1η Υποχρεωτική Εργασία
Στο Μάθημα της Αριθμητικής Ανάλυσης

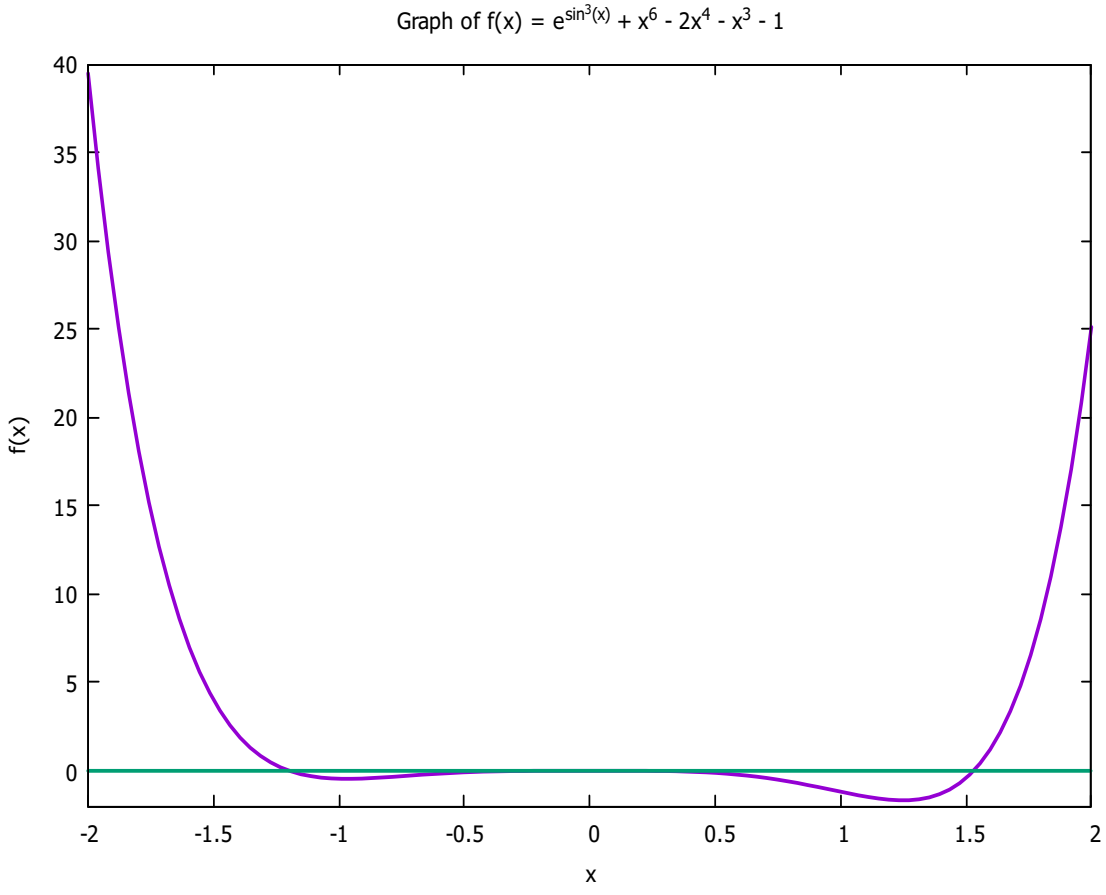
Όνοματεπώνυμο: Γεώργιος Κεσογλίδης
ΑΕΜ: 3911

29 Δεκεμβρίου 2022

Η υλοποίηση όλων των ασκήσεων πραγματοποιήθηκε στην python.

1 Άσκηση 1

Σχεδιάζουμε την γραφική παράσταση με χρήση του gnuplot:



Για την συνάρτηση f(x) εύκολα υπολογίζεται ότι έχει ρίζα στο x = 0. Επίσης, από το παραπάνω γράφημα βλέπουμε ότι έχει μόνο αλλές δύο ρίζες κόντα στο x = 1.5 και στο x = - 1.25. Θα υπολογίσουμε αύτες τις ρίζες με ακρίβεια πέντε δεκαδικών ψηφίων χρησιμοποιώντας τρεις διαφορετικές επαναληπτικές μεθόδους. Για τον υπολογισμό εκθετικών και τριγωνομετρικών συναρτήσεων χρησιμοποιούμε την βιβλιοθήκη numpy.

