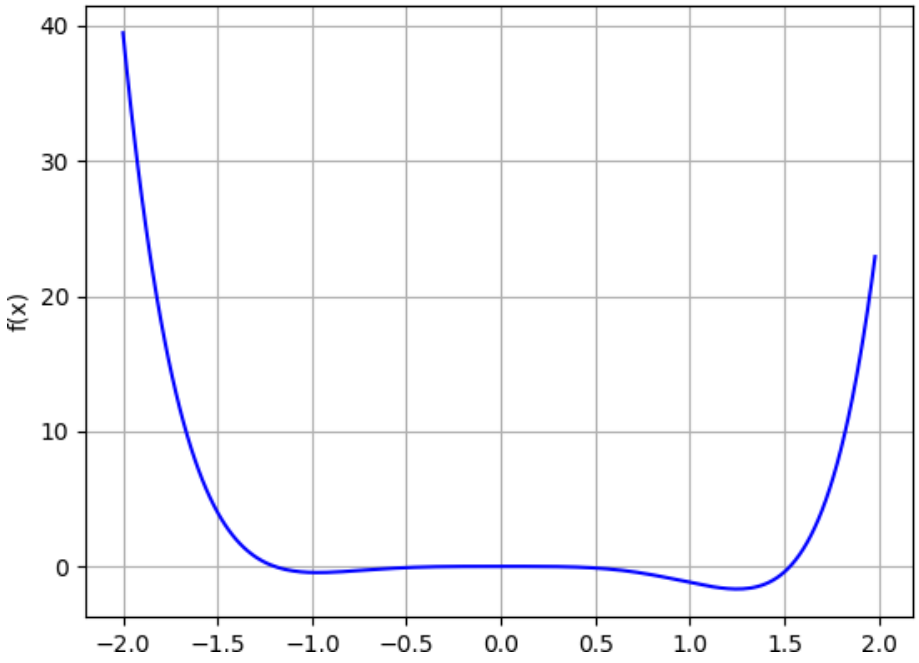
Υποχρεωτική Εργασία στην Αριθμιτική Ανάλυση

Ονοματεπώνυμο : Απόστολος Γιαννουλίδης

ΑΕΜ : 2906

Άσκηση 1)

Η γραφική παράσταση της συνάρτησης υλοποιήθηκε σε python (αρχείο plotask1.py) .



Παρατηρούμε ότι η συνάρτηση έχει τρεις ρίζες στο [-2,2] , εκ των οποίων η ρίζα που βρίσκεται κοντά στο μηδέν δεν μπορεί να υπολογισθεί με την μέθοδο διχοτόμησης αφού δεν ισχύει εκεί κοντά ο Bolzano.

Το πρόγραμμα εύρεσης ριζών είναι γραμμένο στην γλώσσα C . Το αρχείο βρίσκεται στον φάκελο ask1_aran και ο κώδικας είναι στο αρχείο main.c . Αρχικά φτιάχνω σε μορφή συναρτήσεων την f(x) την f'(x) (f1(x)) και την f''(x) (f2(x)) οι οποίες έχουν υπολογισθεί στο χαρτί .Έπειτα η halfmethod περιγράφει την μέθοδο διχοτόμησης, Newton_Raphson την Newton Raphson και η temnousa την μέθοδο της τέμνουσας. Επίσης η συνάρτηση g1 περιγράφει την g'(x) όπου $g(x) = x - \frac{f(x)}{f'(x)}$ στην main() ακολουθούν οι υπολογισμοί των ριζών .

Όσο αναφορά την μέθοδο τέμνουσας θέλουμε ακρίβεια στο 6ο δεκαδικό ψηφίο οπότε λύνουμε την ανίσωση , $2^n > \frac{10^7}{5}$ όπου n = 21 , και $0{,}7 * 2^n > \frac{10^7}{5}$, όπου n=19

Διάστιμα	Newton Raphson	Διχοτόμιση	Τέμνουσα	f'(x*)
[-2,-1]	8 επαναλύψεις Ρίζα :-1.197624	21 επαναλύψεις Ρίζα :-1.197623	14 επαναλύψεις Ρίζα :-1.197624	-4.920587
[1.3,2]	7 επαναλύψεις Ρίζα : 1.530134	19 επαναλύψεις Ρίζα: 1.530134	11 επαναλύψεις Ρίζα : 1.530134	14.972776
[-0.5,0.5]	90 επαναλύψεις Ρίζα : 0.000067	-	54 επαναλύψεις Ρίζα : -0.000065	0

Στον υπολογισμό της τελευταίας ρίζας έχουμε πολύ αργή σύγκλιση με newton Raphson και αυτό εξηγείται απο το γεγονός ότι η πρώτη παράγωγος της f είναι μηδέν κοντά στο μηδέν .

