Dokumentacja - projekt podwójnego wahadła, MATLAB

Zuzanna Bożek, Aleksandra Chenczke, Paweł Poręba, Marta Wleklińska

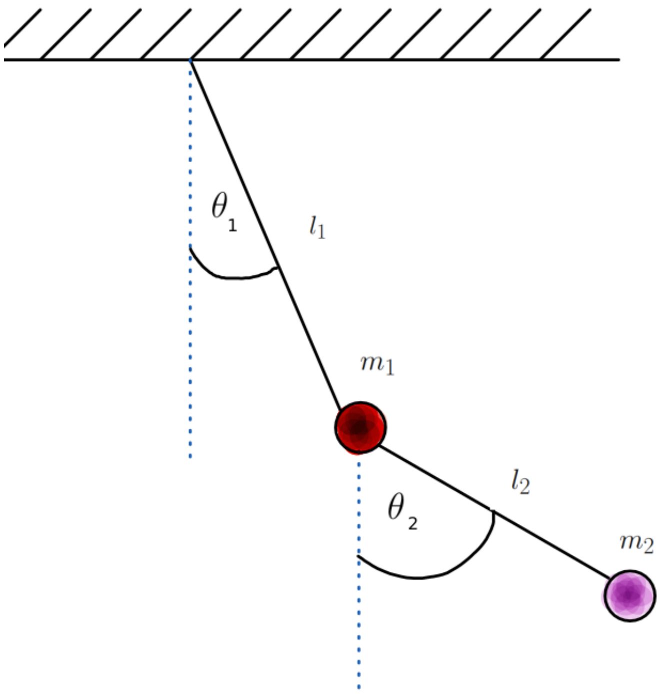
styczeń 2023

1 Cel projektu

Celem projektu, który został opisany poniżej, było stworzenie animacji ukazującej ruch podwójnego wahadła przy dowolnych parametrach początkowych (kątów wychylenia obu kulek, ich mas, długości nitek oraz natężenia grawitacyjnego).

2 Fizyka podwójnego wahadła

Podwójne wahadło jest przykładem systemu, w którym najmniejsza zmiana aprametrów może diametralnie zmienić jego zachowanie, czyli jest powiązany z pojęciem *chaosu deterministycznego*.
Taki układ składa się z kulki o masie m_1, która jest przymocowana nitką do stałej powierzchni i drugiej kulki o masie m_2 połączonej nitką z pierwszą kulką. Nitki mają długości kolejno l_1, l_2. Schemat takiego układu przybliża rysunek 1.



Rysunek 1: Schemat budowy wahadła
m_1, m_2 reprezentują wartości mas poszczególnych kulek, l_1, l_2 - długości kolejnych nici, a θ_1, θ_2 - kąty nachylenia nitek od normalnej

Jak większość problemów fizycznych, jest wiele sposobów na rozwiązanie problemu równania ruchu takiego wahadła w 2D. Najbardziej popularnym jest rozwiązanie za pomocą funkcji Lagrange'a L. Należy wprowadzić równania na współrzędne x_1, x_2, y_1, y_2 jako współrzędne pierwszej kulki (1) oraz drugiej (2):

$$
\begin{aligned}
x_1 &= l_1 sin(\theta_1),\\
x_2 &= l_1 sin(\theta_1),\\
y_1 &= -l_1 cos(\theta_1),\\
y_2 &= -l_1 cos(\theta_1) - l_2 cos(\theta_2).
\end{aligned}
$$

Następnie należy zdefiniować energię potencjalną układu, U:

$$U = m_1 g y_1 + m_2 g y_2 = -(m_1 + m_2) g l_1 cos\theta_1 - m_2 g l cos\theta_2.$$

Energię kinetyczną opisuje zależność:

$$\frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 = \frac{1}{2}m_1 l_1^2 \dot{\theta_1}^2 + \frac{1}{2}m_2(l_1^2\dot{\theta_1} +^2 l_2\dot{\theta_2}^2 + 2l_1 l_2 \dot{\theta_1}\dot{\theta_2} cos(\theta_1 - \theta_2)),$$

gdzie oznaczenie kropki nad zmienną reprezentuje jej pochodną po czasie.
Zatem lagranżian, który jest zdefiniowany jako różnica energii kinetycznej i potencjalnej układu, opisuje zależność:

$$L \equiv T - U = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta_1}^2 + 2m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2} cos(\theta_1 - \theta_2) + \frac{1}{2} m_2 l_2^2 \dot{\theta_2}^2 + (m_1 + m_2) g l_1 cos\theta_1 + m_2 g l_2 cos\theta_2.$$