1.1 Μέθοδος της διχοτόμησης

Για την μέθοδο της διχοτόμησης θα επιλέξουμε τα διαστήματα [-2,-1] και [1, 2]. Από την γραφική παράσταση επιβεβαιώνουμε ότι ισχύουν οι προύποθεσεις του Θεωρήματος Bolzano. Δηλαδή, ότι η συνάρτηση είναι συνεχείς και έχει ετερόσημες τιμές στα άκρα των διαστημάτων που αναφέραμε. Δεν μπορούμε να εντοπίσουμε ότι η συνάρτηση έχει ρίζα στο x = 0 γιατί δεν έχει ετερόσημες τιμές κοντά της.
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For the interval [-2,-1] bisection loops 17 times
and the root is -1.1976242065429688
For the interval [1, 2] bisection loops 17 times
and the root is 1.5301284790039062

1.2 Μέθοδος Newton-Raphson

Για την μέθοδο Newton-Raphson θα πάρουμε ως αρχική τιμή πρώτα x = -1.75 , μετά x = 0.3 και τέλος x = 1.75. Μπορούμε να διαλέξουμε αυτές τις αρχικές τιμές διότι η συνάρτηση κοντά στο x = 0.3 είναι αρνητική και κοίλη ενώ για τις αλλές είναι θετική και κυρτή, άρα για όλες ισχύει

$$f(x_0)f"(x_0) > 0$$

Επίσης, από την θεωρία γνωρίζουμε ότι η μέθοδος Newton-Raphson συγκλίνει τετραγωνικά για τις ρίζες που η πρώτη και δεύτερη παράγωγος δεν μηδενίζονται για κάποιο διάστημα που περιέχει την ρίζα και την αρχική τιμή. Από την γραφική παράσταση της f παρατηρούμε ότι αυτό δεν ισχύει για την x = 0 διότι είναι τοπικό μέγιστο και άρα έχει παράγωγο 0. Ενώ για τις άλλες ρίζες υπάρχουν διαστήματα που δεν έχουν ούτε σημεία καμπής ούτε τοπικά ακρότατα οπότε συγκλίνουν τετραγώνικα. Περιμένουμε η ακρίβεια να επιτευχθεί πολύ πιο αργά για x = 0.
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For x = -1.75 Newton-Raphson loops 7 times
and the root is -1.1976237221338035
For x = 0.3 Newton-Raphson loops 30 times
and the root is 8.576575582112126e-05
For x = 1.75 Newton-Raphson loops 5 times

and the root is 1.5301335081746195

Να τονίσουμε ότι από εδώ και πέρα, για να αποφύγουμε τον υπολογισμό της παραγώγου με το χέρι, Θα χρησιμοποιούμε τον συμμετρικό ορισμό της παραγώγου, δηλαδή

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

για h που είναι ίσο με το σφαλμά μας ώστε να μην επηρεάσει την εύρεση λύσης. Συγκεκριμένα

