

Project 2020-2021

Matlab Implementation

Σιγούρου Άλκηστις-Αικατερίνη AM 1059661

24 Φεβρουαρίου 2021



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Εργασία στα πλαίσια του μαθήματος
Επιστημονικός Υπολογισμός

Τμήμα Μηχ. Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Περιεχόμενα

1	Εισαγωγικά	1
1.1	Στοιχεία Υπολογιστικού Συστήματος	1
1.1.1	1
1.1.2	1
1.1.3	1
2	Αραιή αναπαράσταση BCRS	3
2.1	3
2.2	4
2.3	5
3	Τανυστές και διαδρομές	6
3.1	6
3.2	8
3.3	9
4	Επαναληπτικές μέθοδοι	10
4.1	Ειδικά μητρώα	10

1 Εισαγωγικά

1.1 Στοιχεία Υπολογιστικού Συστήματος

1.1.1

Χαρακτηριστικά	Απαντήσεις
Έναρξη/λήξη εργασίας	18/02/2021 - 23/02/2021
model	DELL Laptop
O/S	Windows 10
processor name	Intel Core i7 6500U
processor speed	2.50GHz (base)
number of processors	1
total # cores	2
total # threads	4
FMA instruction	yes
L1 cache	32KB, 32KB
L2 cache	(per core) 256KB
L3 cache	(shared) 4MB
Gflops/s	107.1
Memory	8GB
Memory Bandwidth	
MATLAB Version	9.8.0.1323502 (R2020a)
BLAS	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch AVX2
LAPACK	Intel(R) Math Kernel Library Version 2019.0.3 Product Build 20190125 for Intel(R) 64 architecture applications, CNR branch AVX2 Linear Algebra PACKage Version 3.7.0

Table 1: Πίνακας Χαρακτηριστικών

Οι πληροφορίες αντλήθηκαν από την εφαρμογή CPUz , καθώς και από τις Ιδιότητες του System της συσκευής μας. Επίσης χρησιμοποιήθηκαν οι σύνδεσμοι που υπήρχαν στην εκφώνηση του Προθεστού από τους διδάσκοντες.

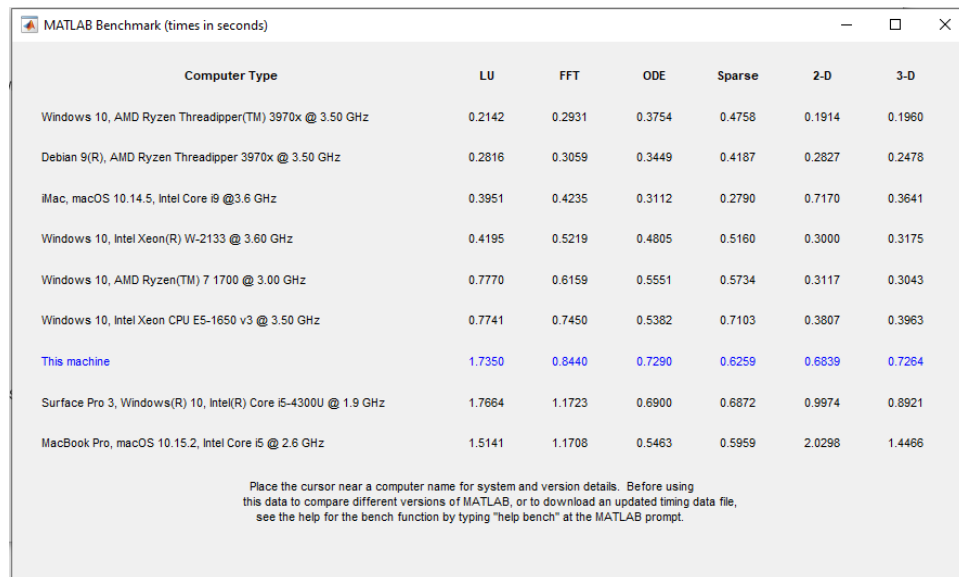
1.1.2

Η έκδοση του MATLAB , που χρησιμοποιήθηκε για την υλοποίηση των ασκήσεων, είναι η 9.8.0.1323502 (R2020a)

Δεν χρησιμοποιήθηκαν βιβλιοθήκες για τα παρακάτω ερωτήματα.

