Αριθμητική Ανάλυση - Δεύτερη υποχρεωτική εργασία

Ονοματεπώνυμο: Δεληγιαννάκης Χαράλαμπος
ΑΕΜ: 4383

Ιανουάριος 2024

Εισαγωγή

Γενικά σχόλια για τους κώδικες

Οι κώδικες γράφτηκαν σε γλώσσα C, χρησιμοποιώντας το Visual Studio Code ως προγραμματιστικό περιβάλλον και το MinGW ως compiler. Σε κάθε κώδικα οποιασδήποτε άσκησης, υπάρχουν βοηθητικά σχόλια για την καλύτερη αποσαφήνισή του, αλλά και έξοδοι στην κονσόλα, όπως έξοδοι που σχετίζονται με την άσκηση αλλά και μια επιπλέον έξοδο σχετικά με το πόσα δευτερόλεπτα πήρε ο κάθε αλγόριθμος για να τρέξει. Το pre-compile παίρνει σχετικά μεγάλο χρονικό διάστημα, οπότε η συγκεκριμένη έξοδος προστέθηκε, επίσης, για να διευκρινήσει το πόσος χρόνος πραγματικά αφιερώθηκε για την εκτέλεση του αλγορίθμου. Στην εισαγωγή της κάθε άσκησης της εργασίας, αναφέρονται λεπτομέρειες για τον κώδικα, αλλά και άλλες πληροφορίες, όπως για παράδειγμα αν, και κατά πόσο, χρησιμοποιήθηκε κάποιο γλωσσικό μοντέλο για τη διευκόλυνση της επίλυσης.

Πρώτη Άσκηση

Γενικά σχόλια για τους κώδικες της άσκησης

Για την συγκεκριμένη άσκηση, **δεν χρησιμοποιήθηκε καθόλου κάποιο γλωσσικό μοντέλο για την διευκόλυνση της γραφής του δικού μου κώδικα.** Οι κώδικες υπάρχουν στον φάκελο `5`.

10 γνωστά σημεία

Τα 10 σημεία για τα οποία θέτουμε γνωστές τις τιμές του ημιτόνου, είναι εξής (κατενεμήθηκαν ανομοιόμορφα):

```
1 long double x_values[N] = {-PI/2, -PI/4, -PI/6, -PI/8,
      0, PI/10, PI/7, PI/5, PI/3, PI}; // Initialize
     known x values disparately
```

Ισχυεί, προφανώς, ότι `N=10`.

Πολυωνυμική προσέγγιση

Για την πολυωνυμική προσέγγιση της συνάρτησης του ημιτόνου, χρησιμοποιήθηκε το **πολυώνυμο Lagrange**.

Σχετικά με τον κώδικα, αρχικά στην `main()`, αρχικοποιούνται 10 ανομοιόμορφες τιμές και υπολογίζονται οι 10 τιμές της συνάρτησης του ημιτόνου για κάθε μία από αυτές. Επομένως, έχουν παραχθεί 10 γνωστά σημεία (x_i, y_i) της $f(x) = sin(x)$. Στη συνέχεια, αρχικοποιούνται 200 σημεία στον άξονα x εντός του διαστήματος $[-\pi, \pi]$ και για καθένα από αυτά, υπολογίζεται η προσσέγιση Lagrange καθώς και το αντίστοιχο σφάλμα. Τέλος, για κάθε ένα από τα 200 σημεία, εμφανίζεται η πραγματική τιμή της συνάρτησης, η προσσεγιστικη τιμή Lagrange και το σφάλμα. Υπάρχει, επίσης, η δυνατότητα να δώσει ο χρήστης κάποιο σημείο και να υπολογιστεί γι'αυτό μία προσέγγιση.

Σχετικά με τη συνάρτηση `lagrange()`, δέχεται το σημείο `x‘` για το οποίο ζητείται η προσέγγιση, καθώς και τα 10 ζευγάρια (x_i, y_i), δηλαδή τα 10 γνωστά σημεία της $sin(x)$. Έπειτα, με τη χρήση δύο `for-loop`, υπολογίζεται η τιμή της Lagrange στο σημείο `x‘`, δηλαδή η τιμή:

$$p_n(x) = \sum_{i=0}^{9} y_i L_i(x‘)$$

όπου ισχύει:

$$L_i(x‘) = \frac{(x‘ - x_0)...(x‘ - x_{i-1})(x‘ - x_{i+1})...(x‘ - x_9)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_9)}$$

Για τα πρώτα 5 και τελευταία 5 σημεία από τα 200 για τα οποία παράγεται έξοδος, η έξοδος είναι η εξής:

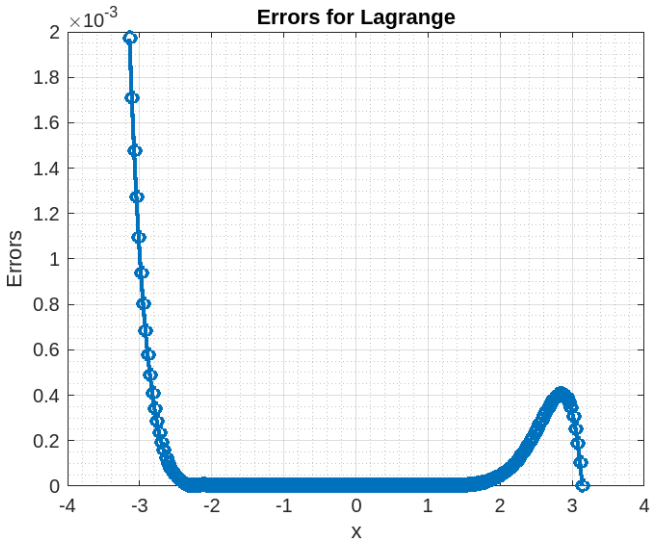
For x=-3.141593: Real=-0.000000 Approx=-0.001971 Error:=0.00197057063893785146
For x=-3.110019: Real=-0.031569 Approx=-0.033278 Error:=0.00170898009161623188
For x=-3.078445: Real=-0.063106 Approx=-0.064584 Error:=0.00147804612512948555
For x=-3.046871: Real=-0.094580 Approx=-0.095854 Error:=0.00127461526852522982
For x=-3.015297: Real=-0.125960 Approx=-0.127056 Error:=0.00109581576406832487
...
For x=3.015297: Real=0.125960 Approx=0.125654 Error:=0.00030537714852207860
For x=3.046871: Real=0.094580 Approx=0.094326 Error:=0.00025344924032056281
For x=3.078445: Real=0.063106 Approx=0.062919 Error:=0.00018676739315301016
For x=3.110019: Real=0.031569 Approx=0.031465 Error:=0.00010310785395559134
For x=3.141593: Real=0.000000 Approx=0.000000 Error:=0.00000000000000000000