$$h = 10^{-5}$$

1.3 Μέθοδος της τέμνουσας

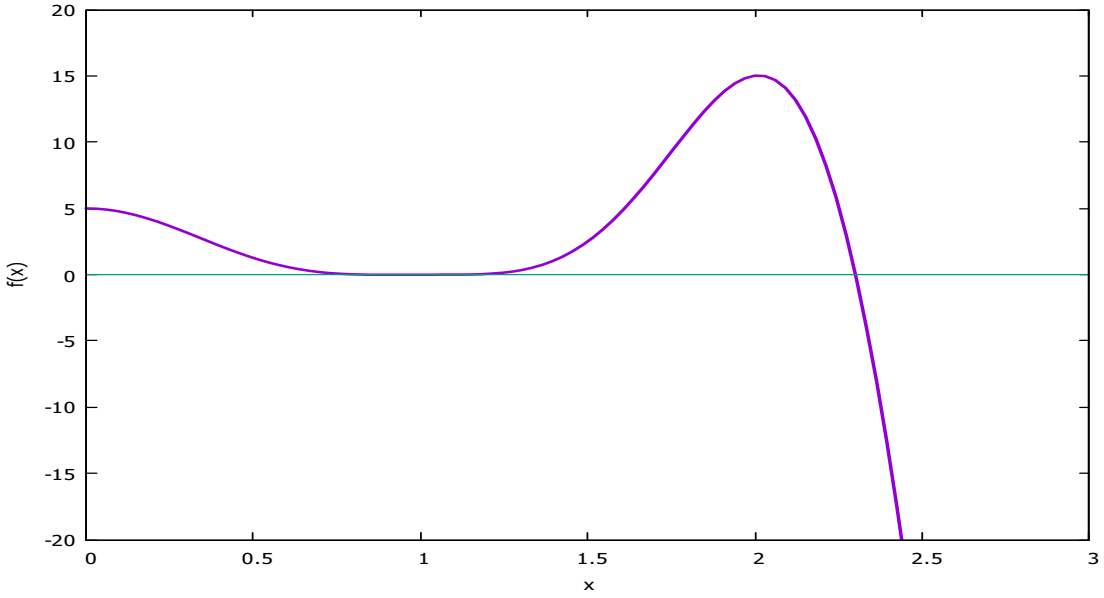
Η μέθοδος της τέμνουσας απαίτει τις ίδιες προϋποθέσεις με την μέθοδο Newton-Raphson. Για ευκολία κάθε x0 πού θα χρησιμοποιήσουμε θα είναι από τις παραπάνω αρχικές τιμές και για x1 θα πάρουμε κάποιο κοντινό αριθμό που τηρεί τις ίδιες προϋποθέσεις. Θα πάρουμε λοιπόν πρώτα x0 = -1.75, x1 = 2, μετά x0 = 0.3, x1 = 0.5 και τέλος x0 = 1.75, x1 = 2.
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For x0 = -1.75 and x1 = -2 secant loops 10 times
and the root is -1.1976237221353676
For x0 = 0.3 and x1 = 0.5 secant loops 41 times
and the root is 9.870092240214844e-05
For x0 = 1.75 and x1 = 2 secant loops 7 times
and the root is 1.5301335086057053

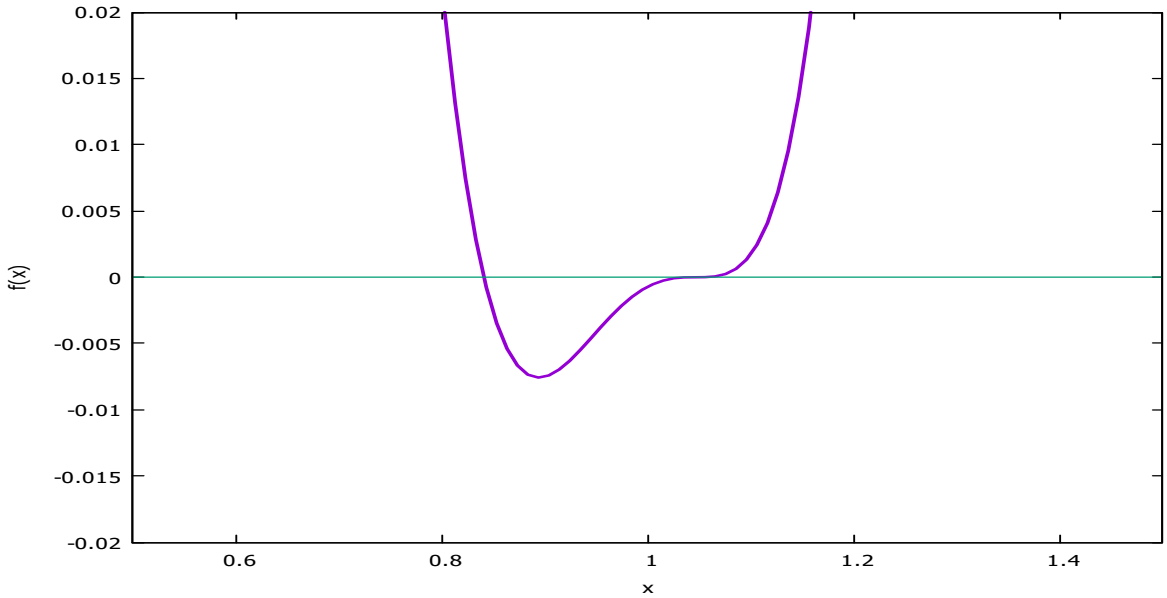
2 Άσκηση 2

2.1 Ρίζες f(x)

