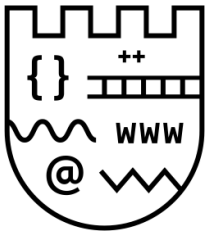


Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Πληροφορικής

Αριθμητική ανάλυσή

2η υποχρεωτική εργασία

Συντάκτης:
Αλέξανδρος Κόρκος

Καθηγητής:
Τέφας Αναστάσιος

ΑΚΑΔΗΜΑΪΚΌ ΈΤΟΣ 2021/22

Περιεχόμενα

Άσκηση 5

Για την υλοποίηση των παρακάτω μεθόδων χρησιμοποιήθηκαν οι παρακάτω τιμές των x στο ανομοιόμορφο κατανεμημένο διάστημα $[-\pi, \pi]$.

i	0	1	2	3	4	5	6	7	8	9
x_i	π	-2.80	-2.22	-1.31	-0.19	0.07	0.8	1.94	3.07	π
$sin(x_i)$	0	-0.33499	-0.79567	-0.96618	-0.18886	0.06994	0.71736	0.93262	0.07153	0

Πίνακας 1: Τιμές της $sin(x)$ για x

Μέθοδος Lagrange

Για την υλοποίησή της μεθόδου Lagrange χρησιμοποιήθηκαν οι παρακάτω τύποι:

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

όπου

$$L_i(x) = \frac{(x - x_0)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

με $i = 0, 1, ..., n$. Η υλοποίηση φαίνεται παρακάτω:

```python
def lagrangeCoefficients(xi, x, n):
    L = np.ones(n)

    for i in range(n):
        for j in range(n):
            if i != j:
                L[i] *= (x - xi[j]) / (xi[i] - xi[j])
    return L

def lagrange(xi, yi, x):
    n = xi.shape[0]
    pn = 0
    for i in range(n):
        pn += yi[i] * lagrangeCoefficients(xi, x, n)[i]

    return pn
```

Κώδικας 1: Μέθοδος Lagrange

Εκτελώντας των κώδικα 1 προκύπτει (όπως φαίνεται και από τον πίνακα 2) πως ταυτίζεται με την πραγματική τιμή του $sin(x_i)$ για $i = 0, 1, ..., 9$ που χρησιμοποιήθηκε.

i	0	1	2	3	4	5	6	7	8	9
x_i	π	-2.80	-2.22	-1.31	-0.19	0.07	0.8	1.94	3.07	π
$sin(x_i)$	0	-0.33499	-0.79567	-0.96618	-0.18886	0.06994	0.71736	0.93262	0.07153	0
$lagrange(x_i)$	-0	-0.33499	-0.79567	-0.96618	-0.18886	0.06994	0.71736	0.93262	0.07153	0

Πίνακας 2: Τιμές της $sin(x)$ και $lagrange(x)$.

Μέθοδος Splines

Το πολυώνυμο της φυσικής κυβικής Spline ορίζεται από τον εξής τύπο:

$$S_{n-1}(x) = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3$$

για $[x_{n-1}, x_n]$.
Για να υπολογιστεί λοιπόν το πολυώνυμο, θα πρέπει να υπολογιστούν οι συντελεστές b_i, c_i, d_i.
Για τον υπολογισμό λοιπόν, θα χρειαστεί να χρησιμοποιηθούν πίνακες, ως εξης:

- $\mathbf{d}$, που αποθηκεύει τις τιμές d_i σε κάθε σειρά του πίνακα με τον τύπο:

  $$d_i = \frac{c_{i+1} - c_i}{3\delta_i}$$

  για $i = 1, ..., n - 1$

- $\mathbf{b}$, που αποθηκεύει τις τιμές b_i σε κάθε σειρά του πίνακα με τον τύπο:

  $$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1})$$

  για $i = 1, ..., n - 1$

- $\mathbf{c}$, που προκύπτει από την λύση του παρακάτω συστήματος:

$$\begin{bmatrix} 1 & 0 & 0 & & & \\ \delta_i & 2\delta_1 + 2\delta_2 & \delta_2 & \ddots & & \\ 0 & \delta_2 & 2\delta_2 + 2\delta_3 & \delta_3 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & & \delta_{n-2} & 2\delta_{n-2} + 2\delta_{n-1} & \delta_{n-1} \\ & & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 3(\frac{\Delta_2}{\delta_2} - \frac{\Delta_1}{\delta_1}) \\ \vdots \\ \\ 3(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}) \\ 0 \end{bmatrix}$$