Żeby wyznaczyć pochodną lagranżianu po pochodnej θ_1 po czasie należy wyznaczyć:

$$\frac{\partial L}{\partial \dot{\theta_1}} = m_1 l_1^2 \dot{\theta_1} + m_2 l_1 l_2 \dot{\theta_2} cos(\theta_1 - \theta_2).$$

A następnie pochodną po czasie tego wyrażenia:

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_1}}\right) &= (m_1 + m_2) l_1^2 \ddot{\theta_1} + m_2 l_1 l_2 \ddot{\theta_2} cos(\theta_1 - \theta_2) - m_2 l_1 l_2 \dot{\theta_2}(\dot{\theta_1} - \dot{\theta_2}) sin(\theta_1 - \theta_2)\\
\frac{\partial L}{\partial \theta_1} &= -lg(m_1 + m_2) sin\theta_1 - m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2} sin(\theta_1 - \theta_2).
\end{aligned}
$$

Wykonaliśmy powyższe obliczenia, żeby móc zapisać równanie Eulera - Lagrange'a dla jednej z współrzędnych uogólnionych, θ_1. Równanie to opisuje trajektorię obiektu, który znajduje się w danym układzie. Zatem znając langranżian układu, jest się w stanie znaleźć położenie i prędkość, a właściwie pochodne uogólnione i ich czasowe pochodne. Równanie Eulera - Lagrange'a przyjmuje postać:

$$\frac{\partial L}{\partial q_\alpha} - \frac{d}{dt}(\frac{\partial L}{\partial \dot{q_\alpha}}) = 0,$$

przy α reprezentuje kolejną współrzędną uogólnioną.
Możemy, wobec tego, zapisać:

$$(m_1 + m_2) l_1^2 \ddot{\theta_1} + m_2 l_1 l_2 \ddot{\theta_2} cos(\theta_1 - \theta_2) + m_2 l_1 l_2 \dot{\theta}_2^2 sin(\theta_1 - \theta_2) + l_1 g(m_1 + m_2) sin\theta_1 = 0.$$

Podzielenie powyższego wyrażenia przez l_1 prowadzi do zapisania zależności:

$$(m_1 + m_2) l_1 \ddot{\theta_1} + m_2 l_2 \ddot{\theta_2} cos(\theta_1 - \theta_2) + m_2 l_2 \dot{\theta}_2^2 sin(\theta_1 - \theta_2) + g(m_1 + m_2) sin\theta_1 = 0.$$

Analagicznie postępujemy dla kąta θ_2 uzyskując równania Eulera - Lagrange'a:

$$m_2 l_2 \ddot{\theta_2} + m_2 l_1 l_2 \ddot{\theta_1} cos(\theta_1 - \theta_2) - m_2 l_1 \dot{\theta}_1^2 sin(\theta_1 - \theta_2) + m_2 g sin\theta_2 = 0.$$

Stosując formalizm Hamiltona możemy zapisać zależności na składowe pędu:

$$p_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2) l_1^2 \dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 cos(\theta_1 - \theta_2),$$
$$p_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = m_2 l_2^2 \dot{\theta}_2 + m_2 l_1 l_2 \dot{\theta}_1 cos(\theta_1 - \theta_2).$$

Hamiltonian układu przyjmuje postać

$$H = \theta_i p_i - L = \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta_2}^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 cos(\theta_1 - \theta_2) - (m_1 + m_2) g l_1 cos\theta_1 - m_2 g l_2 cos\theta_2.$$

Przekształcenie powyższych zależności skutkuje napisaniu poniższych równości

$$\dot{\theta}_1 = \frac{\partial H}{\partial p_{\theta_1}} = \frac{l_2 p_{\theta_1} - l_1 p_{\theta_2} cos(\theta_1 - \theta_2)}{l_1^2 l_2 [m_1 + m_2 sin^2(\theta_1 - \theta_2)]},$$
$$\dot{\theta}_2 = \frac{\partial H}{\partial p_{\theta_2}} = \frac{l_1(m_1 + m_2) p_{\theta_2} - l_2 m_2 p_{\theta_1} cos(\theta_1 - \theta_2)}{l_1 l_2^2 m_2 [m_1 + m_2 sin^2(\theta_1 - \theta_2)]},$$
$$\dot{p_{\theta_1}} = -\frac{\partial H}{\partial \theta_1} = -(m_1 + m_2) g l_1 sin\theta_1 - C_1 + C_2,$$
$$\dot{p_{\theta_2}} = -\frac{\partial H}{\partial \theta_2} = -m_2 g l_2 sin\theta_2 + C_1 - C_2,$$

