathbf{g}' = 0$ otrzymujemy:

$$\mathbf{r}(t_0 + h) = \mathbf{r}(t_0) + \mathbf{v}(t_0)h + \frac{\mathbf{g}h^2}{2} \tag{3.10}$$

Kiedy w modelu uwzględnimy dodatkowe siły poza samą siłą grawitacji rozwiązanie zadania metodą numeryczną praktyczni nie ulega zmianie, jednak widać ,że metoda `MidPoint` nie da już rozwiązania dokładnego, ponieważ przy zmieniającym się w czasie przyspieszeniu kolejne pochodne nie będą się już zerowały.

3.3. Zadanie

Plik `zad3.rar` zawiera projekt dla środowiska `MinGW Developer Studio`, w pliku `main.cpp` należy dokonać następujących zmian:

1. Zaimplementować metodę `solveMidPoint`
2. Napisać procedurę `calcForce` uwzględniającą opór ośrodka w którym porusza się punk materialny.
3. Zmodyfikować program tak aby metody `solveEuler`, `solveMidPoint` poprawnie wyznaczały wypadkową siłę działającą na punkt, oraz zaimplementować procedurę `solveRK4`

Po uruchomieniu programu możemy przełączyć się na odpowiedni solver korzystając z następujących klawiszy:

- e - `solveEuler`
- m - `solveMidPoint`
- r - `solveRK4`