Set 3

Σεϊτανίδου Δήμητρα

23 Φεβρουαρίου 2020

Άσκηση 1

Στην πρώτη άσκηση θέλουμε να λύσουμε το ολοκλήρωμα

$$\int_0^3 xe^{2x}dx \tag{1}$$

με τη μέθοδο του Simpson 1/3. Θα χρησιμοποιήσουμε και την απλή μορφή, που εφαρμόζεται σε τρία σημεία, αλλά και την γενική μορφή που εφαρμόζεται σε πολλά σημεία αρκεί οι υποδιαιρέσεις του διαστήματος ολοκλήρωσης να είναι άρτιος αριθμός.

Η απλή μορφή δίνεται από την σχέση:

$$I = (b-a)\frac{f(a)+4f(x_1)+f(b)}{6} \tag{2}$$

όπου α και β είναι τα άκρα του διαστήματος ολοκλήρωσης και $x_1 = (b-a)/2$.

Η γενική μορφή δίνεται από τη σχέση:

$$I = (b-a)\frac{f(a)+4\sum_{i=1,3}^{n-1} f(x_i)+2\sum_{i=2,4}^{n-2} f(x_i)+f(b)}{3n} \tag{3}$$

όπου n ο αριθμός των υποδιαιρέσεων.

Εφαρμόζοντας τις σχέσεις (2) και (3) στο ολοκλήρωμα υπολογίζουμε την τιμή του:

- Για την απλή μέθοδο: $\mathbf{I_1 = 665.3998}$
- Για την γενική μέθοδο με $n = 8$: $\mathbf{I_2 = 506.0084}$

Αν λύσουμε αναλυτικά το ολοκλήρωμα βρίσκουμε ότι η πραγματική τιμή του είναι $I = 504.536$ και αν το συγκρίνουμε με τις τιμές που βρήκαμε αριθμητικά καταλήγουμε στο συμπέρασμα ότι όσο πιο πολλές υποδιαιρέσεις πάρουμε στο διάστημα ολοκλήρωσης τόσο πιο κοντά θα είμαστε στην πραγματική τιμή του ολοκληρώματος.

Ο κώδικας δίνεται παρακάτω.

```python
import numpy as np

f = lambda x: x*np.exp(2*x)

#integration limits
a = 0
b = 3

#Simple Simpson's 1/3 integration
h = (b-a)/2
I1 = (b-a)*(f(a)+4*f(a+h)+f(b))/6
print(I1)


#Simpson's 1/3 integration for n=8
n=8
h = (b-a)/n
integr = f(a)+f(b)
for i in range(1,n):
    if i%2 == 0:
        integr += 2*f(a+i*h)
    else:
        integr += 4*f(a+i*h)
I2 = integr*(h/3)
print(I2)
```

Άσκηση 2

Στην δεύτερη άσκηση καλούμαστε να λύσουμε την διαφορική εξίσωση

$$\frac{dy}{dt} = yt^2 - 1.1y \tag{4}$$

χρησιμοποιώντας την μέθοδο Euler και την μέθοδο Runge-Kutta τέταρτης τάξης.

Η μέθοδος Euler δίνεται από τον παρακάτω απλό αναδρομικό τύπο:

$$y_{i+1} = y_i + hf(t_i, y_i) \tag{5}$$

όπου $f(t, y) = dy/dt$ και $h = t_{i+1} - t_i$ το σταθερό βήμα.

Η μέθοδος Runge-Kutta τέταρτης τάξης δίνεται από τις παρακάτω σχέσεις:

$$y_{i+1} = y_i + h(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4) \tag{6}$$

$$k_1 = f(t_i, y_i) \tag{7}$$

$$k_2 = f(t_i + h/2, y_i + hk_1/2) \tag{8}$$

$$k_3 = f(t_i + h/2, y_i + hk_2/2) \tag{9}$$

$$k_4 = f(t_i + h, y_i + hk_3) \tag{10}$$

Η ακρίβεια της μεθόδου Euler είναι $O(h)$ ενώ της μεθόδου Runge-Kutta είναι $O(h^4)$.

Οι δύο παραπάνω μέθοδοι είναι πολύ εύκολο να προγραμματιστούν με έναν βρόχο επανάληψης, δίνοντας την αρχική συνθήκη πριν τον βρόχο. Τρέχουμε τον κώδικα δύο φορές, μια για $h = 0.5$ και μία για $h = 0.25$, στο διάστημα $t \in [0, 2]$ και βρίσκουμε τα εξής:

- Μέθοδος Euler : για $h = 0.5$, $\quad y(2) = 0.9485$
  για $h = 0.25$, $\quad y(2) = 1.3181$

- Μέθοδος Runge-Kutta : για $h = 0.5$, $\quad y(2) = 10.9363$
  για $h = 0.25$, $\quad y(2) = 3.7404$

Η πραγματική τιμή της συνάρτησης είναι $y(2) = 1.5947$. Από τις αριθμητικές λύσεις φαίνεται ότι όσο μικραίνουμε το βήμα πλησιάζουμε την πραγματική τιμή αλλά πρέπει να το μικρύνουμε κι άλλο για να έχουμε ικανοποιητική ακρίβεια.

Ο κώδικας δίνεται παρακάτω.

```python
import numpy as np
import math as m

f = lambda t,y: y*m.pow(t,2)-1.1*y
y0 = 1
t0=0
#h = 0.5
h = 0.25

#Euler's method
y = y0+h*f(t0,y0)
for i in np.arange(t0+h,2+h,h):
    ynew = y+h*f(i,y)
    y = ynew
print(y)


#4rth order Runge-Kutta
k1 = f(t0,y0)
k2 = f(t0+h/2,y0+h/2*k1)
k3 = f(t0+h/2,y0+h/2*k2)
k4 = f(t0+h,y0+h*k3)
y = y0+h*(1/6*k1+1/3*k2+1/3*k3+1/6*k4)
for i in np.arange(t0+h,2+h,h):
    k1 = f(i,y)
    k2 = f(i+h/2,y+h/2*k1)
    k3 = f(i+h/2,y+h/2*k2)
    k4 = f(i+h,y+h*k3)
    ynew = y+h*(1/6*k1+1/3*k2+1/3*k3+1/6*k4)
    y = ynew
print(y)
```