gdzie

$$C_1 \equiv \frac{p_{\theta_1} p_{\theta_2} sin(\theta_1 - \theta_2)}{l_1 l_2 [m_1 + m_2 sin^2(\theta_1 - \theta_2)]},$$
$$C_2 \equiv \frac{l_2^2 m_2 p_{\theta_1}^2 + l_1^2 (m_1 + m_2) p_{\theta_2}^2 - l_1 l_2 m_2 p_{\theta_1} p_{\theta_2} cos(\theta_1 - \theta_2)}{2 l_1^2 l_2^2 [m_1 + m_2 sin^2(\theta_1 - \theta_2)]^2} sin[2(\theta_1 - \theta_2)].$$

Dodanie do takich równań warunków początkowych umożliwia zapisanie równania ruchu wahadła.

3 Rozwiązanie problemu

W poprzedniej sekcji ukazane zostało rozwiązanie problemu znalezienia równania ruchu dla podwójnego wahadła. My jednak w projekcie zdecydowaliśmy się na inne rozwiązanie, które opiszemy w tej sekcji.
Należało zacząć od zdefiniowania zmiennych oraz zależności między nimi (np. zależności prędkości kulek i ich położenia).
Po definicji odpowiednich zmiennych i parametrów, posłużyliśmy się II zasadą dynamiki Newtona, żeby zapisać równania opisujące siły naprężenia działające w układzie, `tension_1` oraz `tension_2`:

```
            syms tension_1 tension_2;

            differental_equation_x_1 = mass_1*acceleration_x_1(t) == -tension_1*sin(angle_1(t)) + tension_2*sin(angle_2(t));
```

```
            differental_equation_y_1 = mass_1*acceleration_y_1(t) == tension_1*cos(angle_1(t)) - tension_2*cos(angle_2(t)) - mass_1*gravity;

            differental_equation_x_2 = mass_2*acceleration_x_2(t) == -tension_2*sin(angle_2(t));
            differental_equation_y_2 = mass_2*acceleration_y_2(t) == tension_2*cos(angle_2(t)) - mass_2*gravity;

            overall_tension = solve([differental_equation_x_1 differental_equation_y_1],[tension_1 tension_2]);
```

Poniżej zapisaliśmy równania różniczkowe ze względu na wartości sił naprężenia `tension_2`, `tension_1`:

```
            general_differental_equation_1 = subs(differental_equation_x_2,[tension_1 tension_2], [overall_tension.tension_1, overall_tension.tension_2]);
            general_differental_equation_2 = subs(differental_equation_y_2,[tension_1 tension_2], [overall_tension.tension_1, overall_tension.tension_2]);

            substituted_equation_1 = subs(general_differental_equation_1, [length_1, length_2, mass_1, mass_2, gravity], [l1, l2, m1, m2, g]);
            substituted_equation_2 = subs(general_differental_equation_2, [length_1, length_2, mass_1, mass_2, gravity], [l1, l2, m1, m2, g]);
```

Następnie zdefiniowaliśmy rozwiązania powyższych równań posługując się funkcją `odeToVectorField`, która odpowiada macierzy, której elementami są powyższe niejednorodne równania różniczkowe.
Poniżej, funkcja `matlabFunction` konwertuje macierz `V` w funkcję dwóch argumentów: czasu `t` oraz wektora `Y`, czyli zmiennych z poprzedniej funkcji.
Kolejna funkcja `ode45` rozwiązuje `M` w przedziale `[0 max_t]`, przy warunkach początkowych `[a1 0 a2 0]`, czyli amplitudach obu kątów `a1, a2` oraz czasie początkowym 0 dla obu kątów:

```
            V = odeToVectorField(substituted_equation_1, substituted_equation_2);
            M = matlabFunction(V,'vars',{'t','Y'});

            modified_object.solutions = ode45(M,[0 max_t], [a1 0 a2 0]);
```