Η newton Raphson έχει τετραγωνική σύγκλιση στις πρώτες δύο καθώς η f''(x*) είναι διάφορη του 0. Ενώ στην ρίζα στο 0 όχι , καθώς f''(0) = 0 .Η επιλογή του αρχικού σημείο που περνάμε ως παράμετρο στην Newton Raphson γίνεται έτσι ώστε να ισχύει η σχέση f(x0) * f''(x0) > 0 .

Άσκηση 2)

1) Το πρόγραμμα που υλοποιεί τις μεθόδους που περιγράφονται στην Άσκηση 2 είναι γραμμένο στην γλώσσα python (στο αρχείο ask2.py) . Φτιάχνω συναρτήσεις f, f1=f' , f2 = f'' και r , q , s που περιγράφουν τα r , q, s της εκφώνησης και στην συνέχεια τους τροποποιημένους αλγορίθμους της άσκησης (halfmethod (διχοτόμηση) , Newton Raphson και Secand (τέμνουσα)) .Έπειτα υπολογίζω και τις 5 ρίζες της συνάρτησης .Οι αρχικοποιήσεις φαίνονται μέσα απο τα ορίσματα που δίνουμε στις συναρτήσεις . Στην ρίζα της συνάρτησης που βρίσκεται στο -0,666666 δεν ισχύει ο Bolzano άρα δεν μπορεί να δουλέψει η
μέθοδος της διχοτόμησης .
2) Αν εφαρμόσουμε τον αλγόριθμο β) 10 φορές στο ίδιο σημείο (συγκεκριμένα τον εφαρμόζω στο διάστημα [-2,-1]) θα παρατηρήσουμε ότι κάθε φορά συγκλίνει στην ίδια ρίζα με διαφορετικό αριθμό επαναλήψεων . Αυτό είναι κάτι που περιμένουμε αφού ο αλγόριθμος βασίζεται σε τυχαίους αριθμούς.
3) Συγκρίνουμε πειραματικά τις τροποποιημένες μεθόδους στην συνάρτηση της άσκησης 1) (αρχείο sygkrisi1m2.py)
Στο διάστημα [-2,-1] στην μέθοδο Newton Raphson παρατηρούμε μία αρκετά αισθητή διαφορά , έχουμε πιο γρήγορη σύγκλιση , με 5 επαναλήψεις στην Newton Raphson έναντί 8 επαναλήψεων που είχαμε με την αρχική μέθοδο .Επίσης στην μέθοδο της Τέμνουσας έχουμε 3 επαναλήψεις με την τροποποιημένη μέθοδο έναντι 14 επαναλήψεων στην πρώτη άσκηση .Το ίδιο συμπέρασμα βγάζουμε και στο διάστημα [1.3,2] . Αλλά όσο αναφορά την μέθοδο της διχοτόμησης , στην τροποποιημένη μέθοδο έχουμε πιο αργή σύγκλιση και στα δύο διαστήματα . Όμως δεν μπορούμε να βγάλουμε με

σιγουριά το αποτέλεσμα πως η τροποποιημένη είναι πιο αργή καθώς υπάρχει ο συντέλεσής της τυχαιότητας .