Σχεδιάζουμε την γραφική παράσταση της συνάρτησης ώστε να δούμε περίπου που βρίσκονται οι ρίζες



Δεν φαίνεται καθαρά τι γίνεται κοντά στο 1 οπότε μεγενθύνουμε την γραφική παράσταση



Βλέπουμε οτι η f έχει τρεις ρίζες και τώρα θα τις υπολογίσουμε

2.1.1 Τροποποιήμενη μέθοδος Newton-Raphson Μέθοδος Halley

Για να αποφύγουμε διαίρεση με πολύ μικρό αριθμό θα γράψουμε τον τύπο ως εξής:

$$x_{n+1} = x_n - \frac{1}{\frac{f'(x_n)}{f(x_n)} - \frac{1}{2}\frac{f"(x_n)}{f(x_n)}} => x_{n+1} = \frac{2f(x_n)f'(x_n)}{2f^2(x_n) - f(x_n)f"(x_n)}$$

Διαλέγουμε x = 0.5, μετά x = 1 και τέλος x = 2.5 επειδή ικανοποιούν τα κριτήρια που προαναφέρθηκαν στην 1.2.
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For x = 0.5 Halley loops 5 times
and the root is 0.8410686705678356
For x = 1 Halley loops 12 times
and the root is 1.047185946044355
For x = 2.5 Halley loops 4 times
and the root is 2.300523983021863

2.1.2 Μέθοδος τυχαίας διχοτόμησης

Για τον υπολογισμό τυχαίου σημείου μέσα στο διάστημα χρησιμοποιούμε τη συνάρτηση uniform της βιβλιοθήκης random.
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For the interval [0,1] randsection loops 13 times
and the root is 0.8410390768456916
For the interval [1,2] randsection loops 24 times
and the root is 1.0472775423679617
For the interval [2,3] randsection loops 16 times
and the root is 2.3006581620827893

2.1.3 Τροποιήμενη μέθοδος τέμνουσας Μέθοδος Muller

Για αρχικές τιμές θα διαλέξουμε x0 = 0, x1 = 0.25, x2 = 0.5, μετά x0 = 1, x1 = 1.1, x2 = 1.2 και τέλος x0 = 3, x1 = 2.75 and x2 = 2.5
Τα αποτελέσματα του προγράμματος είναι τα εξής:

For x0 = 0, x1 = 0.25 and x2 = 0.5 Muller loops 10 times
and the root is 0.8410686705648635
For x0 = 1, x1 = 1.1 and x2 = 1.2 Muller loops 22 times
and the root is 1.0471767047051557
For x0 = 3, x1 = 2.75 and x2 = 2.5 Muller loops 6 times
and the root is 2.300523983021863

2.2 Εκτελώ το 2.1.2 για 10 επαναλήψεις

Ο αριθμός επαναλήψεων είναι ο εξής:
1. 12 2. 24 3. 21 4. 10 5. 19 6. 23 7. 17 8. 22 9. 20 10. 25
Όπως βλέπουμε μερικές φορές η τυχαία διχοτόμηση συγκλίνει γρηγορότερα και μερικές αργότερα. Άρα δεν συγκλίνει πάντα σε ίδιο αριθμό επαναλήψεων

2.3 Σύγκριση μεθόδων

Τα αποτελέσματα της μεθόδου Newton-Raphson είναι:
For x = 0.5 Newton-Raphson loops 8 times
and the root is 0.8410686705678766
For x = 1 Newton-Raphson loops 19 times
and the root is 1.0471782059890036
For x = 2.5 Newton-Raphson loops 5 times
and the root is 2.300523983021863
Άρα η μέθοδος Halley είναι γρηγορότερη.

Τα αποτελέσματα της κανονικής διχοτόμησης είναι:
For the interval [0,1] bisection loops 17 times
and the root is 0.8410720825195312
For the interval [1,2] bisection loops 17 times
and the root is 1.0472335815429688
For the interval [2,3] bisection loops 17 times
and the root is 2.3005294799804688
Παρατηρούμε ότι η τυχαία διχοτόμηση μπορεί να συγκλίνει γρηγορότερα ή αργότερα από την κανονική, ανάλογα από το τυχαίο σημείο. Ενώ η σύγκλιση της κανονικής επηρεάζετε μόνο από το μέγεθος του διαστήματος. Εκτελούμε την μέθοδο τέμνουσας με αρχικές τιμές τα x0, x2 της 2.1.3.