Από τα 200 σημεία, το σημείο με το μέγιστο σφάλμα είναι το εξής:
For x=-3.141593: Real=-0.000000 Approx=-0.001971 Error:=0.00197057063893785146

Και αυτό με το μικρότερο σφάλμα είναι το εξής:
For x=3.141593: Real=0.000000 Approx=0.000000 Error:=0.00000000000000000000

Η ευκλείδια νόρμα του πίνακα σφαλμάτων είναι η εξής: 0.00417711955750670830

Το διάγραμμα που δείχνει την πορεία του σφάλματος και για τα 200 σημεία είναι το εξής:



Splines προσέγγιση

Συναρτήσεις helpers

Το αρχείο `helpers.h` περιέχει δήλωση συναρτήσεων, οι οποίες χρησιμοποιήθηκαν στην προηγούμενη εργασία, καθώς και παραλλαγές τους:

```
1 void multiplyMV(long double **P, long double *b, long
    double *result, int rowFirst, int columnFirst, int
    rowSecond); // Function to multiply a matrix with a
     vector
2 void swap(long double *x, long double *y, int n); //
    Function to swap two vectors
3 void initializeIn(long double **X, int n); // Function
     to initialize In matrix
4 void copy(long double **X, long double **Y, int n); //
     Function to copy two matrices
5 void palu(long double **P, long double **A, long
    double **L, long double **U, int n); // Function to
     do the PA = LU factorization
6 void solveY(long double **L, long double *y, long
    double *b, int n); // Function to solve Ly = b with
     current b = Pb
7 void solveX(long double **U, long double *x, long
    double *y, int n); // Function to solve Ux = y
8 long double* solve(long double **P, long double **A,
    long double **L, long double **U, long double *b,
    long double *y, long double *x, int n); // Function
     to solve a linear system using the PA = LU
    factorization method
```

Οι υλοποιήσεις τους βρίσκονται στο αρχείο `helpers.c`

Σχετικά με τον κώδικα, η `main()` λειτουργεί με τον ίδιο τρόπο όπως στην πολυωνυμική προσέγγιση Lagrange, με τη διαφορά ότι αντί για τη συνάρτηση `lagrange()`, καλεί τις συναρτήσεις `splines()` και `calculateWithSplines()`.

Σχετικά με τη συνάρτηση `splines()`, αρχικά αρχικοποιούνται οι κατάλληλοι πίνακες. Στη συνέχεια, υπολογίζονται τα `dx[]` $= \delta_i = x_{i+1} - x_i$ και `dy[]` $= \Delta_i = y_{i+1} - y_i$. Έπειτα, κατασκευάζονται οι εξής πίνακες:

Ο πίνακας **A** για τον οποίο ισχύει:
A[0][0] = A[N-1][N-1] = 1
A[i][i-1] = dx[i-1]
A[i][i] = 2 * (dx[i-1] + dx[i])
A[i][i+1] = dx[i]
Το διάνυσμα-στήλη **b** για το οποίο ισχύει:
b[0] = b[N-1] = 0
b[i] = 3 * (dy[i] / dx[i] - dy[i-1] / dx[i-1])

Οι παραπάνω τύποι προέρχονται από το βιβλίο του μαθήματος. (σελ. 185)

Στη συνέχεια, με τη χρήση helper-συναρτήσεων, λύνεται το γραμμικό σύστημα $\mathbf{Ac} = \mathbf{b}$, με τον πίνακα **c** να περιέχει τους συνελεστές c‘ των 9 spline συναρτήσεων. Το παραπάνω γραμμικό σύστημα λύνεται με τη μέθοδο της παραγοντοποιήσης $\mathbf{PA} = \mathbf{LU}$, που υλοποιήθηκε στο πρώτο υποερώτημα της 3ης άσκησης της προηγούμενης εργασίας.

Έτσι, πλέον έχουμε τους συνελεστές c‘, από τους οποίους μπορούμε να βρούμε τους συντελεστές b‘ και d‘ από τους εξής τύπους:

$$d‘_i = \frac{c‘_{i+1} - c‘_i}{3\delta_i}$$
$$b‘_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c‘_i + c‘_{i+1})$$

Οι παραπάνω τύποι προέρχονται από το βιβλίο του μαθήματος. (σελ. 184)

Τέλος, υπολογίζονται τα οι σταθεροί τελεστές, για τους οποίους ισχύει $a‘_i = y_i$, όπου y_i οι γνωστές τιμές της $f(x) = sin(x)$. Επιστρέφονται οι, πλέον. γνωστοί τέσσερις συντελεστές ($a‘, b‘, c‘, d‘$ - αποθηκεύονται σε `struct`) για κάθε μία από τις 9 splice συναρτήσεις (συνολικά έχουμε 36 συντελεστές).

Σχετικά με τη συνάρτηση `calculateWithSplines()`, αρχικά ανάλογα με το $x‘$ που δέχεται, δηλαδή την τιμή για τη οποία ζητείται η προσέγγιση, αναγνωρίζει το διάστημα στο οποίο ανήκει (από τα 9), και με την αντίστοιχη συνάρτηση splice συνάρτηση $s_n(x)$ του διαστήματος n, υπολογίζει και επιστρέφει την προσέγγιση (χρησιμοποιώντας προφανώς τους αντίστοιχους συντελεστές που υπολογίστηκαν στην συνάρτηση `splines()`).