Άσκηση 3)

1) Ο αλγόριθμος επίλυσης γραμμικού συστήματος με την μέθοδο PA=LU υλοποιήθηκε σε python (αρχείο as3.py) .
   Σύντομη ανάλυση του κώδικα :
   Μέθοδος swap , αντιμεταθέτει δύο γραμμές ενός πίνακα (συγκεκριμένα την γραμμή pos με την γραμμή j) . Χρησιμοποιείται στην για το χτίσιμο του πίνακα P κυρίως .
   Η μέθοδος down_triangle(L,Y,b) λύνει το γραμμικό σύστημα LY=b όπου το L είναι κάτω τριγωνικός .
   Η μέθοδος up_triangle(U,X,Y) λύνει το γραμμικό σύστημα UX=Y όπου το U είναι άνω τριγωνικός .
   Η μέθοδος pa_lu(A1,b) λύνει το γραμμικό σύστημα A1x=b με την μέθοδο της PA=LU . Αρχικά αρχικοποιούνται κατάλληλα οι πίνακες που θα χρησιμοποιηθούν , και έπειτα υλοποιείται ο αλγόριθμος που χρησιμοποιήθηκε και στην επίλυση των ασκήσεων στις διαλέξεις . Πρώτα γίνεται εύρεση του max της j στήλης ώστε να έχουμε τον μεγαλύτερο συντελεστή στην διαγώνιο . Στην συνέχεια χρησιμοποιείται η swap για να γίνουν οι αλλαγές και έπειτα υπολογίζονται οι πίνακες U και L . Τέλος καλούνται οι μέθοδοι down_triangle και up_triangle για να παραχθεί ο τελικό αποτέλεσμα του πίνακα των αγνώστων χ.

2) Ο αλγόριθμος Cholesky υλοποιείται σε γλώσσα python στο αρχείο choleski.py . Ο αλγόριθμος Cholesky παίρνει ως είσοδο έναν συμμετρικό
   πίνακα A και επιστρέφει έναν κάτω τριγωνικό πίνακα L όπου η σχέση που τους ενώνει είναι εξής , $\mathrm{A} = \mathrm{L}^{\mathrm{T}}L$ έτσι το σύστημα Aχ=B γίνεται $\mathrm{L}^{\mathrm{T}}L\chi = B$ όπου είναι πιο εύκολο να λυθεί και με μικρότερη πολυπλοκότητα. Ο αλγόριθμος του Cholesky βασίζεται στην συμμετρικότητα του A για να πράξει τον L , συγκριμένα (εξήγηση για 3χ3 πίνακα) : $\mathrm{A} = \mathrm{L}^{\mathrm{T}}L$ =

$$\begin{pmatrix} L11 & 0 & 0 \\ L21 & L22 & 0 \\ L31 & L32 & L33 \end{pmatrix} \begin{pmatrix} L11 & L12 & L13 \\ 0 & L22 & L32 \\ 0 & 0 & L33 \end{pmatrix} =$$

$$\begin{pmatrix} L11^2 & & (simetric) \\ L21 * L11 & L21^2 + L22^2 & \\ L31 * L11 & L31 * L21 + L32 * L22 & L31^2 + L32^2 + L33^2 \end{pmatrix}$$

Οπότε απο αυτήν την σχέση έχουμε αρκετά δεδομένα να υπολογίσουμε τις παραμέτρους L.

3) Ο αλγόριθμος του Gauss-Siedel υλοποιήθηκε σε python (αρχείο gaussSiedel.py). Χρησιμοποιούνται οι μέθοδοι Make_A και Make_b για την αναπαραγωγή των πινάκων που ζητούνται απο την άσκηση στην επίλυση του γραμμικού συστήματος .

Στην μέθοδο gauss_Siedel(A,b) υλοποιείται ο αλγόριθμος του Gauss-Siedel έτσι όπως περιεγράφηκε στο μάθημα .(Σημείωση , όταν τον τρέχουμε για n=10000 είναι πολύ αργός)

Άσκηση 4)

1) Απόδειξη ότι ο πίνακας G είναι στοχαστικός :

Για κάθε στήλη j έχουμε το εξής άθροισμα :

$$\sum_{i=1}^{n} \frac{q}{n} + \frac{A(j,i)(1-q)}{nj} = \sum_{i=1}^{n} \frac{q}{n} + \sum_{i=1}^{n} \frac{A(j,i)(1-q)}{nj}$$

$$= n * \frac{q}{n} + \frac{(1-q)}{nj} * \sum_{i=1}^{n} A(j,i)$$

$$= q + \frac{(1-q)}{nj} * \sum_{i=1}^{n} A(j,i)$$

Όπου ο πίνακας A(j,i) έχει 1 εκεί που η ιστοσελίδα j δείχνει στην ιστοσελίδα i , άρα το άθροισμα της από 1 μέχρι n ταυτίζεται με το nj . Οπότε έχουμε για κάθε στήλη j :

$$q + \frac{(1-q)}{nj} * \sum_{i=1}^{n} A(j,i) = q + (1-q) * \frac{nj}{nj} = q - q + 1 = 1$$

