Metody Obliczeniowe Fizyki

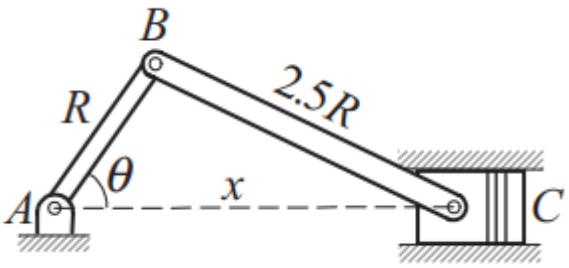
Projekt w języku Python

Różniczkowanie numeryczne, Opisanie przyspieszenia tłoka

Rozdział 5, zad. 12, “Numerical methods in enginering with Python 3” - Jaan Kiusalaas

1.Wstęp i opis doświadczenia

W tym doświadczeniu będziemy opisywać przyspieszenie tłoka, przedstawionego poniżej:



rys. 1

Opiszemy przyspieszenie tłoka dla odchylenia korby od 0 stopni do 180 stopni dla teta względem osi x, przyjmując odstępy między punktami pomiarowymi co 5 stopni, t.j. (0, 5, 10, … , 180) stopni. Oś obrotu znajduje się w punkcie A. Kąt teta opisany jest pomiędzy osią x, a drążkiem AB o długości R=90mm. Drążek BC ma długość równą 2,5R. Korba porusza się ze stałą prędkością kątową równą $\frac{d\theta}{dt} = 5000\ \frac{obr}{min}$.

Pozycja tłoka na osi x opisana jest wzorem:

$$x = R\left(cos\theta\ + \sqrt{2{,}5^2 - \sin^2\ \ \theta}\right) \qquad (1.)$$

Do napisania programu użyjemy edytora Visual Studio Code oraz języka programowania Python. Do napisania programu potrzebować będziemy biblioteki numpy, która umożliwi nam korzystanie z funkcji trygonometrycznych przy obliczeniach. Zaś biblioteka mathplotlib.pyplot pozwoli nam wyświetlić wykres funkcji.

2.Opracowanie

Rozpoczynamy od umieszczenia potrzebnych bibliotek w pamięci programu korzystając z komendy import:

```
1   import numpy as np
2   import matplotlib.pyplot as plt
```

Będziemy korzystać z zawartości bibliotek poprzez komendy typu np.cos() lub plt.xlabel().

Definiujemy następnie funkcję x zależną od kąta teta korzystając ze wzoru nr.1 podanego powyżej:

```
4   def x(teta):
5       return 90 * (np.cos(teta) + np.sqrt(2.5**2 - np.sin(teta)**2))
```

Korzystamy przy tym z wywołania funkcji obliczającej wartości trygonometryczne sinus, cosinus oraz pierwiastek od kąta teta za pomocą np.sin(), np.cos() oraz np.sqrt().

Definiujemy funkcję przedstawiającą różniczkę numeryczną i korzystamy z wzoru:

$$\frac{f(x+h) - f(x-h)}{2h}$$

```
7   def rozniczka_numeryczna(teta):
8       delta_teta = 1e-3
9       x_p = x(teta + delta_teta)
10      x_m = x(teta - delta_teta)
11      v_x = (x_p - x_m) / (2 * delta_teta)
12
13      return v_x
```

Ta funkcja zależeć będzie od kąta teta i definiujemy w niej delta_teta będąca małym epsilonem dla różniczki. W tym przypadku 1e-3 jest równe 0.001 . W dziewiątej i dziesiątej linijce kodu obliczamy wartości dla f(x+h) i odpowiednio f(x-h) z wzoru. Za to v_x jest naszą prędkością obliczoną za pomocą różniczki numerycznej, którą zwraca ta funkcja.

Definiujemy funkcję obliczającą przyspieszenie za pomocą opisanej wcześniej funkcji różniczki numerycznej:

```
15  def przysp_numeryczne(teta):
16      delta_teta = 1e-3
17      v_x_p = rozniczka_numeryczna(teta + delta_teta)
18      v_x_m = rozniczka_numeryczna(teta - delta_teta)
19      a_x = (v_x_p - v_x_m) / (2 * delta_teta)
20      return a_x
```

Funkcja przysp_numeryczne korzysta z poprzedniej funkcji prędkości numerycznej i oblicza przyspieszenie numeryczne. Podobnie jak w poprzednim przypadku wybieramy mały epsilon będący delta_teta = 1e-3 = 0.001 .

Korzystamy następnie z funkcji z biblioteki numpy, która pozwoli nam zadać tecie konkretne wartości, oraz funkcji np.zeros_like() :

```
23  theta_values = np.linspace(0, 180, 36)
24  przysp_values = np.zeros_like(theta_values)
```

Przypisujemy tutaj dla theta_values wartości od 0 do 180, co 5 stopni. Korzystamy do tego z funkcji np.linspace(), gdzie trzecią podaną wartością jest podział przedziału na wybraną ilość punktów pomiarowych. Ze względu na to, że odstępy pomiędzy pomiarami mają mieć 5 stopni, potrzebujemy 36 punktów pomiarowych.

Dla przysp_values przypisujemy wartości theta_values, używając np.zeros_like(), aby wartości dla każdego punktu pomiarowego były równe 0.