Για τα πρώτα 5 και τελευταία 5 σημεία από τα 200 για τα οποία παράγεται έξοδος, η έξοδος είναι η εξής:

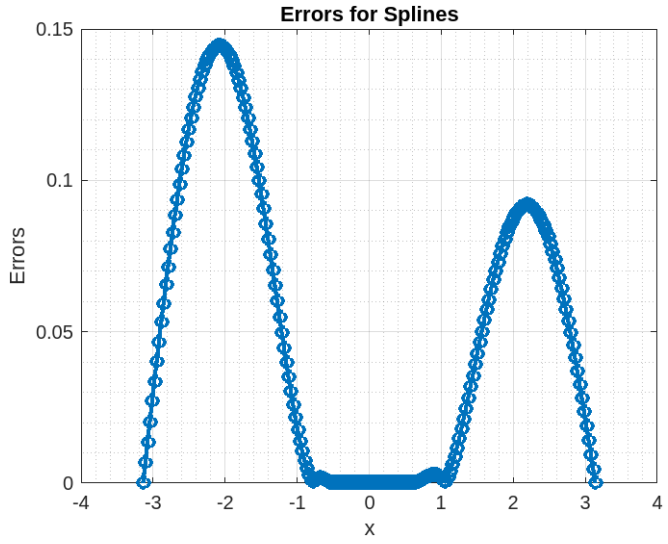
For x=-3.141593: Real=-0.000000 Approx=-0.000000 Error:=0.0000000000000000000
For x=-3.110019: Real=-0.031569 Approx=-0.024776 Error:=0.0067923290371049794 1
For x=-3.078445: Real=-0.063106 Approx=-0.049536 Error:=0.013569678061507519 51
For x=-3.046871: Real=-0.094580 Approx=-0.074263 Error:=0.020317098428708300 21
For x=-3.015297: Real=-0.125960 Approx=-0.098940 Error:=0.027019704199345198 63
...
For x=3.015297: Real=0.125960 Approx=0.107047 Error:=0.018912677891293340 48
For x=3.046871: Real=0.094580 Approx=0.080351 Error:=0.014228884868839495 81
For x=3.078445: Real=0.063106 Approx=0.053599 Error:=0.009507086245961875 53
For x=3.110019: Real=0.031569 Approx=0.026809 Error:=0.004759898296642044 52
For x=3.141593: Real=0.000000 Approx=0.000000 Error:=0.0000000000000000000

Από τα 200 σημεία, το σημείο με το μέγιστο σφάλμα είναι το εξής:
For x=-2.099657: Real=-0.863382 Approx=-0.718950 Error:=0.14443240561886061535

Και αυτό με το μικρότερο σφάλμα είναι το εξής:
For x=3.141593: Real=0.000000 Approx=0.000000 Error:=0.0000000000000000000

Η ευκλείδια νόρμα του πίνακα σφαλμάτων είναι η εξής: 0.98631006765259232605

Το διάγραμμα που δείχνει την πορεία του σφάλματος και για τα 200 σημεία είναι το εξής:



Προσέγγιση ελαχίστων τετραγώνων

Για την προσέγγιση της συνάρτησης με τη μέθοδο ελαχίστων τετραγώνω, ως μοντέλο επιλέχθηκε η γραμμή, δηλαδή πολυώνυμο $1^{ου}$ βαθμού ώστε να χρησιμοποιηθούν πολυώνυμα διαφόρων τάξεων, καθώς χρησιμοποιείται και στην Άσκηση 7, αλλά με πολυώνυμα $2^{ου}$, $3^{ου}$ και $4^{ου}$ βαθμού.

Σχετικά με τον κώδικα, η `main()` λειτουργεί με τον ίδιο τρόπο όπως στην πολυων-

υμική προσέγγιση Lagrange και Splines, με τη διαφορά ότι καλεί τις συναρτήσεις `leastSquares()` και `calculateWithLeastSquares()`.

Σχετικά με τη συνάρτηση `leastSquares()`, υπολογίζονται τα m και b της αναζητούμενης προσέγγισης-ευθείας $y = mx + b$, με τους εξής τύπους:

$$m = \frac{N\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}$$

$$b = \frac{\sum_{i=1}^{N} y_i - m\sum_{i=1}^{N} x_i}{N}$$

όπου προφανώς $N = 10$.

Σχετικά με τη συνάρτηση `calculateWithleastSquares()`, υπολογίζει και επιστρέφει την προσέγγιση της συνάρτησης στο σημείο x' για το οποίο ζητείται προσέγγιση (χρησιμοποιώντας προφανώς τους αντίστοιχους συντελεστές που υπολογίστηκαν στην συνάρτηση `least_squares()`).

Για τα πρώτα 5 και τελευταία 5 σημεία από τα 200 για τα οποία παράγεται έξοδος, η έξοδος είναι η εξής:

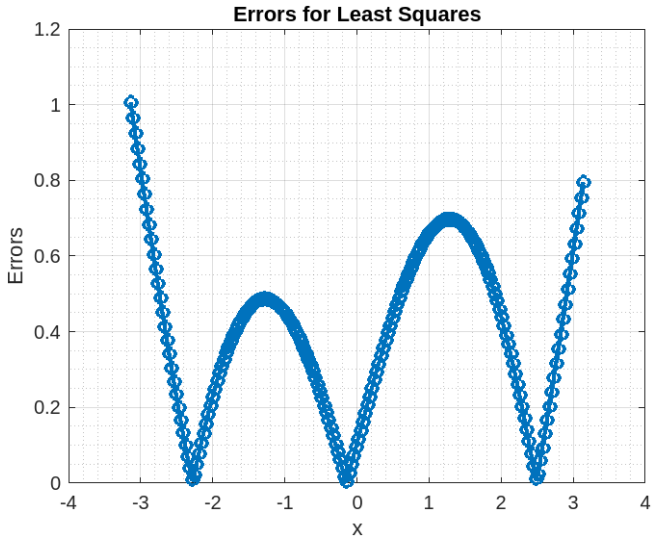
For x=-3.141593: Real=-0.000000 Approx=-1.006003 Error:=1.00600312604127095639
For x=-3.110019: Real=-0.031569 Approx=-0.996952 Error:=0.96538384402297672260
For x=-3.078445: Real=-0.063106 Approx=-0.987902 Error:=0.92479603022162981674
For x=-3.046871: Real=-0.094580 Approx=-0.978851 Error:=0.88427112148597408758
For x=-3.015297: Real=-0.125960 Approx=-0.969800 Error:=0.84384049195961763346
...
For x=3.015297: Real=0.125960 Approx=0.758890 Error:=0.63292995832037368675
For x=3.046871: Real=0.094580 Approx=0.767940 Error:=0.67336058784673069599
For x=3.078445: Real=0.063106 Approx=0.776991 Error:=0.71388549658238520390
For x=3.110019: Real=0.031569 Approx=0.786042 Error:=0.75447331038373388612
For x=3.141593: Real=0.000000 Approx=0.795093 Error:=0.79509259240202756480