2) Ο αλγόριθμος της μέθοδος των δυνάμεων υλοποιήθηκε με python (αρχείο asknumanal.py, σημείωση : όλα τα ερωτήματα της άσκησης 4 βρίσκονται σε αυτό το αρχείο)
Στην συνάρτηση Google υπολογίζεται ο στοχαστικός πίνακας G σύμφωνα με τους τύπους που δίνονται στην εκφώνηση .
Η συνάρτηση power_method υλοποιεί τον αλγόριθμο της μεθόδους των δυνάμεων έτσι όπως περιγράφεται στις διαφάνειες του μαθήματος και επιστρέφει το ιδιοδιάνυσμα της μέγιστης ιδιωτικής .
Η συνάρτηση make_sum_one κανονικοποιεί ένα διάνυσμα έτσι ώστε το άθροισμα των στοιχείων της να βγάζει 1 , στην περίπτωση μας πρέπει να κανονοικοποιήσουμε το ιδιοδιανυσμα που βρίσκουμε γιατί περιγράφει ποσοστά .Το διάνυσμα p που υπολογίζεται απο τον αλγόριθμο είναι πολύ κοντά στο διάνυσμα που δίνεται και στην εκφώνηση με ένα σφάλμα της τάξης του 10^{-4} .
3) Επιλέγω να αυξήσω τον βαθμό σημαντικότητας του κόμβου 4 , έτσι αφαιρώ το 4 -> 12 , και προσθέτω τις ακμές : 15->4 , 14-> 4 . 10- >4 και 11-> 4 , με το σκεπτικό ότι οι ιστοσελίδες 15,14,11,10 έχουν το μεγαλύτερο βαθμό
σημαντικότητας έτσι δημιουργώντας ακμές απο αυτές προς την 4 μεταβιβάζω αυτόν τον βαθμό σημαντικότητας στον 4 κόμβο και αφαιρώντας το 4->12
προκαλώ το να μην μεταπηδήσει αυτος ο βαθμός σημαντικότητας
που πέτυχα για το 4 στον κόμβο 12 . Το αποτέλεσμα είναι το 4 να πάει απο

0.02682457 σε 0.13898914 , παρατηρούμε ότι ανέβηκε και η σημαντικότητα του κόμβου 2 , γιατι ο 4 δείχνει σε αυτόν .