Επιπλέον θα χρειαστούν και οι βοηθητικοί πίνακές:

- δ, που προκύπτει από τον τύπο:

  $$\delta_i = x_{i+1} - x_i$$

  για $i = 1, ..., n - 1$

- $\mathbf{\Delta}$, που προκύπτει από τον τύπο:

  $$\Delta_i = y_{i+1} - y_i$$

  για $i = 1, ..., n - 1$

Αφού εκτελεσθεί η διαδικασία της φυσικής κυβικής Spline,υπολογίζεται η τιμή του S(x) (όπου για x είναι η γωνία του sin(x) που θέλουμε να υπολογίσουμε).

Η υλοποίηση των προηγουμένων, φαίνεται παρακάτω:

```python
def splines(xi, yi, x):
    n = xi.shape[0]
    A = np.zeros((n, n))
    r = np.zeros(n)
    dx = np.zeros(n - 1)
    dy = np.zeros(n - 1)
    a = yi.copy()
    b = np.zeros(n)
    d = np.zeros(n)

    for i in range(n - 1):
        dx[i] = xi[i + 1] - xi[i]
        dy[i] = yi[i + 1] - yi[i]

    A[0, 0] = 1
    A[-1, -1] = 1

    for i in range(1, n - 1):
        A[i, i] = 2 * (dx[i-1] + dx[i])
        A[i, i + 1] = dx[i]
        A[i, i - 1] = dx[i - 1]

        r[i] = 3 * (dy[i] / dx[i] - dy[i - 1] / dx[i - 1])

    c = np.linalg.inv(A).dot(r)

    for i in range(n - 1):
        b[i] = dy[i] / dx[i] - dx[i] * (2 * c[i] + c[i + 1]) / 3
        d[i] = (c[i + 1] - c[i]) / (3 * dx[i])

    return a, b, c, d
```

Κώδικας 2: Μέθοδος Splines

Εκτελώντας των κώδικα 2 προκύπτει πως ταυτίζεται με την πραγματική τιμή του $sin(x)$ για κάθε x που χρησιμοποιήθηκε.

Μέθοδος ελαχίστων τετραγώνων

Για να λειτουργήσει η μέθοδος των ελάχιστων τετραγώνων, πρέπει να προσδιορίσουμε το αρχικό πολυώνυμο με έναν συγκεκριμένο βαθμό, δηλαδή να έχει την μορφή:

$$P(x) = c_1 + c_2x + ... + c_nx^n$$

όπου n ο βαθμός του πολυωνύμου. Μέσω λοιπόν της μεθόδου θέλουμε η γραφική παράσταση του $P(x)$ να πλησιάσει όσο τον δυνατό πιο κοντά στην γραφική συνάρτηση του $sin(x)$. Συνεπώς, πρέπει να υπολογιστούν οι τιμές των $c_i, i = 1, 2, ..., n$. Για να γίνει αυτό θα πρέπει να χρησιμοποιηθούν οι εξής πίνακες:

- $\mathbf{A}$, που αποθηκεύει της τιμές των x υψωμένες στον βαθμό του πολυωνύμου (για κάθε

σειρά διαφορετικό x), δηλαδή:

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^n \\ 1 & x_2 & x_2^2 & ... & x_2^n \\ 1 & x_3 & x_3^2 & ... & x_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & ... & x_m^n \end{bmatrix}$$

όπου m το πλήθος των τιμών x που δίνονται την μέθοδο.

- $\mathbf{c}$, που αποθηκεύει της τιμές c_i για $i = 1, 2, ..., n$, δηλαδή:

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

- $\mathbf{b}$, που αποθηκεύει σε κάθε σειρά την τιμή $sin(x_i)$ για $i = 1, 2, ..., m$, δηλαδή:

