3η Άσκηση

Παρατηρείται ότι το πολυώνυμο που δίνεται είναι πολυώνυμο Newton. Επομένως, μπορεί πολύ εύκολα να υπολογιστεί ένα νέο πολυώνυμο που να παρεμβάλει και το πέμπτο σημείο.

Συγκεκριμένα, αν $p_3 = 2 - (x+1) + x(x+1) - 2x(x+1)(x-1)$, που είναι τρίτου βαθμού, τότε το νέο πολυώνυμο $p_4(x)$ θα είναι της μορφής (1) και θα είναι τέταρτου βαθμού.

$$p_4(x) = p_3(x) + cx(x+1)(x-1)(x-2) \tag{1}$$

Ο προσδιορισμός της σταθεράς c γίνεται μέσω της τιμής του νέου πολυωνύμου στο επιπλέον σημείο (2). Το υπολογιστικό φορτίο που απαιτείται για να προστεθεί ένας επιπλέον όρος είναι μικρό: 8 πράξεις κινητής υποδιαστολής (4 προσθαφαιρέσεις, 3 πολλαπλασιασμοί και 1 διαίρεση). Εδώ φαίνεται η υπεροχή της χρήσης των πολυωνύμων Newton σε σχέση με τα πολυώνυμα Lagrange, για τα οποία θα χρειαζόταν ο υπολογισμός όλου του πολυωνύμου εξαρχής.

$$\begin{aligned} &p_4(x_4) = 10 \rightarrow \\ &\quad y_4 = p_3(x_4) + cx_4(x_4+1)(x_4-1)(x_4-2) \rightarrow \\ &\quad c = \frac{y_4 - p_3(x_4)}{x_4(x_4+1)(x_4-1)(x_4-2)} \end{aligned} \tag{2}$$

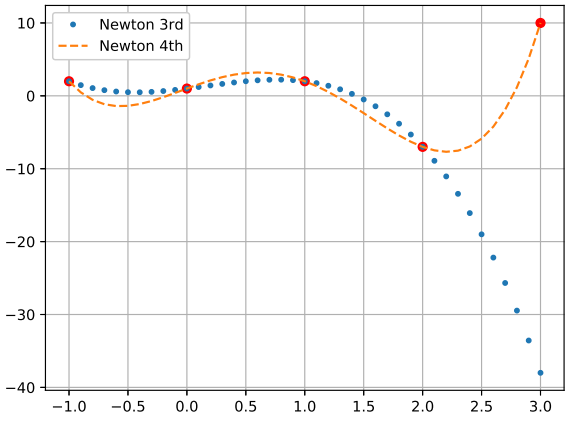
Στο Σχήμα 1 δίνονται τα δύο πολυώνυμα. Όπως είναι αναμενόμενο, τα δύο πολυώνυμα είναι σχετικά κοντά μέχρι το $(2, -7)$ και αποκλίνουν σημαντικά μετά από αυτό.

Παρακάτω ακολουθεί ο κώδικας που γράφτηκε σε Python και έγινε χρήση της βιβλιοθήκης Numpy. Η υλοποίηση των αλγορίθμων προσθήκης νέου όρου στο πολυώνυμο Newton και φωλιασμένου υπολογισμού της τιμής του πολυωνύμου δίνονται στο Παράρτημα.

ex3.py

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

```
import interpolation

xValues = np.array([-1, 0., 1, 2, 3])
yValues = np.array([2., 1, 2, -7, 10])
coefficients = np.array([2., -1, 1, -2])

plt.close('all')
x = np.arange(-1, 3.1, 0.1)
y = interpolation.NestedMultiplication(x,
                                       xValues,
                                       coefficients)

plt.plot(xValues, yValues, 'ro')  # Plot points
plt.plot(x, y, '.', label='Newton 3rd')  # Plot 3rd degree
    Newton

newCoefficients = interpolation.addCoefficient(xValues,
                                               yValues,
                                               coefficients)
print('Order', newCoefficients.size - 1,
      'coefficent is', newCoefficients[-1])

y = interpolation.NestedMultiplication(x,
                                       xValues,
                                       newCoefficients)
plt.plot(x, y, '--', label='Newton 4th')  # Plot 4th degree
    Newton
plt.legend()
plt.grid()
```



Σχήμα 1: Παρεμβολή με πολυώνυμα Newton 3ου και 4ου βαθμού