1.1.3

Μετά την εκτέλεση της εντολής bench προέκυψαν τα στοιχεία στο Σχήμα 1 .



MATLAB Benchmark (times in seconds)

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 10, AMD Ryzen Threadripper(TM) 3970x @ 3.50 GHz	0.2142	0.2931	0.3754	0.4758	0.1914	0.1960
Debian 9(R), AMD Ryzen Threadripper 3970x @ 3.50 GHz	0.2816	0.3059	0.3449	0.4187	0.2827	0.2478
iMac, macOS 10.14.5, Intel Core i9 @3.6 GHz	0.3951	0.4235	0.3112	0.2790	0.7170	0.3641
Windows 10, Intel Xeon(R) W-2133 @ 3.60 GHz	0.4195	0.5219	0.4805	0.5160	0.3000	0.3175
Windows 10, AMD Ryzen(TM) 7 1700 @ 3.00 GHz	0.7770	0.6159	0.5551	0.5734	0.3117	0.3043
Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz	0.7741	0.7450	0.5382	0.7103	0.3807	0.3963
This machine	1.7350	0.8440	0.7290	0.6259	0.6839	0.7264
Surface Pro 3, Windows(R) 10, Intel(R) Core i5-4300U @ 1.9 GHz	1.7664	1.1723	0.6900	0.6872	0.9974	0.8921
MacBook Pro, macOS 10.15.2, Intel Core i5 @ 2.6 GHz	1.5141	1.1708	0.5463	0.5959	2.0298	1.4466

Place the cursor near a computer name for system and version details. Before using this data to compare different versions of MATLAB, or to download an updated timing data file, see the help for the bench function by typing "help bench" at the MATLAB prompt.

Σχήμα 1: Στιγμιότυπο από την εκτέλεση της συνάρτησης bench

2 Αραιή αναπαράσταση BCRS

2.1

Στο 1^ο ερώτημα καλούμαστε να κατασκευάσουμε την συνάρτηση `sp_mx2bcrs`, η οποία εκτελεί την μέθοδο CSR (Compressed Sparse Row) σε μητρώα, κοιτάζοντας τα ανά μπλοκ.

Δηλαδή έστω το μητρώο στο Σχήμα 2 το σπάμε ανά μπλοκ των 2×2 όπως έχουμε σχηματίσει. Μετά σαρώνοντας όλο το μητρώο κατά γραμμές και κρατάμε τα μπλοκ που περιέχουν έστω και ένα nnz στοιχείο. Έπειτα με την σειρά που τα ανακαλύπτουμε τα τοποθετούμε στο `val` και παράλληλα στο `col_idx` βάζουμε τον αριθμό της στήλης στην οποία βρίσκεται. Οι στήλες πλέον καθορίζονται από την αρχική διάσταση διαιρεμένη με το `nb`, οπότε μπαίνει και η ανάλογη τιμή στο `col_idx`. Στην μεταβλητή `row_blk` βάζουμε την αρίθμηση του πρώτου nnz block που βρίσκουμε σαρώνοντας το μητρώο μας κατά στήλη. Στη Λίστα 2.1 βλέπουμε τον κώδικα για το πρώτο ερώτημα.