Τα αποτελέσματα της μεθόδου τέμνουσας είναι:
For x0 = 0 and x1 = 0.5 secant loops 10 times
and the root is 0.8410686611901758
For x0 = 1 and x1 = 1.2 secant loops 27 times
and the root is 1.0471738791351972
For x0 = 2.5 and x1 = 3 secant loops 6 times
and the root is 2.300523983022272
Άρα η μέθοδος Muller είναι γρηγορότερη.

3 Άσκηση 3

Παρακάτω θα εξηγήσουμε την λειτουργία των συναρτήσεων του κάθε υποερωτήματος αντίστοιχα.
1. gauss()
2. cholesky()
3. gaussSeidel()

3.1 PA = LU

```
def gauss(A, b):
    n = len(A)

    L = [[0.0] * n for i in range(n)]
    U = [[0.0] * n for i in range(n)]

    P = [[0.0] * n for i in range(n)]
    x = np.empty(n)

    for i in range(n):
        P[i][i] = 1.0
```

Η συνάρτηση καλείται με ορίσματα τους πίνακες A[nxn] και b[n]. Παίρνουμε το μέγεθος του πίνακα και αρχικοποιούμε τους πίνακες L[nxn], U[nxn], P[nxn], x[n]. Οι πίνακες L, P αρχικοποιούνται ως μοναδιαίοι ενώ οι άλλοι ως μηδενικοί. Στόχος είναι να λύσουμε την γραμμική σχέση:

$$Ax = b$$

```
15      #Pivot
16      for j in range(n):
17          row = max(range(j, n), key=lambda i: abs(A[i][j]))
18          if j != row:
19              P[j], P[row] = P[row], P[j]
20
21      PA = [[0] * n for i in range(n)]
22      for j in range(n):
23          for i in range(n):
24              PA[i][j] = sum(P[i][k] * A[k][j] for k in range(n))
25
26      Pb = b.copy()
27      for i in range(n):
28          Pb[i] = sum(P[i][k] * b[k] for k in range(n))
```

Αρχικά πραγματοποιούμε την οδήγηση. Δηλαδή ελέγχουμε κάθε στήλη του πίνακα A ώστε να βρούμε το μεγαλύτερο κατάπόλυτη τιμή στοιχείο της στήλης j και αντιμεταθέτουμε την γραμμή j με τη γραμμή που περιέχει το συγκεκριμένο στοιχείο. Αφού γίνει αυτό πολλαπλασιάζουμε τον πίνακα A και b με τον πίνακα μετάβασης P.

$$PAx = Pb$$

```
30      for j in range(n):
31          L[j][j] = 1
32          for i in range(j+1):
33              s1 = sum(L[i][k] * U[k][j] for k in range(i))
34              U[i][j] = PA[i][j] - s1
35
36          for i in range(j, n):
37              s2 = sum(L[i][k] * U[k][j] for k in range(j))
38              L[i][j] = (PA[i][j] - s2) / U[j][j]
```

Στη συνέχεια, υπολογίζουμε τους άνω και κάτω τριγωνικους πίνακες U, L με τους εξής τύπους:

$$U_{ij} = PA_{ij} - \sum_{k=0}^{i-1} L_{ik}U_{kj}$$

$$L_{ij} = \frac{(PA_{ij} - \sum_{k=0}^{i-1} L_{ik}U_{kj})}{U_{jj}}$$

Και φτιάχνουμε την σχέση

$$LU = PA => LUx = Pb$$

```
40      y = np.empty(n)
41      for i in range(n):
42          if L[i][i] == 0:
43              x[i] = 0
44          else:
45              value = Pb[i]
46
47              for j in range(i):
48                  value -= L[i][j] * y[j]
49
50              value /= L[i][i]
51              y[i] = value
```

Λύνουμε πρώτα την σχέση

$$Ly = Pb$$

```
53      x = np.empty(n)
54      for i in range(n - 1, -1, -1):
55          if U[i][i] == 0:
56              x[i] = 0
57          else:
58              value = y[i]
59
60              for j in range(i + 1, n, 1):
61                  value -= U[i][j] * x[j]
62
63              value /= U[i][i]
64              x[i] = value
65      return x
```