4) Όταν αλλάζουμε την πιθανότητα q=0.02 αυτό ποιοτικά σημαίνει οτι ο χρήστης δεν κατευθύνεται τυχαία σε ιστοσελίδες αλλα μόνο μέσα απο την ιστοσελίδα που βρίσκεται εκείνη την στιγμή , πράγμα που σημαίνει οτι το που δείχνουν οι κόμβοι έχει μεγαλύτερη βαρύτητα πλέον άρα τα ποσοστά του πίνακα p απομακρύνονται μεταξύ τους , η απόσταση μεταξύ των ιστοσελίδων που έχουν πολλούς κόμβους που εισέρχονται σε αυτούς μεγαλώνει και
αυτών που έχουν λίγους , μεγαλώνει .
Ενώ αντίθετα όταν κάνουμε το q = 0.6 αυτό ποιοτικά σημαίνει ο ο χρήστης πηγαίνει πλέον σε σελίδες με τυχαίο τρόπο και όχι μέσα απο τα link που έχουν οι ιστοσελίδες που βρίσκεται κάθε δεδομένη στιγμή . Σαν αποτέλεσμα αυτό εχει να δίνουμε λιγότερη σημασία στις ακμές μεταξύ των κόμβων , οπότε τα ποσοστά ειναι πλέον πιο κοντά μεταξύ τους , τα μεγάλά ποσοστά μειώνονται και τα μικρά μεγαλώνουν .
5) Η τακτική αυτή έχει ως αποτέλεσμα μια αύξηση στην τάξη του κόμβου 11 κατά 2% πάνω σε σχέση με την προηγούμενη τάξη , σε ποσοστό έχουμε 20% αύξηση .Μπορούμε να πούμε ότι δούλεψε , καθώς τα ποσοστά δεν απέχουν πολύ μεταξύ τους , άρα το 2% είναι μια ουσιώδης αύξηση .
6) Αν αφαιρέσουμε τον κόμβο 10 αυτό που πετυχαίνουμε είναι (παρακάτω είναι το αρχικό διάνυσμα p και αυτό μετά την αφαίρεση του 10 κόμβου)
   1) [0.02682457] [0.04709499]
   2) [0.02986108] [0.04091142]
   3) [0.02986108] [0.03593562]
   4) [0.02682457] [0.03207005]
   5) [0.03958722] [0.04280082]
   6) [0.03958722] [0.04139102]
   7) [0.03958722] [0.05165866]
   8) [0.03958722] [0.05024885]
   9) [0.07456438] [0.04822346]
   10)[0.10631995]
   11)[0.10631995] [0.1709627]
   12)[0.07456438] [0.10359814]
   13)[0.12509164] [0.0411619]
   14)[0.11632789] [0.10746218]
   15)[0.12509164] [0.18648019]]

   Αρχικά παρατηρούμε ότι έχουν αυξηθεί ελάχιστα όλες οι τάξεις ,αυτό συμβαίνει γιατί έχουμε μία λιγότερη ιστοσελίδα άρα ο ποσοστό που μοιράζονται οι εναπομείναντα σελίδες ειναι μεγαλύτερο .
   Πιο ειδικά παρατηρούμε ότι οι ιστοσελίδες που έδειχνε το 10 , όπως το 13 μειώνονται δραματικά , το οποίο συμβαίνει γιατι πλέον αυξημένη τάξη που είχε το 10 λόγο του ότι είχε πολλές εισερχόμενες ακμές , πλέον δεν μεταπηδάν στην 13 .

Επίσης πλέον το 11 παίρνει όλη την πιθανότητα μεταπήδησης σε αυτόν το κόμβο καθώς κόμβοι όπως οι 6 και 7 δεν δείχνουν πλέον στο 10 και στο 11 αλλα μόνο στο 11 , πράγμα που αυξάνει και την τάξη του 15 , για τον λόγο ότι ο 11 δείχνει σε αυτόν .

Άσκηση 5)

Σημεία (10 τιμές του ημιτόνου) :

x	-π	-2.23	$-\frac{\pi}{2}$	-0.36	0	0.69	$\frac{\pi}{2}$	2.19	2.79	π
y	0.0	-0.79	-1	-0.352	0	0.637	1	0.814	0.344	0

Οι προσεγγιστικοί αλγόριθμοι γίναν στην γλώσσα python (αρχείο ask5.py) .

Η συνάρτηση par υλοποιεί την μέθοδο παρεμβολής με τον πίνακα του Newton ,έτσι όπως περιγράφεται στις διαφάνειες και στις σημειώσεις .