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Τέλος, για να βρούμε τις τιμές των c, θα πρέπει να λυθεί το σύστημα:

$$\mathbf{A^T A c} = \mathbf{A^T b}$$

Αφού εκτελεσθεί η διαδικασία των ελάχιστων τετράγωνων, υπολογίζεται η τιμή του $P(x)$ (όπου για x είναι η γωνία του $sin(x)$ που θέλουμε να υπολογίσουμε). Η υλοποίηση των παραπάνω, φαίνεται παρακάτω:

```
def leastSquares(xi, yi, x, degree = 9):
    n = xi.shape[0]
    A = np.zeros((n, degree))
    b = yi

    for i in range(degree):
        A[:, i] = xi[:] ** i
    c = np.linalg.inv(np.dot(A.T, A)).dot(A.T).dot(b)

    return c
```

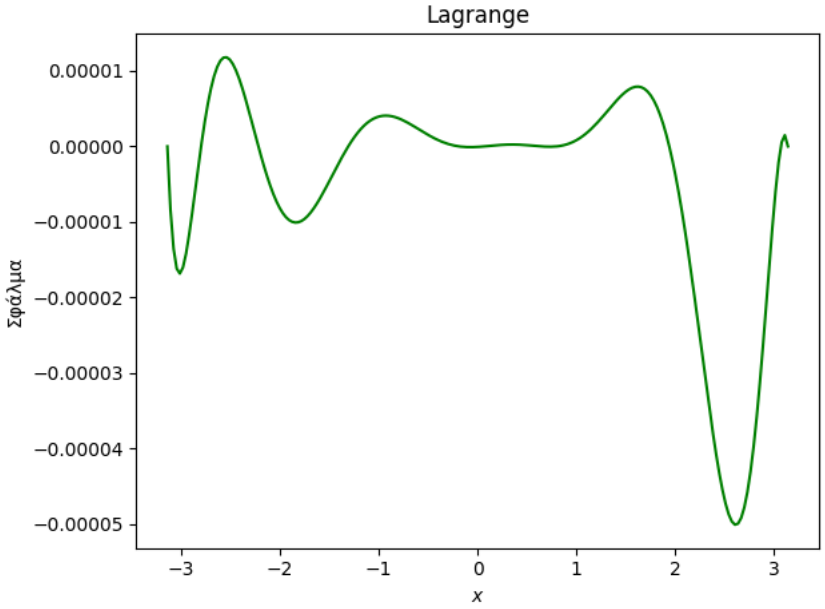
Κώδικας 3: Μέθοδος ελαχίστων τετραγώνων

Εκτελώντας των κώδικα 3 προκύπτει πως ταυτίζεται με την πραγματική τιμή του $sin(x)$ για κάθε x που χρησιμοποιήθηκε.

Υπολογισμός σφαλμάτων

Τα διαγράμματά σφαλμάτων που ακολουθούν, διαμορφώνονται εκτελώντας τα προγράμματα 1, 2, 3. Επιπλέον, εξετάζονται κάθε φορά 200 σημεία στο διάστημα $[-\pi, \pi]$ και χρησιμοποιούνται οι κατάλληλες μετρικές (RMSE) για την αξιολόγηση.