Ύστερα την σχέση

$$Ux = y$$

Και η συνάρτηση μας επιστρέφει την λύση x

3.2 Cholesky

```
78  def cholesky(A):
79      n = len(A)
80
81      L = [[0.0] * n for i in range(n)]
82
83      for i in range(n):
84          s = sum(pow(L[i][k], 2) for k in range(i))
85          L[i][i] = np.sqrt(A[i][i]- s)
86
87          for j in range(i+1, n):
88              s = sum(L[i][k]*L[j][k] for k in range(j))
89              L[j][i] = (A[j][i] - s)/L[i][i]
90      return L
```

Η συνάρτηση καλείται με ορίσμα έναν συμμετρικό και θετικά ορισμένο πίνακα A[nxn]. Παίρνουμε το μέγεθος(n) του πίνακα και αρχικοποιούμε τον πίνακα L[nxn] ως μηδενικό. Στην συνέχεια, εφαρμόζουμε τους παρακάτω τύπους του αλγορίθμου Cholesky.

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=0}^{i-1} L_{ik}^2}$$

$$L_{ji} = \frac{1}{L_{ii}}(A_{ij} - \sum_{k=0}^{j-1} L_{ik}L_{jk}), \quad for\ j > i$$

Στο τέλος η συνάρτηση επιστρέφει τον πίνακα L.

3.3 Gauss-Seidel

```
92  def gaussSeidel(A, b):
93      n = len(A)
94      x = np.empty(n)
95      oldx = np.empty(n)
96      norm = 1
97
98      while norm > 0.5*pow(10,-4):
99          oldx = x.copy()
100         for i in range(n):
101             s1 = sum(A[i][j]* x[j] for j in range(i))
102             s2 = sum(A[i][j]*x[j] for j in range(i+1,n))
103
104             x[i] = (b[i]-s1-s2)/A[i][i]
105
106         norm = max(abs((x[i] - oldx[i])) for i in range(n))
107
108     return x
```

Η συνάρτηση καλείται με ορίσματα τους πίνακες
A[nxn] και b[n]. Παίρνουμε το μέγεθος(n) του πίνακα, ορίζουμε την νόρμα (norm) με 1 και αρχικοποιούμε τους πίνακες x[n], oldx[n] ως μηδενικούς. Στην συνέχεια, όσο ισχύει

$$norm = ||x - oldx|| > 0,5 * 10^{-4}$$

αποθηκεύουμε το x στο oldx και εφαρμόζουμε τον παρακάτω τύπο του αλγορίθμου Gauss-Seidel.

$$x_i^{k+1} = \frac{b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n-1} a_{ij} x_j^k}{a_{ii}}$$

Τελικά, η συνάρτηση επιστρέφει τον πίνακα x που είναι η προσεγγιστική λύση.
Για τον πίνακα A με το μάτι καταλαβαίνουμε ότι η λύση είναι
x = [1, 1, ..., 1]
Άρα θα εξετάσουμε αν κάθε στοιχείο της x απέχει στο 1 λιγότερο από το σφάλμα.

```
148  for i in range(n-1):
149      C[i][i+1] = -2
150      C[i+1][i] = -2
151  b = [1.0 for i in range(n)]
152  b[0] = 3
153  b[n-1] = 3
154
155  x = gaussSeidel(C ,b)
156  print("The result for n = 10000 is " + str(isCorrect))
```

The result for n = 10 is True
The result for n = 10000 is True

4 Άσκηση 4

4.1 Απόδειξη ότι ο G είναι στοχαστικός

Στοχαστικός πίνακας ονομάζεται ένας τετραγωνικός πίνακας με μη αρνητικά στοιχεία τα οποία αναπαριστούν πιθανότητες. Οι γραμμές (ή οι στήλες) του έχουν άθροισμα 1. Άρα πρέπει να αποδείξουμε ότι το άθροισμα των γραμμών (ή των στηλών του) έχει άθροισμα 1.

$$\sum_{j=0}^{n-1} Gij = \sum_{j=0}^{n-1} (\frac{q}{n} + \frac{A_{ij}(1-q)}{n_j}) = \frac{nq}{n} + \frac{1-q}{n_j} \sum_{j=0}^{n-1} A_{ij} =$$

$$q + \frac{(1-q)n_j}{n_j} = q + 1 - q = 1$$

Άρα ο πίνακας είναι στοχαστικός γιατί οι στήλες του έχουν άθροισμα 1.