Kluczowym krokiem było opisanie konstruktora, który tworzy obiekt (opis w komentarzu w kodzie) i przekazuje mu parametry widniejące nawiasach (`a1, a2, m1, m2, l1, l2, g, max_t` = kolejno dwa kąty odchylenia, poszczególne masy kulek, kolejne długości nici oraz maksymalny czas animacji):

```
    methods
        % constructor, as a parameter takes first and second ball initial angles, masses and lengths, the gravity and the animation duration
        function obj = Pendulum(a1, a2, m1, m2, l1, l2, g, max_t)
```

```
            obj = obj.solve_equations(a1, a2, m1, m2, l1, l2, g, max_t);
        end
```

Dzięki temu byliśmy w stanie wprowadzić funkcję `get_first_ball_coordinates` oraz `get_second_ball_coordinates`, które generują współrzędne kolejnych kulek za pomocą funkcji `deval`, która zwraca odpowiedni komponent rozwiązania w odpowiedniej chwili `t`:

```
        % a function that returns the first ball coordinates within given time
        function coords = get_first_ball_coordinates(self, t)
            coords = [self.length_first*sin(deval(self.solutions, t, 3)), -self.length_first*cos(deval(self.solutions, t, 3))];
        end

        % a function that returns the second ball coordinates within given time
        function coords = get_second_ball_coordinates(self, t)
            first_ball_coordinates = self.get_first_ball_coordinates(t);
            coords = [first_ball_coordinates(1) + self.length_second*sin(deval(self.solutions, t, 1)), first_ball_coordinates(2) - self.length_second*cos(deval(self.solutions, t, 1))];
        end
```

Ostatnim krokiem przed wykonaniem animacji było "wyciągnięcie" danych parametrów z obiektu, żeby można było wprowadzić później ich konkretne wartości:

```
        % a function that lets us change the pendulum parameters - takes identical parameters as the constructor
        function modified_object = change_values(self, a1, a2, m1, m2, l1, l2, g, max_t)
            modified_object = self;
            modified_object = modified_object.solve_equations( a1, a2, m1, m2, l1, l2, g, max_t);
        end

        % a function that returns the pendulum parameters
        function values = get_values(self)
            values = [self.length_first, self.length_second, self.mass_first, self.mass_second];
        end

        % a function that returns the animation duration
        function time = get_max_time(self)
            time = self.max_time;
        end
```

Przechodząc już do opisywania animacji zapisaliśmy, jak chcemy, żeby animacja wyglądała - rozwiązane współrzędne kulek, przedziały osi etc.

```
        for t = 0:.1:self.pendulum.get_max_time()
            first_coordinates = self.pendulum.get_first_ball_coordinates(t);
            second_coordinates = self.pendulum.get_second_ball_coordinates(t);
            x_1 = first_coordinates(1);
            y_1 = first_coordinates(2);
            x_2 = second_coordinates(1);
            y_2 = second_coordinates(2);
            plot(x_1,y_1,'ro','MarkerSize',m_1*7,'MarkerFaceColor','r');
            hold on;
            plot([0 x_1],[0 y_1],'r-');
            plot(x_2,y_2,'go','MarkerSize',m_2*10,'MarkerFaceColor','g');
            plot([x_1 x_2],[y_1 y_2],'g-');
            text(-0.3,0.3,"Timer: "+num2str(t,2));
            xlim([-L_1-L_2-1,L_1+L_2+1]);
            ylim([-L_1-L_2-1,L_1+L_2+1]);
            hold off;
            temp_frames(round(t*10)+1) = getframe;
        end
```

Na koniec chcieliśmy pozwolić użytkownikowi, żeby wybierał początkowe warunki układu, jak kąty wychyelnia obu kulek, ich masy, długości nici oraz natężeniowi pola grawitacyjnego, czyli na Ziemi, przyspieszenia ziemskiego g.

4 Podsumowanie i kontrybucje

Celem naszego projektu było zaprojektowanie układu podwójnego wahadła, który porusza się po chaotycznej trajektorii kulek. Chcieliśmy również, żeby użytkownik mógł dobierać parametry dla układu, co się udało i działa wszystko poprawnie.
Zuzanna Bożek i Paweł Poręba zajęli się zapisaniem kodu, który rozwiązuje problem znalezienia równania ruchu dla układu, Aleksandra Chenczke podjęła się stworzenia graficznego interfejsu użytkownika, a Marta Wleklińska spisała dokumentację.