Από τα 200 σημεία, το σημείο με το μέγιστο σφάλμα είναι το εξής:
For x=-3.141593: Real=-0.000000 Approx=-1.006003 Error:=1.00600312604127095639

Και αυτό με το μικρότερο σφάλμα είναι το εξής:
For x=2.478543: Real=0.615523 Approx=0.605027 Error:=0.01049602582931335147

Η ευκλείδια νόρμα του πίνακα σφαλμάτων είναι η εξής: 6.50036693477988603007

Το διάγραμμα που δείχνει την πορεία του σφάλματος και για τα 200 σημεία είναι το εξής:



Σύγκριση

Νόρμες:

Lagrange: 0.004177
Splines: 0.98631
Least Squares: 6.500367

Άρα, για τη συγκεκριμένη συνάρτηση, με τα συγκεκριμένα 10 αρχικά σημεία και 200 σημεία για τα οποία έγινε προσέγγιση που διάλεξα και τον συγκεκριμένο τρόπο που υλοποίησα αυτές τις τρεις μεθόδος, η πιο αποδοτική μέθοδος είναι η Lagrange, ενώ η λιγότερη αποδοτική είναι η Least Squares.

Δεύτερη Άσκηση

Γενικά σχόλια για τους κώδικες της άσκησης

Για την συγκεκριμένη άσκηση, **δεν χρησιμοποιήθηκε καθόλου κάποιο γλωσσικό μοντέλο για την διευκόλυνση της γραφής του δικού μου κώδικα.** Οι κώδικες υπάρχουν στον φάκελο 6 με ονόματα `simpson.c` και `trapeziod.c` αντίστοιχα.

Σημεία x

Οι δύο προσεγγίσεις έγιναν με τα εξής σημεία x:
$a = 0, 0.157080, 0.314159, 0.471239, 0.628319, 0.785398,$
$0.942478, 1.099557, 1.256637, 1.413717, 1.570796 = b = \frac{\pi}{2}$
Ο τρόπος που προέκυψαν αναλύεται παρακάτω.

Μέθοδος Simpson

Ο κώδικας καλεί την συνάρτηση `void simpsonMethod(double a, double b, int N)`, που υλοποιεί τη μέθοδο Simpson για τη προσέγγιση του $\int_0^{\frac{\pi}{2}} \sin(x)\,dx$. Τα `a` και `b` αποτελούν τα όρια του ολοκληρώματος, 0 και $\frac{\pi}{2}$ αντίστοιχα, ενώ

η μεταβλητή `N` έχει αρχικοποιηθεί στην `main()` με $N = 10$ και αποτελεί τον αριθμό των υποδιαιρέσεων των 11 ζητούμενων σημείων (11 σημεία σχηματίζουν 10 διαστήματα). Αρχικά, η συνάρτηση `simpsonMethod()` αρχικοποιεί τις μεταβλητές, και έπειτα τρέχει μία `for-loop` από `k=1` μέχρι και `k=N-1` για να υπολογίσει τα σημεία x σύμφωνα με τον τύπο $x_i = a + k \cdot h$ όπου $a = 0$, k είναι ο δείκτης της `for-loop` και $h = \frac{b-a}{N}$. Η `for-loop` υπολογίζει, επίσης, τα δύο αθροίσματα $\sum_{i=1}^{\frac{N}{2}-1} f(x_{2i})$ και $\sum_{i=1}^{\frac{N}{2}} f(x_{2i-1})$. Έπειτα, όταν τερματίσει η `for-loop`, η συνάρτηση υπολογίζει την προσέγγιση σύμφωνα με τον γνωστό τύπο της μεθόδου Simpson:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{3N}(f(x_0) + f(x_N) + 2\sum_{i=1}^{\frac{N}{2}-1} f(x_{2i}) + 4\sum_{i=1}^{\frac{N}{2}} f(x_{2i-1})) = 1.000003$$

Μέθοδος Τραπεζίου

Ο κώδικας καλεί την συνάρτηση `void trapezoidalMethod(double a, double b, int N)`, που υλοποιεί τη μέθοδο τραπεζίου για τη προσέγγιση του $\int_0^{\frac{\pi}{2}} \sin(x)\,dx$. Τα `a` και `b` αποτελούν τα όρια του ολοκληρώματος, 0 και $\frac{\pi}{2}$ αντίστοιχα, ενώ η μεταβλητή `N` έχει αρχικοποιηθεί στην `main()` με $N = 10$ και αποτελεί τον αριθμό των υποδιαιρέσεων των 11 ζητούμενων σημείων (11 σημεία σχηματίζουν 10 διαστήματα). Αρχικά, η συνάρτηση `trapezoidalMethod()` αρχικοποιεί τις μεταβλητές, και έπειτα τρέχει μία `for-loop` από `k=1` μέχρι και `k=N-1` για να υπολογίσει τα σημεία x σύμφωνα με τον τύπο $x_i = a + k \cdot h$ όπου $a = 0$, k είναι ο δείκτης της `for-loop` και $h = \frac{b-a}{N}$. Η `for-loop` υπολογίζει, επίσης, το άθροισμα $\sum_{i=1}^{N-1} f(x_i)$. Έπειτα, όταν τερματίσει η `for-loop`, η συνάρτηση υπολογίζει την προσέγγιση σύμφωνα με τον γνωστό τύπο της μεθόδου τραπεζίου:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2N}(f(x_0) + f(x_N) + 2\sum_{i=1}^{N-1} f(x_i)) = 0.997943$$