A =

		1			2			3	4
1	5	3	0	0	0	-2	3	0	
	1	5	0	0	0	0	3	0	
2	0	0	0	0	0	0	2	0	
	0	0	0	0	8	0	0	0	
3	0	0	0	0	0	0	5	-1	
	3	-4	0	0	0	0	5	0	
4	0	0	0	3	6	0	7	0	
	0	0	3	0	0	0	9	0	

Σχήμα 2: A 8x8 Sparse Matrix divided in blocks

Listing 1: Ερώτημα 2.1

```

1 function [val,col_idx,row_blk] = sp_mx2bcrs(A,nb)
2 % Author: ALKISTIS-AIKATERINI SIGOIROU, AM 1059661, Date: 22/02/2021
3
4 [n,n]=size(A);
5     if size(A)~= [n,n], error('wrong dimensions!'), end
6
7     val=[];
8
9
10    col_idx=[];
11
12    row_blk=[];
13
14    col_idx_checker = (1:n/nb);
15    count=0;
16
17    for i=1:nb:n
18        k=0;
19        flag=1;
20        for j=1:nb:n
21            k=k+1;
22            block=A(i:i+nb-1,j:j+nb-1);
23            if sum(nnz(block))~=0
24                val=[val block];
25
26                col_idx=[col_idx col_idx_checker(k)];
27

```

```

28         count=count+1;
29         if flag==1
30             row_blk=[row_blk count];
31         end
32         flag = 0;
33     end
34 end
35 end
36 row_blk=[row_blk count+1];
37
38
39
40
41 end

```

2.2

Στο 2^ο ερώτημα μας ζητήθηκε να πολλαπλασιάσουμε το μητρώο μας στο ερώτημα 1 με ένα διάνυσμα. Προκειμένου να γλιτώσουμε πράξεις προγραμματίζουμε την συνάρτησή μας να προσπερνάει τα μηδενικά μπλοκ και να πολλαπλασιάζει μόνο τα nnz στοιχεία του μητρώου.

Έτσι πχ, αν είχαμε το μητρώο και το διάνυσμα στην Λίστα 2.2 οι πολλαπλασιασμοί που θα λάβουν χώρα είναι οι εξής :

```

A =
    5     3     0     0     0    -2     3     0
    1     5     0     0     0     0     3     0
    0     0     0     0     0     0     2     0
    0     0     0     0     8     0     0     0
    0     0     0     0     0     0     5    -1
    3    -4     0     0     0     0     5     0
    0     0     0     3     6     0     7     0
    0     0     3     0     0     0     9     0

>> V=(1:8)'

V =

     1
     2
     3
     4
     5
     6
     7
     8

```

Σχήμα 3: A 8x8 Sparse Matrix & a Vector

Listing 2: Ερώτημα 2.2

```

1 function y = spmv_bcsr(y ,val , row_blk,col_idx , x)
2 % Author: ALKISTIS-AIKATERINI SIGOYROU, AM 1059661, Date: 22/02/2021
3 [m,n]=size(val);
4 nb = m;
5
6 n=length(row_blk)-1;
7 for i=1:n
8     k1=row_blk(i)*nb-1;
9     k2=row_blk(i+1)*nb-2;

```

```

10
11         xtemp=[];
12
13         checker=(k2-k1+1)/nb;
14         for j=1:checker
15             xtemp=[xtemp; x(col_idx(j)*nb-1:col_idx(j)*nb-1+nb-1)];
16         end
17         col_idx=col_idx(checker+1:end);
18
19         y(i*nb-nb+1:i*nb) = y(i*nb-nb+1:i*nb)+val(:,k1:k2)*xtemp;
20     end
21
22 end

```

2.3

Στο 3^ο ερώτημα κληθήκαμε να επιλέξουμε 2 μητρώα από την συλλογή του SuiteSparse. Τα μητρώα που επιλέχθηκαν ήταν τα bcspr06 και Ishp1882, τα οποία μπορείτε να τα βρείτε στον φάκελο με τα συμπιεσμένα αρχεία. Τρέχουμε τις 2 συναρτήσεις μας από τα προηγούμενα ερωτήματα χρησιμοποιώντας την εξής ακολουθία εντολών :