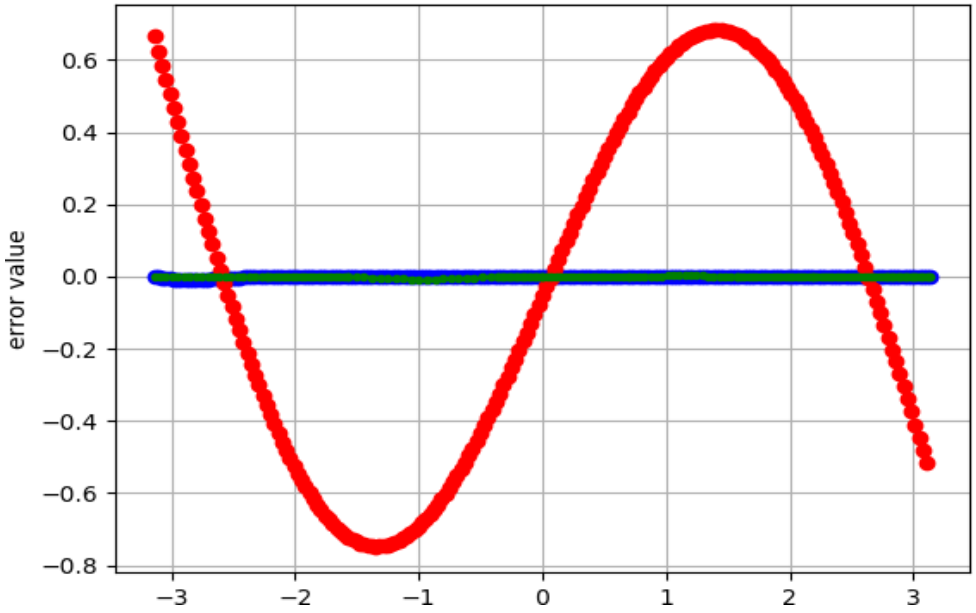
Η συνάρτηση Splines , υλοποιεί τον αλγόριθμο natural cubic Splines έτσι όπως περιγράφεται στην Wikipedia.

Η συνάρτηση lq , υλοποιεί τον αλγόριθμο ελάχιστον τετραγώνων όπου επιστρέφει τους συντελεστές δευτεροβάθμιας εξίσωσης .

Χρησιμοποιούνται οι συναρτήσεις calculatePx(A,X,x) όπου υπολογίζει την τιμή του χ μέσα απο την συνάρτηση που έχουμε ως αποτέλεσμα από την μέθοδο της παρεμβολής .

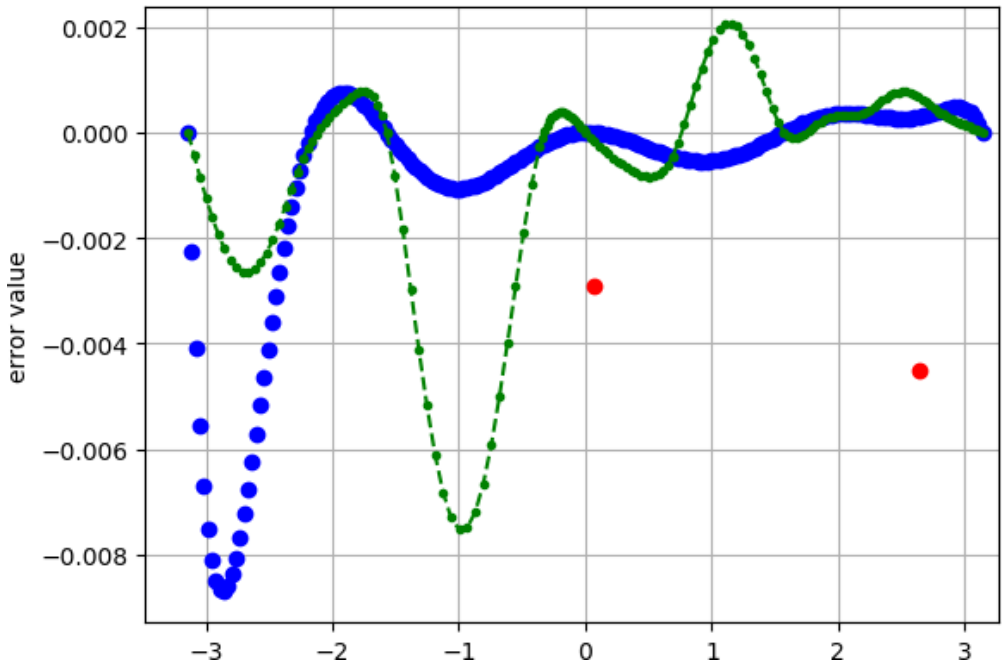
Calculateflq(X,xn) όπου υπολογίζει τις τιμές του χ σύμφωνα με την συνάρτηση που έχουμε ως αποτέλεσμα απο την lq .

Στα παρακάτω διαγράμματα με μπλε χρώμα είναι το σφάλμα απο την μέθοδο παρεμβολής ,με πράσινο χρώμα είναι το σφάλμα απο την μέθοδο Splines,με κόκκινο χρώμα είναι το σφάλμα απο την μέθοδο ελάχιστον τετραγώνων .