Σφάλματα Μεθόδων

Αρχικά, ισχύει ότι $\int_0^{\frac{\pi}{2}} \sin(x)\,dx = 1$. Επίσης, ισχυέι ότι $f''(x) = f^{(4)}(x) = sinx$ και το μέγιστο της δεύτερης και τέταρτης παραγώγου στο διάστημα $[\mathbf{0}, \frac{\pi}{\mathbf{2}}]$ ισούται με 1, άρα $M = \mathbf{max}(|f''(x)|, x \in [0, \frac{\pi}{2}]) = \mathbf{max}(|f^{(4)}(x)|, x \in [0, \frac{\pi}{2}]) = 1$. Ας υπολογίσουμε τα σφάλματα της κάθε μεθόδου:

Μέθοδος Simpson:
Το θεωρητικό σφάλμα, σύμφωνα με τον γνωστό τύπο, είναι το εξής:
$|e| \leq \frac{(b-a)^3}{12N^2} \cdot M = \frac{\pi^3}{9600} \approx 0.003$

Το αριθμητικό σφάλμα ισούται με $|1 - 1.000003| \approx 0.000003$.

Μέθοδος Τραπεζίου:
Το θεωρητικό σφάλμα, σύμφωνα με τον γνωστό τύπο, είναι το εξής:
$|e| \leq \frac{(b-a)^5}{180N^4} \cdot M = \frac{\pi^5}{57600000} \approx 5.31 \cdot 10^{-6}$

Το αριθμητικό σφάλμα ισούται με $|1 - 0.997943| \approx 0.002057$.

Τρίτη Άσκηση

Γενικά σχόλια για τους κώδικες της άσκησης

Για την συγκεκριμένη άσκηση, **δεν χρησιμοποιήθηκε καθόλου κάποιο γλωσσικό μοντέλο για την διευκόλυνση της γραφής του δικού μου κώδικα.** Παρ'όλα, αυτά για την γραφή του αρχείου `ymd_converter.c`, χρησιμοποιήθηκε έτοιμος κώδικας ανερτημένος στο **Stack Overflow** (https://stackoverflow.com/a/65877308), ο οποίος θα εξηγηθεί. Οι κώδικες υπάρχουν στον φάκελο 7.

Έτοιμος κώδικας

Οι ημερομηνίες κλεισίματος, αποτελούν τον άξονα x της γραφικής παράστασης της τιμής κλεισίματος. Όμως, οι ημερομηνίες πρέπει με κάποιον τρόπο να γραφτούν ως αριθμοί. Αυτό για να γίνει στη C είναι λίγο πολύπλοκο οπότε χρησιμοποιήθηκε ο έτοιμος κώδικας του προαναφερθόντος συνδέσμου:

```c
time_t YMD_to_time(const char *ymd) {
    if (ymd == NULL) {
        return (time_t)-1;
    }

    struct tm tm = {0}; // Important: initialize all members to 0
    int n = 0;
    sscanf(ymd, "%4d-%2d-%2d %n", &tm.tm_year, &tm.tm_mon, &tm.tm_mday, &n);

    // Scan incomplete or extra junk?
    if (n == 0 || ymd[n]) {
        return (time_t)-1; // Mal-formed string
    }

    // Could add extra checks for months/days outside primary range, spaces in string, etc.

    // Adjust ranges as struct tm uses different references
    tm.tm_year -= 1900;
    tm.tm_mon--;
    tm.tm_isdst = -1; // Important to get right time for Year-Month-Day 00:00:00 _local time_

    // The following conversion assumes tm is in local time.
```

```
23      return mktime(&tm); // This may return -1;
24  }
```

Αρχικά, περιέχει δύο `if statements` που ελέγχουν αν υπάρχει λάθος στο format της εισόδου (`ymd`) το οποίο πρέπει να είναι της μορφής yyyy-mm-dd. Επίσης, αποσυνθέτει την είσοδο και αναθέτει τη χρονία (yyyy), τον μήνα (mm) και τη μέρα (dd) στα πεδία year, mon και day της μεταβλητής tm που είναι τύπου struct tm, το οποίο είναι built-in. Επίσης, θέτει το flag-πεδίο `isdst` με -1, για να υποδηλώσει ότι η συνάρτηση `time()` πρέπει να καθορίσει αυτόματα εάν η θερινή ώρα (Daylight Saving Time - DST) ισχύει για τον καθορισμένο χρόνο.Έτσι, επιτρέπεται στη συνάρτηση `time()` να ρυθμίσει αυτόματα την ώρα για τη θερινή ώρα ανάλογα με τους κανόνες της τοπικής χρονικής ζώνης. Τέλος, η μεταβλητή `tm` καλείται από τη συνάρτηση `time()` με αναφορά, για να μετατρέψει τη μεταβλητή τύπου tm σε έναν ακέραιο, ο οποίος δηλώνει έναν αριθμό δευτερολέπτων από την έναρξη της περιόδου (Epoch) στις 00:00:00 UTC, 1 Ιανουαρίου 1970. Γι'αυτό τον λόγο, πιο πριν, η μεταβλητή `tm.tm_year` μειώνεται κατά 1900 και η μεταβλητή `tm.tm_mon` μειώνεται κατά 1 (Ιανουάριος). Η δήλωση αυτής της συνάρτησης υπάρχει στο αρχείο `ymd_converter.h`

Εισαγωγή

Μελετήθηκαν οι τιμές κλεισίματος δύο κρυπτονομισμάτων, του **XRP** (https://finance.yahoo.com/quote/XRP-USD/history?p=XRP-USD) και του **Ethereum** (https://finance.yahoo.com/quote/ETH-USD?p=ETH-USD). Για συντομογραφία το τελευταίο θα αναφέρεται ως **ETH**. Οι ημερομηνίες από τις υποίες συλλέχθηκαν οι τιμές κλεισείματος είναι οι εξής: 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 Νοεμυβρίου 2023. Οι ημερομηνίες για τις οποίες θα γίνει πρόβλεψη είναι 28 Νοεμβρίου 2023 και 5 συνεδριάσεις μετά, δηλαδή 2 Δεκεμβρίου 2023. Οι τιμές κλεισίματος του XPR για τις ημερομηνίες που συλλέχθηκαν πληροφορίες είναι 0.613717, 0.611189, 0.627499, 0.612842, 0.580462, 0.611899, 0.620242, 0.621881, 0.623444, 0.616819, ενώ για το το ETH είναι 1961.28, 1963.29, 2013.20, 2022.24, 1937.07, 2064.43, 2062.21, 2081.15, 2084.41, 2063.29.