Tworzymy następnie pętlę, która korzystając z wcześniej zdefiniowanej funkcji przysp_numeryczne przypisze wartości dla przysp_values zależnego od zmiennej i :

```
26  for i, theta in enumerate(theta_values):
27      przysp_values[i] = przysp_numeryczne(np.deg2rad(theta))
```

W pętli for wpisujemy dwie zmienne, i oraz theta. Skorzystamy z funkcji enumerate, która będzie zapamiętywała ilość wykonanych pętli. Przy wprowadzeniu do funkcji przysp_numeryczne wartości tety użyjemy funkcji np.deg2rad(), aby przeliczyć stopnie na radiany.

Sporządzimy teraz wykres, który przedstawi zależność przyspieszenia od wychylenia kąta teta. Skorzystamy do tego z możliwości biblioteki matplotlib.pyplot :

```
29    plt.plot(teta_values, przysp_values)
30    plt.xlabel('Teta (deg)')
31    plt.ylabel('Przyspieszenie')
32    plt.title('Przyspieszenie tłoka(teta)')
33    plt.grid()
34    plt.show()
```

Funkcji plt.pyplot przypisujemy teta_values oraz przysp_values, co zwróci nam wykres tych wartości.

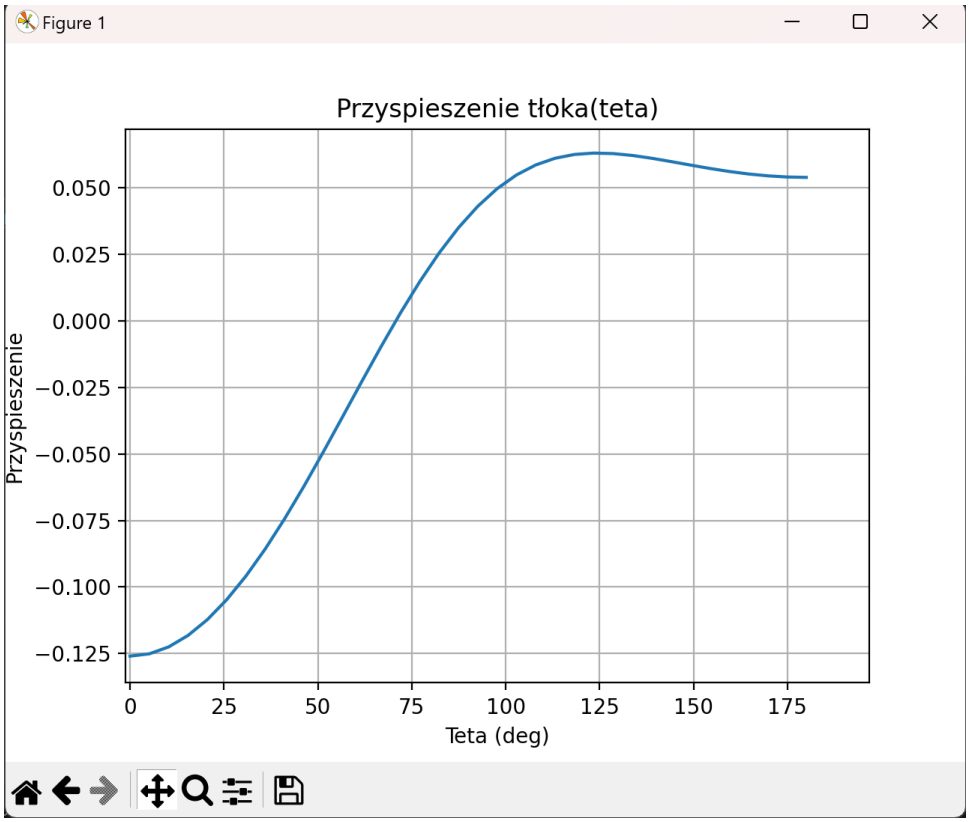
Funkcja plt.xlabel przypisuje nazwę osi x, a plt.ylabel przypisuje nazwę osi y.

Funkcja plt.title przypisuje wykresowi tytuł.

Funkcja plt.grid konfiguruje linie siatki.

Funkcja plt.show wyświetla skonfigurowany wykres.

Otrzymujemy w ten sposób wykres zależności przyspieszenia od kąta teta:



3. Wnioski

Zauważamy, że przyspieszenie tłoka na przedziale od 0 do 180 stopni przyjmuje największą wartość przy cofaniu się tłoka.

Pełna wersja kodu w wersji ostatecznej:

```python
import numpy as np
import matplotlib.pyplot as plt

def x(teta):
    return 0.09* (np.cos(teta) + np.sqrt(2.5**2 - np.sin(teta)**2))

def rozniczka_numeryczna(teta):
    delta_teta = 1e-3
    x_p = x(teta + delta_teta)
    x_m = x(teta - delta_teta)
    v_x = (x_p - x_m) / (2 * delta_teta)

    return v_x

def przysp_numeryczne(teta):
    delta_teta = 1e-3
    v_x_p = rozniczka_numeryczna(teta + delta_teta)
    v_x_m = rozniczka_numeryczna(teta - delta_teta)
    a_x = (v_x_p - v_x_m) / (2 * delta_teta)
    return a_x


teta_values = np.linspace(0, 180, 36)
przysp_values = np.zeros_like(teta_values)

for i, theta in enumerate(teta_values):
    przysp_values[i] = przysp_numeryczne(np.deg2rad(theta))

plt.plot(teta_values, przysp_values)
plt.xlabel('Teta (deg)')
plt.ylabel('Przyspieszenie')
plt.title('Przyspieszenie tłoka(teta)')
plt.grid()
plt.show()
```

Literatura: Wikipedia.pl, numpy.org, matplotlib.org.