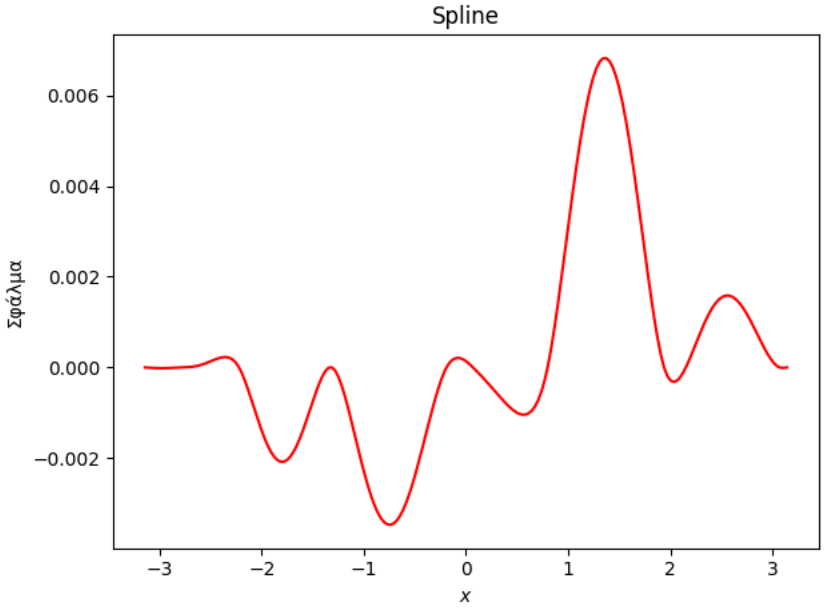
Σφάλμα Lagrange



Σχήμα 1: Σφάλμα 200 σημείων με Lagrange

Εξετάζοντάς την ελάχιστη και την μέγιστη τιμή σφάλματος, βρίσκουμε: $|\min| = |-0.0005| = 0.0005, \max = 0.00055$. Έτσι βλέπουμε ότι η ακρίβεια που επιτυγχάνετε είναι 3 δεκαδικά ψηφία. Επιπλέον το RMSE ισούται με 0.000768. Έχοντας το μικρότερο RMSE από όλες τις μεθόδους καθίσταται και η καλύτερή μέθοδος προσέγγισής από τις τρεις.

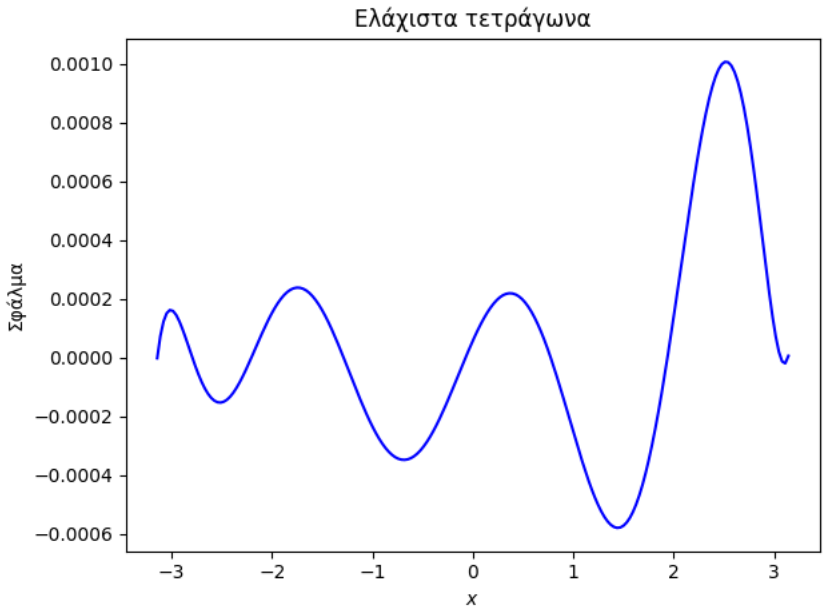
Σφάλμα Splines



Σχήμα 2: Σφάλμα 200 σημείων με Splines

Εξετάζοντάς την ελάχιστη και την μέγιστη τιμή σφάλματος, βρίσκουμε: $|\min| = |-0.003470| = 0.00347, \max = 0.006819$. Έτσι βλέπουμε ότι η ακρίβεια που επιτυγχάνετε είναι 2 δεκαδικά ψηφία. Το RMSE για την παραπάνω μέθοδο ανέρχεται σε 0.002274. Το RMSE της είναι το μεγαλύτερο από τις τρεις, κάνοντας την την χειρότερη μέθοδο προσέγγισης από τις τρεις.

Σφάλμα ελάχιστων τετράγωνων



Σχήμα 3: Σφάλμα 200 σημείων με ελάχιστα τετράγωνα

Εξετάζοντάς την ελάχιστη και την μέγιστη τιμή σφάλματος, βρίσκουμε: $|\min| = |-0.00058| = 0.00058, \max = 0.001007$. Έτσι βλέπουμε ότι η ακρίβεια που επιτυγχάνετε είναι 2 με 3 δεκαδικά ψηφία. Το RMSE για τη παραπάνω μέθοδο είναι 0.00076, καθιστώντας την μέθοδο των ελαχίστων τετραγώνων την δεύτερη καλύτερη από τις τρεις.

Άσκηση 6

Όπως γνωρίζουμε, για έναν αριθμό από n σημεία (εδώ n = 11) χρειάζονται N = n - 1 (εδώ N = 10) διαμερίσεις.