Συγκεκριμένα το μεγαλύτερο σφάλμα με την μέθοδο παρεμβολής είναι μικρότερο του 0.009 κατα απόλυτη τιμή

Και στην το μεγαλύτερο σφάλμα με την μέθοδο splines είναι μικρότερο του 0.008 κατα απόλυτη τιμή ,



Όσο αναφορά την μέθοδο των ελαχίστων τετραγώνων έχουμε μέγιστο σφάλμα κοντά στο 0.7 .

Άσκηση 6)

Οι αλγόριθμοι , μέθοδος των τραπεζίων και Simpson , υλοποιούνται με την γλώσσα python (αρχείο ask6a.py) .

Οι αλγόρυθμοι υλοποιούνται με τους τύπους που περιγράφτηκαν στο μάθημα και στις διαφάνειες .

Τα σημεία που επιλέχθηκαν :

x	0	$\frac{\pi}{8}$	$\frac{\pi}{4}$	$\frac{3\pi}{8}$	$\frac{\pi}{2}$
y	0	$\sin(\frac{\pi}{8})$	$\sin(\frac{\pi}{4})$	$\sin(\frac{3\pi}{8})$	$\sin(\frac{\pi}{2})$

Στην συνέχεια υπολογίζουμε τις τιμές του ολοκληρώματος και με τις δύο μεθόδους , αλλά επίσης και το σφάλμα με τον τύπο που δίνεται στις διαφάνειες .

Μέθοδος	αποτέλεσμα	Σφάλμα	θεωριτικό Σφάλμα
Τραπεζίων	0.987115800973	0.0128841990272	0.020186378047070193
Simpson	1.00013458497	0.000134584974194	0.0002075328808493933

Παρατηρούμε ότι το θεωριτικό σφάλμα σε κάθε περίπτωση είναι μεγαλύτερο απο το πραγματικό .

Άσκηση 7)

Τα γενέθλια μου : 1/6

Οι αλγόριθμοι γράφτηκαν σε γλώσσα python , (αρχείο ask7.py) .

Πίνακας των δεδομένων :

Ημερομινία	ΕΤΕ	ΑΛΦΑ
31/5/2017	0.311	2,05
30/5/2017	0.308	2,08
29/5/2017	0.315	2,13
26/5/2017	0.32	2,17
25/5/2017	0.317	2,15
24/5/2017	0.305	2,04
23/5/2017	0.33	2,14
22/5/2017	0.334	2,17
19/5/2017	0.333	2,21
18/5/2017	0.324	2,22

Κάνω πρόβλεψη για τις ημερομηνίες:

Αποτελέσματα για ΕΤΕ :

Ημερομηνία	$2^{ου}$ βαθμου	$3^{ου}$ βαθμου	$4^{ου}$ βαθμου	Πραγματικα
2/6/2017	0.304389393939	0.326518181818	0.24775	0,3320
8/6/2017	0.297934848485	0.386524009324	-0.130062354313	0,3200

Παρατηρούμε ότι με $4^{ου}$ βαθμου αποκλείνου πολυ οι τιμές απο τις πραγματικές , ενω η δευτέρου και η τρίτου βαθμου ειναι πιο κοντά .

Αποτελέσματα για ΑΛΦΑ:

Ημερομηνία	$2^{ου}$ βαθμου	$3^{ου}$ βαθμου	$4^{ου}$ βαθμου	Πραγματικα
2/6/2017	2.06692424242	1.85190909091	1.53204545455	2,1800
8/6/2017	2.06074242424	1.19996270396	-0.8978030303	2,0800

Εδώ επίσης η $2^{ου}$ βαθμού δείχνει να προβλέπει καλύτερα για μακρινές τιμές σε σχεση με αυτές που έχουμε στα δεδομένα μας από τις $3^{ου}$ και $4^{ου}$ βαθμού . Αυτό γιατί η καμπιλότητα των άλλων δυο τις κάνει να αποκλείνουν έξω απο το περιθόριο των δεδομενων μας , εκεί που θέλουμε να κάνουμε προβλέψεις δηλαδη .