Τρόπος υπολογισμού πολυωνύμων

Έστω $f(x)$ μια συνάρτηση για την οποία γνωρίζουμε 10 τιμές σε 10 διαφορετικά σημεία του άξονα x. Γράφτηκε κώδικας, για να υπολιγστεί η καλύτερη τετραγωνική καμπύλη ($y = ax^2 + bx + c$) **(1)** που μπορεί να προσεγγίσει την $f(x)$ με τη μέθοδο των ελαχίστων τετραγώνων. Για να συμβεί αυτό, σύμφωνα με τη θεωρία, θα κατασκευάσουμε ένα γραμμικό σύστημα, το οποίο θα είναι αδύνατο, και θα προσπαθήσουμε με τις κανονικές εξισώσεις να βρούμε τη λύση ελαχίστων τετραγώνων. Αρχικά, θα αντικαταστήσουμε κάθε ένα από τα 10 γνωστά κλεισίματα, δηλαδή τις 10 γνωστές τιμές f με γνωστά x, στην **(1)** και θα προκύψουν οι 10 εξισώσεις του γραμμικού συστήματος. Έτσι, θα έχουμε τον πίνακα $\mathbf{A}$ των συντελεστών, το διάνυσμα-στήλη $\mathbf{x}$ των αγνώστων και το διάνυσμα-στήλη $\mathbf{b}$ των σταθερών συντελεστών. Για να βρούμε τις κανονικές εξισώσεις, θα μετατρέψουμε την εξίσωση $\mathbf{Ax} = \mathbf{b}$ σε $\mathbf{A^T Ax'} = \mathbf{A^T b}$, όπου το διάνυσμα-στήλη $\mathbf{x'}$ αποτελεί τη λύση ελαχίστων τετραγώνων. Έτσι, λύνωντας το γραμμικό σύστημα, υπολογίζουμε τα a, b και c, και επομένως το πολυώνυμο $2^{ου}$ βαθμού. Με παρόμοιο τρόπο,

θα υπολογιστούν και τα πολυώνυμα $3^{ου}$ ($y = ax^3 + bx^2 + cx + d$) και $4^{ου}$ βαθμού ($y = ax^4 + bx^3 + cx^2 + dx + e$).

Συναρτήσεις helpers

Το αρχείο `helpers.h` περιέχει δήλωση συναρτήσεων, οι οποίες χρησιμοποιήθηκαν στην προηγούμενη εργασία, καθώς και παραλλαγές τους:

```
1  void multiplyMatrices(long double **firstMatrix, long
       double **secondMatrix, long double **result, int
       rowFirst, int columnFirst, int rowSecond, int
       columnSecond); // Function to multiply two matrices
2  void multiplyMV(long double **P, long double *b, long
       double *result, int rowFirst, int columnFirst, int
       rowSecond); // Function to multiply a matrix with a
        vector
3  void displayMatrix(long double ** matrix, int row, int
        column); // Function to display a matrix - for
       debugging
4  void displayVector(long double *vector, int row); //
       Function to display a vector
5  void swap(long double *x, long double *y, int n); //
       Function to swap two vectors
6  void initializeIn(long double **X, int n); // Function
        to initialize In matrix
7  void copy(long double **X, long double **Y, int n); //
        Function to copy two matrices
8  void transposeMatrix(long double **matrix, long double
        **result, int row, int col); // Function to
       calculate transposed matrix
9  void palu(long double **P, long double **A, long
       double **L, long double **U, int n); // Function to
        do the PA = LU factorization
10 void solveY(long double **L, long double *y, long
       double *b, int n); // Function to solve Ly = b with
        current b = Pb
11 void solveX(long double **U, long double *x, long
       double *y, int n); // Function to solve Ux = y
12 long double* solve(long double **P, long double **A,
       long double **L, long double **U, long double *b,
       long double *y, long double *x, int n); // Function
        to solve a linear system using the PA = LU
       factorization method
```

Οι υλοποιήσεις τους βρίσκονται στο αρχείο `helpers.c`

Τελικός κώδικας

Οι κώδικες για το πολυώνυμα $2^{ου}$, $3^{ου}$ και $4^{ου}$ βαθμού υπάρχουν στα αρχεία `two.c`, `three.c` και `four.c` αντίστοιχα. Ας αναλύσουμε τον τρόπο λειτουργίας του αρχείου `two.c`:

Αρχικά, αρχικοποιούνται οι ημερομηνίες για τις οποίες υπάρχουν δεδομένα, οι τιμές του κάθε νομίσματος ($\mathbf{y}$) για αυτές τις ημερομηνίες και με τη βοήθεια της `YMD_to_time()` υπολογίζονται και αποθηκεύονται οι ημερομηνίες ως αριθμοί ($\mathbf{x}$), αφού πρώτα διαιρεθούν με το 10^8, ώστε να μην είναι πολύ μεγάλοι (καθώς μετά θα χρειαστεί να πολλαπλασιαστούν εις το τετράγωνο, ακομά και εις την τετάρτη για το πολυώνυμο $4^{ου}$ βαθμού).
Έπειτα, αρχικοποιούνται οι πίνακες και τα διανύσματα, με τη χρήση της `malloc()`. Αφού γίνει αυτό, υπολογίζονται κατάλληλα ο πίνακας $\mathbf{A}$ και το διάνυσμα-στήλη $\mathbf{b}$ του αδύνατου συστήματος $\mathbf{Ax} = \mathbf{b}$. Στον πίνακα $\mathbf{A}$, η πρώτη στήλη έχει μόνο 1, η δεύτερη τα γνωστά σημεία του άξονα x και η τρίτη τα ίδια σημεία εις το τετράγωνο. Το διάνυσμα-στήλη $\mathbf{b}$ περιέχει τις γνωστές τιμές του αντίστοιχου κρυπτονομίσματος για την κάθε γνωστή ημερομηνία. Στη συνέχεια, με την κλήση helper συνάρτησεων, υπολογίζεται και αποθηκεύεται ο πίνακας $\mathbf{A^T}$. Έπειτα, γίνονται και αποθηκεύονται οι πολλαπλασιασμοί $\mathbf{A^TA}$ και $\mathbf{A^Tb}$, για καθένα από τα δύο κρυπτονομίσματα. Τέλος, λύνεται το γραμμικό σύστημα $\mathbf{A^TAx'} = \mathbf{A^Tb}$, δηλαδή οι κανονικές εξισώσεις, με τη μέθοδο της παραγοντοποιήσης $\mathbf{PA} = \mathbf{LU}$, που υλοποιήθηκε στο πρώτο υποερώτημα της 3ης άσκησης της προηγούμενης εργασίας. Τα αποτελέσματα είναι οι συντελεστές του $2^{ου}$ πολυωνύμου για το κάθε κρυπτονόμισμα.
Τέλος, προβλέπονται τα αποτελέσματα για τις ζητούμενες ημερομηνίες με τη βοήθεια της παρακάτω συνάρτησης:

```
1 void approx(long double *a, long double x, char* date,
      char* coinName) {
2     long double approx = a[0] + a[1]*x + a[2]*x*x;
3     printf("\nApproximation␣of␣%s␣for␣%s␣is:␣%Lf",
          coinName, date, approx);
4 }
```

Οι κώδικες για τον υπολογισμό των πολυωνύμων $3^{ου}$ και $4^{ου}$ βαθμού είναι ακριβώς οι ίδιοι με μικροαλλαγές που αφορούν τα μεγέθη των πινάκων.

Αποτελέσματα

Πολυώνυμο $2^{ου}$ βαθμού:

XPR
$p_2(x) = 682.049364x^2 - 23196.351444x + 197226.334259$

Για **γνωστές** ημερομηνίες:
Approximation of XPR for 2023-11-17 is: 0.616311 (Πραγματική: 0.613717)
Approximation of XPR for 2023-11-18 is: 0.613082 (Πραγματική: 0.611189)
Approximation of XPR for 2023-11-19 is: 0.610871 (Πραγματική: 0.627499)
Approximation of XPR for 2023-11-20 is: 0.609679 (Πραγματική: 0.612842)
Approximation of XPR for 2023-11-21 is: 0.609504 (Πραγματική: 0.580462)
Approximation of XPR for 2023-11-22 is: 0.610348 (Πραγματική: 0.611899)
Approximation of XPR for 2023-11-23 is: 0.612210 (Πραγματική: 0.620242)
Approximation of XPR for 2023-11-24 is: 0.615091 (Πραγματική: 0.621881)
Approximation of XPR for 2023-11-25 is: 0.618990 (Πραγματική: 0.623444)

Approximation of XPR for 2023-11-26 is: 0.623907 (Πραγματική: 0.616819)

Για **άγνωστες** ημερομηνίες:
Approximation of XPR for 2023-11-28 is: 0.636796 (Πραγματική: 0.611422)
Approximation of XPR for 2023-12-02 is: 0.674794 (Πραγματική: 0.620976)

ETH
$p_2(x) = -562665.730741x^2 + 19153450.684405x - 162996346.984219$

Για **γνωστές** ημερομηνίες:
Approximation of ETH for 2023-11-17 is: 1956.045304 (Πραγματική: 1961.28)
Approximation of ETH for 2023-11-18 is: 1973.665828 (Πραγματική: 1963.29)
Approximation of ETH for 2023-11-19 is: 1990.446298 (Πραγματική: 2013.20)
Approximation of ETH for 2023-11-20 is: 2006.386712 (Πραγματική: 2022.24)
Approximation of ETH for 2023-11-21 is: 2021.487070 (Πραγματική: 1937.07)
Approximation of ETH for 2023-11-22 is: 2035.747373 (Πραγματική: 2064.43)
Approximation of ETH for 2023-11-23 is: 2049.167621 (Πραγματική: 2062.21)
Approximation of ETH for 2023-11-24 is: 2061.747813 (Πραγματική: 2081.15)
Approximation of ETH for 2023-11-25 is: 2073.487950 (Πραγματική: 2084.41)
Approximation of ETH for 2023-11-26 is: 2084.388031 (Πραγματική: 2063.29)

Για **άγνωστες** ημερομηνίες:
Approximation of ETH for 2023-11-28 is: 2103.668027 (Πραγματική: 2049.34)
Approximation of ETH for 2023-12-02 is: 2132.147355 (Πραγματική: 2165.70)

Πολυώνυμο $3^{ου}$ **βαθμού**:

XPR
$p_3(x) = 1.192260x^3 + 621.211668x^2 - 22161.558557x + 191359.378177$

Για **γνωστές** ημερομηνίες:
Approximation of XPR for 2023-11-17 is: 0.616311 (Πραγματική: 0.613717)
Approximation of XPR for 2023-11-18 is: 0.613082 (Πραγματική: 0.611189)
Approximation of XPR for 2023-11-19 is: 0.610871 (Πραγματική: 0.627499)
Approximation of XPR for 2023-11-20 is: 0.609679 (Πραγματική: 0.612842)
Approximation of XPR for 2023-11-21 is: 0.609504 (Πραγματική: 0.580462)
Approximation of XPR for 2023-11-22 is: 0.610348 (Πραγματική: 0.611899)
Approximation of XPR for 2023-11-23 is: 0.612210 (Πραγματική: 0.620242)
Approximation of XPR for 2023-11-24 is: 0.615091 (Πραγματική: 0.621881)
Approximation of XPR for 2023-11-25 is: 0.618990 (Πραγματική: 0.623444)
Approximation of XPR for 2023-11-26 is: 0.623907 (Πραγματική: 0.616819)