- Import data - (όνομα αρχείου)
- B=Problem.A;
- B=full(B); - μετατρέπουμε το αραιό μητρώο σε πλήρη μορφή
- [val,col_idx,row_blk] = sp_mx2bcrs(B,2);
- y = spmv_bcsr(zeros(n,1),eval, row_blk,col_idx, (1:n)'); - όπου n το μέγεθος του μητρώου του Problem

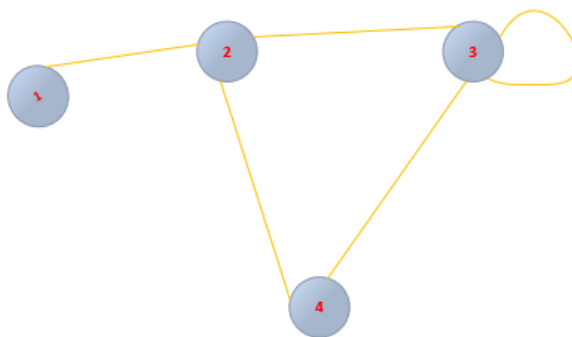
Ύστερα, προκειμένου να επιβεβαιώσουμε την ορθότητα των πράξεων μας χρησιμοποιούμε τις εντολές :

- p=B*(1:n)'; - κάνουμε τον πολλαπλασιασμό ολόκληρου του μητρώου με το διάνυσμα που σχηματίσαμε
- p == y - τεστάρουμε αν οι 2 απαντήσεις είναι λογικά ίδιες, και σας αποτέλεσμα παίρνουμε διάνυσμα με άσσους μήκους n
- sum(ans) - προσθέτουμε όλα τα στοιχεία του διανύσματος και αν το τελικό μας άθροισμα είναι όσο η διάσταση του μητρώου, τότε αντιλαμβανόμαστε ότι ο κώδικας μας είναι σωστός.

3 Τανυστές και διαδρομές

Τα μητρώα γειτνίασης μας παρουσιάζουν το πλήθος των βημάτων που χρειάζεται ο ένας κόμβος για να φτάσει στους άλλους. Το πλήθος των μητρώων γειτνίασης είναι ίσο με το πλήθος των κόμβων μείον ένα. Στις πρώτες τάξεις έχουμε τις διαδρομές που δημιουργούνται με βήμα 1, στις δεύτερες τάξεις με βήμα 2, στις 3ης με βήμα 3 κ.ο.κ.

Για λόγους αναπαράστασης και ευκολίας κατανόησης του κώδικα, έχουμε ορίσει ένα τυχαίο γράφημα 4άρων κόμβων (Σχήμα 4) και στο Σχήμα 5 έχουμε τον πίνακα γειτνίασης τάξης 1ης.



Σχήμα 4: Γράφημα 4άρων κόμβων

```

Command Window

>> A=[0,1,0,1;1,0,1,1;0,1,2,1;0,1,1,0]

A =

     0     1     0     1
     1     0     1     1
     0     1     2     1
     0     1     1     0
  
```

Σχήμα 5: Μητρώο γειτνίασης A*1

3.1

Στο 1ο ερώτημα, μας ζητάτε να εμφανίσουμε τους πίνακες γειτνίασης G των τάξεων. Ο κώδικας παρουσιάζεται στην Λίστα 3.1. Ενώ στα Σχήματα 6,7 παρουσιάζεται η σειρά εντολών που εκτελέσαμε, προκειμένου να πάρουμε αποτελέσματα και να τα επαληθεύσουμε αντίστοιχα.

