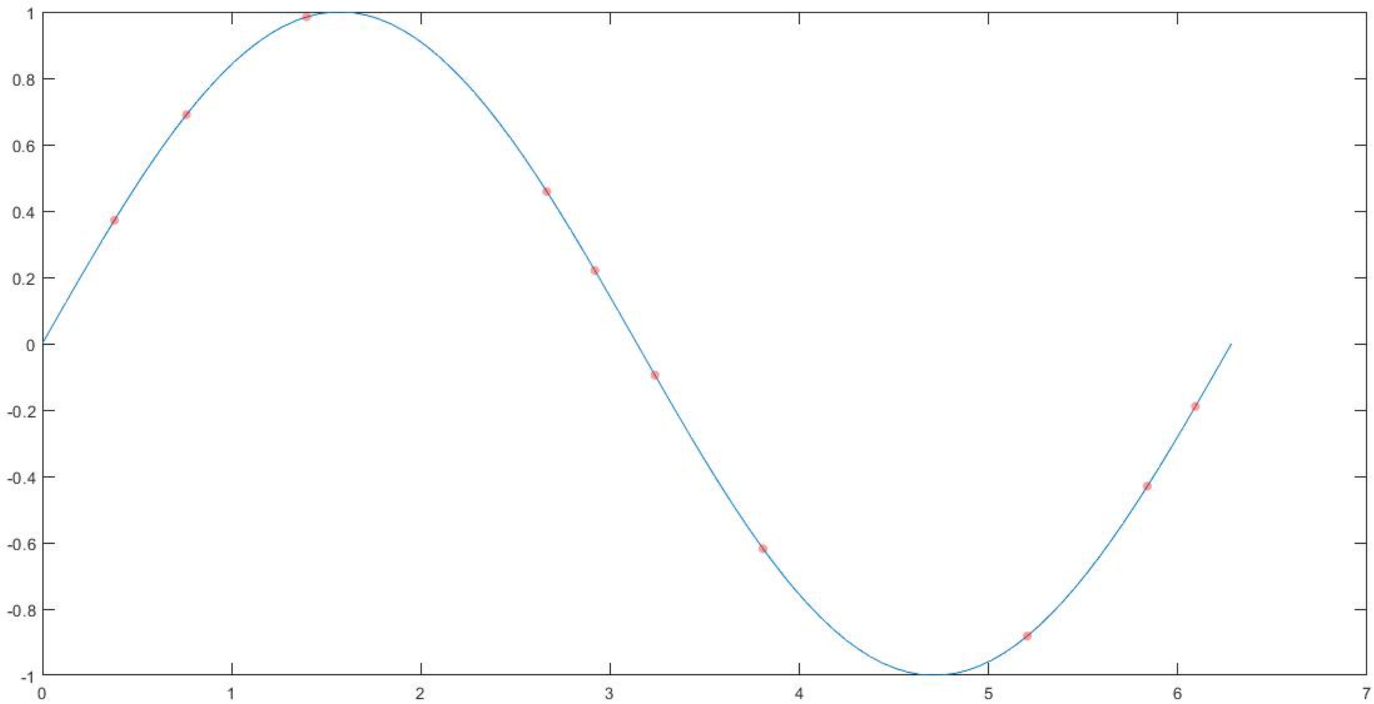
2Η ΕΡΓΑΣΙΑ ΑΡΙΘΜΗΤΙΚΗ ΑΝΑΛΥΣΗ

ΑΘΑΝΑΣΙΟΣ ΞΑΝΘΟΠΟΥΛΟΣ
ΑΕΜ: 2392

- **Άσκηση 5** (κώδικας στο αρχείο **askisi5_2392.py**)

Τα σημεία που επέλεξα(στον οριζόντιο άξονα χ) για να δημιουργήσω την προσέγγιση της συνάρτησης του ημιτόνου είναι:

```
x_my_sin_points = [0.380799109526036, 0.761598219052071, 1.39626340159546,
                   2.66559376668225,
                   2.91945983969961, 3.2367924309713, 3.80799109526036,
                   5.20425449685582, 5.83891967939921, 6.09278575241657]
```



i) **πολυωνυμική προσέγγιση:** Υλοποιείται με την μέθοδο Newton στην συνάρτηση

```
def polyonimiki_prosegisi_newton(x_points, y_points):

    n = len(x_points)

    # Το πολυώνυμο 1x+0 (μεταβλητή x σε συμβατή μορφη)
    x = numpy.poly1d([1, 0])

    # Διαιρεμένες διαφορές υπολογισμένες στον πίνακα Dij
     Dij = coef(x_points, y_points)

     N = 0
```

```
13)     for j in range(0, n):
14)         nj = 1
15)         for i in range(0, j):
16)             nj = numpy.polymul(numpy.polyadd(x, -x_points[i]), nj)
17)         N = numpy.polyadd(numpy.polymul(Dij[j], nj), N)
18)     return N
```

Επιστρέφει το πολυώνυμο(προσέγγιση της συνάρτησης) που περνάει από τα σημεία που παίρνει σαν όρισμα, με την μέθοδο του Newton.
Η μορφή της επιστρεφόμενης μεταβλητής είναι: numpy.poly1d
όπου αποτελεί δομή δεδομένων που προσομοιάζει τα πολυώνυμα
Ο υπολογισμός των διαιρεμένων διαφορών γίνεται στην συνάρτηση **coef.**
Έπειτα από την γραμμή 13 και μετά υπολογίζω του όρους σύμφωνα με τον τύπο:

$$N(x) := \sum_{j=0}^{k} a_j n_j(x)$$

όπου

$$n_j(x) := \prod_{i=0}^{j-1} (x - x_i)$$

ii) μέθοδος ελαχίστων τετραγώνων: Υλοποιείται με την βοήθεια της δομής πολυωνύμων από την βιβλιοθήκη numpy

```
def methodos_elaxistwn_tetragwnwn(x_points, y_points):
    # προσέγγιση των σημειών με πολυώνυμο 1ου βαθμού
    # με την μέθοδο των ελαχίστων τετραγώνων
    x_meso = numpy.mean(x_points)
    y_meso = numpy.mean(y_points)

    sum_arith = 0
    sum_paron = 0
    for i in range(len(x_points)):
        # Δημιουργούμε το άθροισμα του αριθμητή
        sum_arith += (x_points[i] - x_meso) * (y_points[i] - y_meso)
        # Δημιουργούμε το άθροισμα του παρονομαστή
        sum_paron += (x_points[i] - x_meso) ** 2

    b = sum_arith / sum_paron
    a = y_meso - b * x_meso

    eutheia_proseggisis = numpy.poly1d([b, a])

    return eutheia_proseggisis
```

Επιστρέφει το πολυώνυμο(προσέγγιση της συνάρτησης) που περνάει από τα σημεία που παίρνει σαν όρισμα, με την μέθοδο των ελάχιστων τετραγώνων.
Η μορφή της επιστρεφόμενης μεταβλητής είναι: numpy.poly1d

Υπολογίζω του όρους σύμφωνα με τον τύπο:

$$b = \frac{\sum(x-\bar{x}) * (y-\bar{y})}{\sum(x-\bar{x})^2}$$

Όπου **Y = a + bX**

Μέσο σφάλμα πολυωνυμικής προσέγγισης Newton: 0.5047188139829841
Σφάλμα προσέγγισης με την μέθοδο των ελάχιστων τετραγώνων: 18.88863678035914

- **Άσκηση 6** (κώδικας στο αρχείο **askisi6_2392.py**)