Metody Obliczeniowe Fizyki

Projekt w języku Python

Problemy z wartością początkową, Czas spadku

Rozdział 7, zad. 8, “Numerical methods in enginering with Python 3” – Jaan Kiusalaas

Jakub Anzulewicz(281128)

1. Wstęp i opis doświadczenia

W tym doświadczeniu napiszemy program obliczający czas spadku skoczka w spadku swobodnym z określonej wysokości. Do napisania kodu użyjemy edytora kodu Visual Studio Code oraz języka Python.

Skoczek o masie m = 80 kg będzie spadał spadkiem swobodnym w pionie. Na jego ciało będzie działał opór aerodynamiczny, którego wzór wygląda następująco:

$$F_D = C_D \cdot \dot{y}^2$$

Współczynnik oporu aerodynamicznego C_D jest równy 0.2028 $\frac{kg}{m}$.Równanie różniczkowe opisujące spadek ma postać:

$$\frac{d\dot{y}}{dt} = g - \frac{C_D}{m} \cdot \dot{y}^2$$

Wartość przyspieszenia ziemskiego g przyjmiemy jako g = 9.80665 $\frac{m}{s^2}$.

Pisząc program skorzystamy z metody Rungego-Kutty rzędu czwartego do obliczenia równań różniczkowych.

2. Opracowanie

Do napisania programu potrzebować będziemy biblioteki numpy, którą zapiszemy w pamięci programu za pomocą komendy import. Przypiszemy też wartości zmiennym potrzebnym do obliczenia czasu spadku oraz opiszemy warunki początkowe:

```
1   import numpy as np
2
3   g = 9.80665
4   c = 0.2028
5   m = 80
6
7   polozenie0 = 0
8   predkosc0 = 0
9   czas0 = 0
```

Opiszemy następnie odstępy czasu po których będziemy w późniejszym kodzie używać w pętli “for”, oraz dostosujemy liczbę powtórzeń do wysokości skoku, czego użyjemy do stworzenia zerowego szeregu dla położenia, prędkości oraz czasu korzystając z komendy np.zeros():

```
11  dt = 0.1
12  num_iterations = int(5000 / dt)
13
14
15  polozenia = np.zeros(num_iterations + 1)
16  predkosci = np.zeros(num_iterations + 1)
17  czasy = np.zeros(num_iterations + 1)
```

Korzystamy z np.zeros() aby dla każdej iteracji zawartej w num_iterations utworzyć miejsce o wartości 0. Dodajemy jedynkę ze względu na to, że w języku Python numerowanie zaczyna się od 0.

Skorzystamy następnie z pętli “for” dla zmiennej “i” w zasięgu wartości num_iterations. Zdefiniujemy “predkosc” oraz “polozenie” poprzez nadanie im wartości szeregów odpowiednio predkosci[i] oraz polozenia[i]. Podobnie nadamy wartość dla “t”:

```
20    for i in range(num_iterations):
21        predkosc = predkosci[i]
22        polozenie = polozenia[i]
23        t = czasy[i]
```

Implementujemy do pętli metodę Rungego-Kutty czwartego rodzaju w następujący sposób:

```
20  for i in range(num_iterations):
21      predkosc = predkosci[i]
22      polozenie = polozenia[i]
23      t = czasy[i]
24
25      k1v = (g - (c / m) * predkosc**2) * dt
26      k1y = predkosc * dt
27
28      k2v = (g - (c / m) * (predkosc + 0.5 * k1v)**2) * dt
29      k2y = (predkosc + 0.5 * k1v) * dt
30
31      k3v = (g - (c / m) * (predkosc + 0.5 * k2v)**2) * dt
32      k3y = (predkosc + 0.5 * k2v) * dt
33
34      k4v = (g - (c / m) * (predkosc + k3v)**2) * dt
35      k4y = (predkosc + k3v) * dt
36
37      nowa_predkosc = predkosc + (k1v + 2 * k2v + 2 * k3v + k4v) / 6
38      nowa_pozycja = polozenie + (k1y + 2 * k2y + 2 * k3y + k4y) / 6
39
40      predkosci[i + 1] = nowa_predkosc
41      polozenia[i + 1] = nowa_pozycja
42      czasy[i + 1] = t + dt
```

Na końcu pętli zapisujemy wartości otrzymane z metody Rungego-Kutty w szeregach “predkosci” oraz “polozenia”. Inkrementujemy miejsca w pętli zapisując np. predkosci[i + 1]. Dzięki temu przy każdym zapętleniu program zapisze wartości w następnym miejscu szeregu oraz zostawi pierwszą wartość, która równa się 0. Tak samo inkrementujemy “polozenia” i “czasy”.

Podajemy wartość dla wysokości, oraz używamy komendy np.interp() do otrzymania funkcji interpolacyjnej w konkretnym punkcie. Dla nas będzie to moment, którym skoczek przebędzie drogę spadku równą 5000m. Korzystamy z komendy print() i jej właściwości aby wyświetlić otrzymaną wartość w terminalu:

```
45  wysokosc = 5000
46  czas_spadku = np.interp(wysokosc, polozenia, czasy)
47
48  print("Czas spadku z wysokości 5000m jest równy:", czas_spadku, "sekund")
```

3. Wnioski

Otrzymujemy wartość 84.7855611364419 sekundy. Po takim czasie skoczek wylądowałby na ziemi.

Pełny kod:

```python
import numpy as np

g = 9.80665
c = 0.2028
m = 80

polozenie0 = 0
predkosc0 = 0
czas0 = 0

dt = 0.1
num_iterations = int(5000 / dt)


polozenia = np.zeros(num_iterations + 1)
predkosci = np.zeros(num_iterations + 1)
czasy = np.zeros(num_iterations + 1)


for i in range(num_iterations):
    predkosc = predkosci[i]
    polozenie = polozenia[i]
    t = czasy[i]

    k1v = (g - (c / m) * predkosc**2) * dt
    k1y = predkosc * dt

    k2v = (g - (c / m) * (predkosc + 0.5 * k1v)**2) * dt
    k2y = (predkosc + 0.5 * k1v) * dt

    k3v = (g - (c / m) * (predkosc + 0.5 * k2v)**2) * dt
    k3y = (predkosc + 0.5 * k2v) * dt

    k4v = (g - (c / m) * (predkosc + k3v)**2) * dt
    k4y = (predkosc + k3v) * dt

    nowa_predkosc = predkosc + (k1v + 2 * k2v + 2 * k3v + k4v) / 6
    nowa_pozycja = polozenie + (k1y + 2 * k2y + 2 * k3y + k4y) / 6

    predkosci[i + 1] = nowa_predkosc
    polozenia[i + 1] = nowa_pozycja
    czasy[i + 1] = t + dt


wysokosc = 5000
czas_spadku = np.interp(wysokosc, polozenia, czasy)

print("Czas spadku z wysokości 5000m jest równy:", czas_spadku, "sekund")


```

Literatura:

-Wikipedia.pl
-numpy.org
- “Numerical methods in enginering with Python 3” – Jaan Kiusalaas