Για **άγνωστες** ημερομηνίες:
Approximation of XPR for 2023-11-28 is: 0.636796 (Πραγματική: 0.611422)
Approximation of XPR for 2023-12-02 is: 0.674794 (Πραγματική: 0.620976)

ETH
$p_3(x) = -31505.652035x^3 + 1044552.825003x^2 - 8176581.405897x - 8084695.610897$

Για **γνωστές** ημερομηνίες:
Approximation of ETH for 2023-11-17 is: 1956.044922 (Πραγματική: 1961.28)
Approximation of ETH for 2023-11-18 is: 1973.665360 (Πραγματική: 1963.29)

Approximation of ETH for 2023-11-19 is: 1990.446020 (Πραγματική: 2013.20)
Approximation of ETH for 2023-11-20 is: 2006.386781 (Πραγματική: 2022.24)
Approximation of ETH for 2023-11-21 is: 2021.487520 (Πραγματική: 1937.07)
Approximation of ETH for 2023-11-22 is: 2035.748115 (Πραγματική: 2064.43)
Approximation of ETH for 2023-11-23 is: 2049.168446 (Πραγματική: 2062.21)
Approximation of ETH for 2023-11-24 is: 2061.748389 (Πραγματική: 2081.15)
Approximation of ETH for 2023-11-25 is: 2073.487823 (Πραγματική: 2084.41)
Approximation of ETH for 2023-11-26 is: 2084.386626 (Πραγματική: 2063.29)

Για **άγνωστες** ημερομηνίες:
Approximation of ETH for 2023-11-28 is: 2103.661850 (Πραγματική: 2049.34)
Approximation of ETH for 2023-12-02 is: 2132.119362 (Πραγματική: 2165.70)

Πολυώνυμο $4^{ου}$ **βαθμού**:

XPR
$p_4(x) = 0.069943x^4 + 13.837519x^3 - 145.178872x^2 - 8442.232450x + 111649.860996$

Για **γνωστές** ημερομηνίες:
Approximation of XPR for 2023-11-17 is: 0.616312 (Πραγματική: 0.613717)
Approximation of XPR for 2023-11-18 is: 0.613083 (Πραγματική: 0.611189)
Approximation of XPR for 2023-11-19 is: 0.610871 (Πραγματική: 0.627499)
Approximation of XPR for 2023-11-20 is: 0.609678 (Πραγματική: 0.612842)
Approximation of XPR for 2023-11-21 is: 0.609504 (Πραγματική: 0.580462)
Approximation of XPR for 2023-11-22 is: 0.610348 (Πραγματική: 0.611899)
Approximation of XPR for 2023-11-23 is: 0.612210 (Πραγματική: 0.620242)
Approximation of XPR for 2023-11-24 is: 0.615091 (Πραγματική: 0.621881)
Approximation of XPR for 2023-11-25 is: 0.618990 (Πραγματική: 0.623444)
Approximation of XPR for 2023-11-26 is: 0.623908 (Πραγματική: 0.616819)

Για **άγνωστες** ημερομηνίες:
Approximation of XPR for 2023-11-28 is: 0.636800 (Πραγματική: 0.611422)
Approximation of XPR for 2023-12-02 is: 0.674812 (Πραγματική: 0.620976)

ETH
$p_4(x) = -758.192557x^4 + 14523.174693x^3 + 11795.976489x^2 + 1930159.302449x - 44247183.585$

Για **γνωστές** ημερομηνίες:
Approximation of ETH for 2023-11-17 is: 1956.044244 (Πραγματική: 1961.28)
Approximation of ETH for 2023-11-18 is: 1973.665074 (Πραγματική: 1963.29)
Approximation of ETH for 2023-11-19 is: 1990.446073 (Πραγματική: 2013.20)
Approximation of ETH for 2023-11-20 is: 2006.387098 (Πραγματική: 2022.24)
Approximation of ETH for 2023-11-21 is: 2021.488006 (Πραγματική: 1937.07)
Approximation of ETH for 2023-11-22 is: 2035.748653 (Πραγματική: 2064.43)
Approximation of ETH for 2023-11-23 is: 2049.168896 (Πραγματική: 2062.21)
Approximation of ETH for 2023-11-24 is: 2061.748592 (Πραγματική: 2081.15)
Approximation of ETH for 2023-11-25 is: 2073.487597 (Πραγματική: 2084.41)
Approximation of ETH for 2023-11-26 is: 2084.385767 (Πραγματική: 2063.29)

Για **άγνωστες** ημερομηνίες:
Approximation of ETH for 2023-11-28 is: 2103.659032 (Πραγματική: 2049.34)

Approximation of ETH for 2023-12-02 is: 2132.109239 (Πραγματική: 2165.70)

Όπως βλέπουμε οι προσεγγίσεις μεταξύ των πολυωνύμων των τριών τάξεων, έχουν πολύ κοντινές προσεγγίσεις. Για παράδειγμα, οι προσεγγίσεις για το **XPR** των πολυωνύμων $2^{ου}$ και $3^{ου}$ βαθμού για τις δύο άγνωστες ημερομηνίες (2023-11-28 και 2023-12-02) είναι ακριβώς ίδες.

Επίσης, οι προσεγγίσεις, σχετικά με τις πραγματικές τιμές, είναι σχετικά κοντά, χωρίς να εξαρτάται αν η προσέγγιση είναι σε γνωστή ή άγνωστη ημερομηνία.

Παρ'όλα αυτά, η συγκεκριμένη μέθοδος "πρόβλεψης" δεν αρκεί για το συγκεκριμένο θέμα (πρόβλεψη τιμών κρυπτονομισμάτων), καθώς η ακρίβεια χρειάζεται να είναι μικρότερη.