Μέθοδος Simpson

Για τον υπολογισμό του ολοκληρώματος με τον κανόνα του τραπεζίου χρησιμοποιήθηκε ο τύπος:

$$\int_a^b f(x)dx = \frac{b-a}{3n}(f(x_0) + f(x_n) + 2\sum_{i=1}^{n/2+1} f(x_{2i}) + 4\sum_{i=1}^{n/2} f(x_{2i-1}))$$

όπου n ισομήκη υποδιαστήματα του $[a, b]$, $x_0 = a, x_n = b$ και $x_i = x_0 + i\frac{b-a}{n}$ με $i = 0, ..., n$. Η υλοποίηση φαίνεται παρακάτω:

```
def simpson(f, a, b, n):
    I = f(a) + f(b)
    h = (b - a) / n

    firstSum = 0
    for i in range(1, n // 2 + 1):
        xi = a + (2 * i - 1) * h
        firstSum += f(xi)

    secondSum = 0
    for i in range(1, n // 2):
        xi = a + 2 * i * h
        secondSum += f(xi)

    I += 4 * firstSum + 2 * secondSum
    I *= h / 3

    return I
```

Κώδικας 4: Μέθοδος Simpson

Εκτελώντας τον κώδικα 4 για $f(x) = sin(x)$ για $[0, \pi/2]$, επιστρέφεται πως:

$$\int_0^{\pi/2} sin(x)dx = \mathbf{1.000003}$$

.

Για τον υπολογισμό του θεωρητικού σφάλματος χρησιμοποιήθηκε ο τύπος:

$$|e| \leq \frac{(b-a)^5}{180n^4} M$$

με $M = \max_{x \in [a,b]}\{|f^{(4)}(x)| : x \in [\alpha, b]\}$. Η υλοποίηση φαίνεται παρακάτω:

```
def simpsonError(d4f, a, b, n):
    M = abs(d4f(b))
    if abs(d4f(a)) > abs(d4f(b)):
        M = abs(d4f(a))
    return (b - a) ** 5 * M / (180 * n ** 4)
```

Κώδικας 5: Μέθοδος Simpson

Εκτελώντας τον κώδικα 5 το θεωρητικό σφάλμα $|e| \leq 0.000005$ και με αριθμητικό σφάλμα $|e| \approx 0.000003$.

Μέθοδος τραπεζίου

Για τον υπολογισμό του ολοκληρώματος με τον κανόνα του τραπεζίου χρησιμοποιήθηκε ο τύπος:

$$\int_a^b f(x)dx = \frac{b-a}{2n}(f(x_0) + f(x_n) + 2\sum_{i=1}^{n-1} f(x_i))$$

όπου n ισομήκη υποδιαστήματα του $[a, b]$, $x_0 = a, x_n = b$ και $x_i = x_0 + i\frac{b-a}{n}$ με $i = 0, ..., n$. Η υλοποίηση φαίνεται παρακάτω:

```
def trapezoid(f, a, b, n):
    I = f(a) + f(b)
    h = (b - a) / n
    for i in range(1, n):
        xi = a + i * h
        I += 2 * f(xi)
    I *= h / 2
    return I
```

Κώδικας 6: Μέθοδος τραπεζίου

Εκτελώντας τον κώδικα 6 για $f(x) = sin(x)$ για $[0, \pi/2]$, επιστρέφεται πως:

$$\int_0^{\pi/2} sin(x)dx = \mathbf{0.997943}$$

.

Για τον υπολογισμό του θεωρητικού σφάλματος χρησιμοποιήθηκε ο τύπος:

$$|e| \leq \frac{(b-a)^3}{12n^2} M$$

με $M = \max_{x \in [a,b]}\{|f''(x)| : x \in [\alpha, b]\}$. Η υλοποίηση φαίνεται παρακάτω:

```python
def trapezoidError(d2f, a, b, n):
    M = abs(d2f(b))
    if abs(d2f(a)) > abs(d2f(b)):
        M = abs(d2f(a))
    return (b - a) ** 3 * M / (12 * n ** 2)
```

Κώδικας 7: Μέθοδος τραπεζίου

Εκτελώντας τον κώδικα 7 το θεωρητικό σφάλμα $|e| \leq 0.003230$ και με αριθμητικό σφάλμα $|e| \approx 0.002057$.