4.2 p = Gp

G =

$$\begin{pmatrix}
0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.435 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.2933 & 0.01 & 0.435 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 \\
0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 \\
0.435 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.01 & 0.01 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.435 & 0.435 & 0.01 & 0.2933 & 0.01 & 0.01 & 0.01 & 0.01 & 0.2225 & 0.01 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.435 & 0.435 & 0.01 & 0.01 & 0.01 & 0.2933 & 0.01 & 0.2225 & 0.01 \\
0.01 & 0.01 & 0.01 & 0.435 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.435 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.86 & 0.01 & 0.01 & 0.01 & 0.2225 & 0.01 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.435 & 0.01 & 0.435 \\
0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.86 & 0.01 & 0.01 & 0.2225 & 0.01
\end{pmatrix}$$

Το πρόγραμμα βρίσκει με την μέθοδο των δυνάμεων για ακρίβεια 10e-5 το ιδιοδυάνισμα

p = [0.02682389, 0.02986128, 0.02986128, 0.02682389, 0.03958766, ...
0.03958766, 0.03958766, 0.03958766, 0.07456589, 0.10631849, ...
0.10631849, 0.07456589, 0.12508977, 0.11633074, 0.12508977]
το οποίο είναι περίπου ίδιο με το ιδιοδυάνισμα που μας δίνεται

4.3 Βελτίωση βαθμού σημαντικότητας

Διαλέγω να βελτιώσω τον βαθμό σημαντικότητας της πρώτης σελίδας. Οπότε προσθέτω συνδέσεις από την δεύτερη, τρίτη, τέταρτη και έκτη σελίδα (Η πέμπτη ήδη πάει στην πρώτη). Αφαιρώ τυχαία την σύνδεση της τρίτης στην δεύτερη. Το αποτέλεσμα είναι:
π = [0.06184402 0.0362909 0.02482734 0.02509765 0.04261679 ...
0.04193646 0.03621257 0.03553223 0.08790694 0.10360136 ...
0.09418914 0.06532107 0.12139946 0.10982291 0.11340116]
Βλέπουμε ότι ο βαθμός σημαντικότητας της πρώτης σελίδας έχει σχεδόν τριπλασιαστεί

4.4 Αλλαγή πιθανότητας μεταπήδησης

4.4.1 q = 0.02

Result for q = 0.02 in 4.4(A) is:
p = [0.04614265 0.02394582 0.01210989 0.01502407 0.03760068
0.0356886 0.02985731 0.02794523 0.09306153 0.10916014
0.09668641 0.06936467 0.1410327 0.1335713 0.12880899]

4.4.2 q = 0.6

Result for q = 0.6 in 4.4(B) is:
p = [0.07792531 0.05558503 0.05236878 0.05107864 0.05514962
0.05657374 0.05396778 0.05539189 0.07193381 0.08621975
0.0850861 0.06306842 0.08174926 0.07260688 0.08129499]
Ο σκοπός της πιθανότητας μεταπήδησης είναι να χαρακτηρίζει πόσο σημαντικό είναι μια σελίδα να δέχεται συνδέσμους από άλλες σελίδες. Όσο μικρότερο τόσο σημαντικότερο και το αντίστροφο. Βλέπουμε επίσης ότι για μεγάλο q όλες οι σελίδες έχουν μικρότερες διαφορές στις τάξεις τους

4.5 Βελτίωση τάξης της σελίδας 11

Result for 4.5(before) is:
Rank of 11 = 0.10631848768394055
Rank of 10 = 0.10631848768394055
Result for 4.5(after) is:
Rank of 11 = 0.12400835065568959
Rank of 10 = 0.10289405307708084
Όντως αυτή η στρατηγική δουλεύει

4.6 Διαγραφή της σελίδας 10

Difference after deletion of site 10 for 4.6 is
diff = [0.0446, 0.0314, 0.0314, 0.0142, 0.0217, 0.0217, 0.0116, 0.0116, -0.0031, 0, -0.0031, -0.0941, -0.0449, -0.0334]
Παρατηρούμε ότι οι τάξεις σελιδών που αυξάνονται είναι αυτές με μεγάλο βαθμό σημαντικότητας ενώ μειώνονται οι σελίδες με μικρό βαθμό