Άσκηση 7

Η ημερομηνία γέννησης μου είναι 18/4, για αυτό το λόγο θα χρησιμοποιηθούν οι εξής συνεδρίες από το έτος 2021: 1/4, 6/4, 7/4, 8/4, 9/4, 12/4, 13/4, 14/4, 15/4 και 16/4. Η μετοχές που εξετάστηκαν είναι η **ΟΛΘ** και η **ΣΠΕΙΣ**. Παρακάτω θα ακολουθήσουν οι προβλέψεις για το διάστημα 19/4 έως 23/4 και με Α/Α συνεδρίασης στο διάστημα [11, 15].

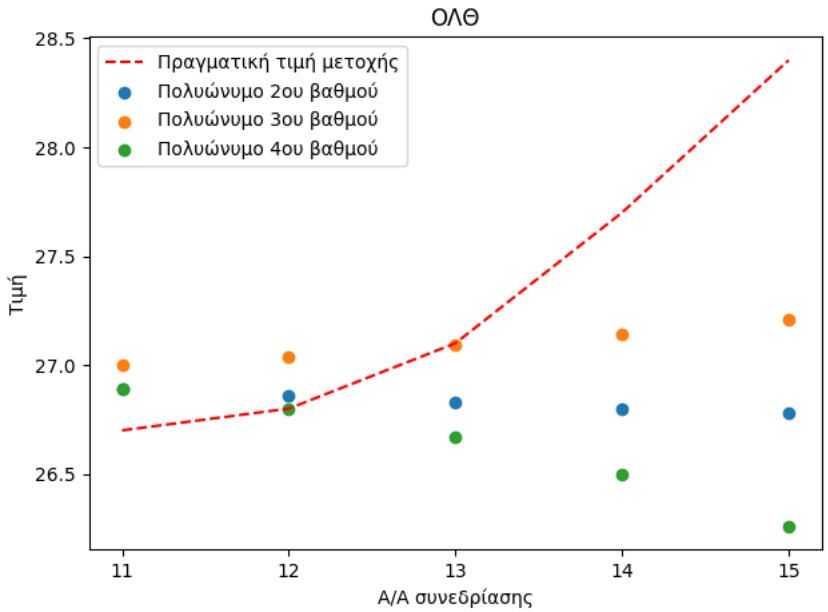
Προβλέψεις ΟΛΘ

Το ιστορικό της μετοχής για τις δέκα παραπάνω ημερομηνίες, διαμορφώνεται όπως δείχνει ο πίνακας 3.

Α/Α συνεδρίασης	Ημερ/νία	Τιμή κλεισίματος
1	1/4	27,20
2	6/4	27,30
3	7/4	27,00
4	8/4	27,00
5	9/4	27,00
6	12/4	26,90
7	13/4	27,10
8	14/4	27,10
9	15/4	26,80
10	16/4	27,00

Πίνακας 3: Ιστορικό ΟΛΘ

Συνεπώς, εκτελώντας τον κώδικα 3 δίνοντας του τα περιεχόμενα του πίνακα 3, παρατηρείται:



Σχήμα 4: Προσεγγίσεις ΟΛΘ

Για να αποτυπωθεί μια σαφέστερη εικόνα για κάθε προσέγγιση θα πρέπει να εξεταστούν και οι τιμές των προσεγγίσεων με την πραγματική τιμή της μετοχής από τον πίνακα 4:

Α/Α	Προσεγγίσεις			Πραγματική τιμή
	2ου βαθμού	3ου βαθμού	4ου βαθμού	
11	26.88	27.00	26.89	26.70
12	26.85	27.03	26.80	26.80
13	26.83	27.08	26.67	27.10
14	26.80	27.14	26.49	27.70
15	26.77	27.21	26.26	28.40

Πίνακας 4: Αναλυτικές προσεγγίσεις για ΟΛΘ

Αξιολογώντας τα δεδομένα του πίνακα 4 καταλήγουμε στο ότι η καλύτερη προσέγγιση προήλθε από το πολυώνυμο 3ου βαθμού μιας και οι προβλέψεις είναι σχετικά κοντά με την πραγματική τιμή. Καλές προβλέψεις φαίνεται να έδωσε και το πολυώνυμο 2ου βαθμού σε αντίθεση με αυτό του 4ου βαθμού που πέρα από την πρόβλεψη για την 12 συνεδρίαση, οι υπόλοιπες απείχαν από την πραγματική τιμή της μετοχής.

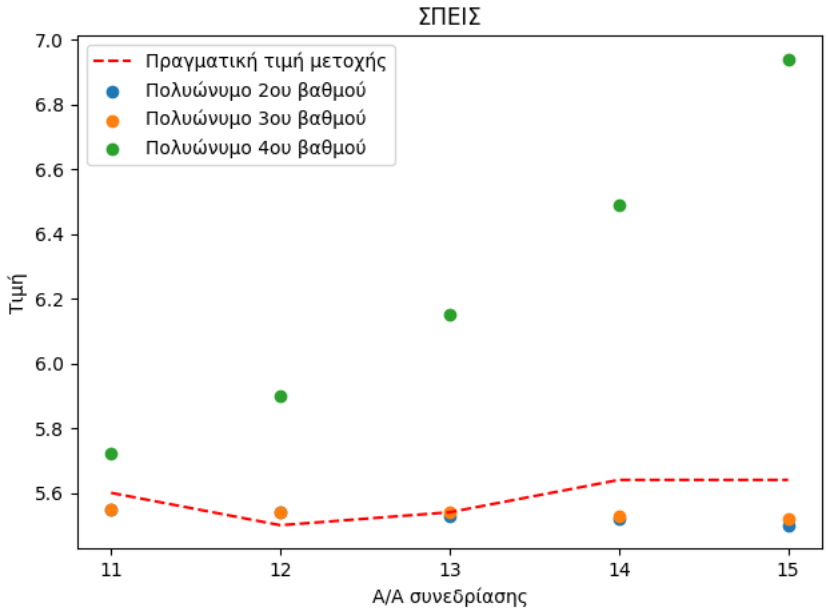
Προβλέψεις ΣΠΕΙΣ

Το ιστορικό της μετοχής για τις δέκα παραπάνω ημερομηνίες, διαμορφώνεται όπως δείχνει ο πίνακας 5.

Α/Α συνεδρίασης	Ημερ/νία	Τιμή κλεισίματος
1	1/4	5,64
2	6/4	5,60
3	7/4	5,64
4	8/4	5,70
5	9/4	5,66
6	12/4	5,60
7	13/4	5,50
8	14/4	5,50
9	15/4	5,60
10	16/4	5,60

Πίνακας 5: Ιστορικό ΣΠΕΙΣ

Συνεπώς, εκτελώντας τον κώδικα 3 δίνοντας του τα περιεχόμενα του πίνακα 5, παρατηρείται:



Σχήμα 5: Προσεγγίσεις ΣΠΕΙΣ

Για να αποτυπωθεί μια σαφέστερη εικόνα για κάθε προσέγγιση θα πρέπει να εξεταστούν και οι τιμές των προσεγγίσεων με την πραγματική τιμή της μετοχής από τον πίνακα 6:

Α/Α	Προσεγγίσεις			Πραγματική τιμή
	2ου βαθμού	3ου βαθμού	4ου βαθμού	
11	5.54	5.55	5.72	5.64
12	5.53	5.54	5.89	5.50
13	5.52	5.53	6.15	5.54
14	5.51	5.52	6.49	5.64
15	5.50	5.52	6.93	5.64

Πίνακας 6: Αναλυτικές προσεγγίσεις για ΣΠΕΙΣ

Κρίνοντας τα αποτελέσματα από τον πίνακα 6 μπορούμε να αποφανθούμε στο ότι η καλύτερη

πρόβλεψη επιτεύχθηκε με το πολυώνυμο 3ου βαθμού. Επιπλέον και η πρόβλεψη του πολυωνύμου 2ου βαθμού είναι και αυτή αρκετά καλή. Τέλος η προσέγγιση του πολυώνυμο 4ου βαθμού είναι εντελώς ανακριβής μιας και δεν πετυχαίνει να βρεθεί κοντά σε καμία τιμή της μετοχής.

Πηγές

[1] Timothy Sauer, *Numerical Analysis 2nd ed.*, Pearson Education, 2012.

[2] Γ. Σ. Παπαγεωργίου, Χ. Γρ. Τσίτσουρας, *Αριθμητική Ανάλυση με εφαρμογές σε Mathematica και Matlab*, Εκδόσεις Τσότρας, 2015.

[3] Jaan Kiusalaas, *NUMERICAL METHODS IN ENGINEERING WITH Python*, Cambridge University Press, 2005.