Listing 3: Ερώτημα 3.1

```

1 function G = tensor_geit(A)
2 % Author: ALKISTIS-AIKATERINI SIGOIROU, AM 1059661, Date: 18/02/2021
3
4     [n,n]=size(A);
5     maxL = n-1;
  
```



```

6
7     G(:, :, 1) = A;
8
9     for i = 2:n-1
10
11         G(:, :, i) = A^(i);
12     end
13
14 end

```

```

>> G = tensor_geit(A)

G(:, :, 1) =

     0     1     0     1
     1     0     1     1
     0     1     2     1
     0     1     1     0

G(:, :, 2) =

     1     1     2     1
     0     3     3     2
     1     3     6     3
     1     1     3     2

G(:, :, 3) =

     1     4     6     4
     3     5    11     6
     3    10    18    10
     1     6     9     5

```

Σχήμα 6: Εκτέλεση 3.1

```

Command Window

>> p=A*A;
>> k=A*A*A;
>> p==G(:, :, 2)

ans =

4x4 logical array

    1    1    1    1
    1    1    1    1
    1    1    1    1
    1    1    1    1

>> k==G(:, :, 3)

ans =

4x4 logical array

    1    1    1    1
    1    1    1    1
    1    1    1    1
    1    1    1    1

```

Σχήμα 7: Επαλήθευση 3.1

3.2

Στο 2ο ερώτημα, μας ζητάτε να βρούμε τις διαδρομές από i, j μήκους k . Ο κώδικας παρουσιάζεται στην Λίστα 3.2. Ενώ στα Σχήματα 8,9 παρουσιάζεται η σειρά εντολών που εκτελέσαμε, προκειμένου να πάρουμε αποτελέσματα και να τα επαληθεύσουμε αντίστοιχα.

Listing 4: Ερώτημα 3.2

```

1 function B = tensor_row(G)
2 % Author: ALKISTIS-AIKATERINI SIGOUREOU, AM 1059661, Date: 18/02/2021
3
4     [n,n,m]=size(G);
5
6     B=zeros(n)
7     for j=1:m
8
9         B=B+G(:, :, j);
10    end
11 end

```

```

Command Window

>> B = tensor_row(G)

B =

    0    0    0    0
    0    0    0    0
    0    0    0    0
    0    0    0    0

B =

    2     6     8     6
    4     8    15     9
    4    14    26    14
    2     8    13     7

```

Σχήμα 8: Εκτέλεση 3.2

```

>> P=G(:, :, 1)+G(:, :, 2)+G(:, :, 3);
>> P==B

ans =

4x4 logical array

    1    1    1    1
    1    1    1    1
    1    1    1    1
    1    1    1    1

```

Σχήμα 9: Επαλήθευση 3.2

3.3

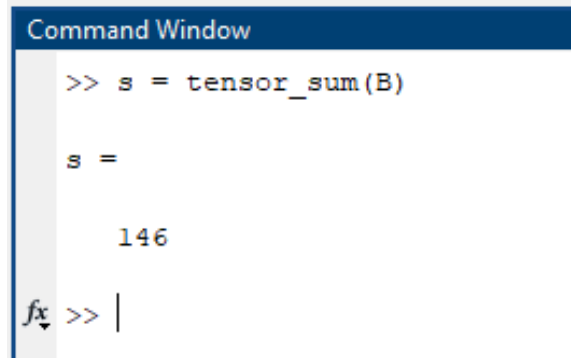
Στο 3ο ερώτημα, ψάχνουμε τις διαδρομές μήκους k μεταξύ 2 κόμβων. Ο κώδικας παρουσιάζεται στην Λίστα 3.3. Ενώ στα Σχήματα 10,11 παρουσιάζεται η σειρά εντολών που εκτελέσαμε, προκειμένου να πάρουμε αποτελέσματα και να τα επαληθεύσουμε αντίστοιχα.

Listing 5: Ερώτημα 3.3

```

1 function s = tensor_sum(B)
2 % Author: ALKISTIS-AIKATERINI SIGOYRΟΥ, AM 1059661, Date: 18/02/2021
3     [n,n]=size(B);
4
5     s=sum(sum(B));
6
7 end

```



```

Command Window

>> s = tensor_sum(B)

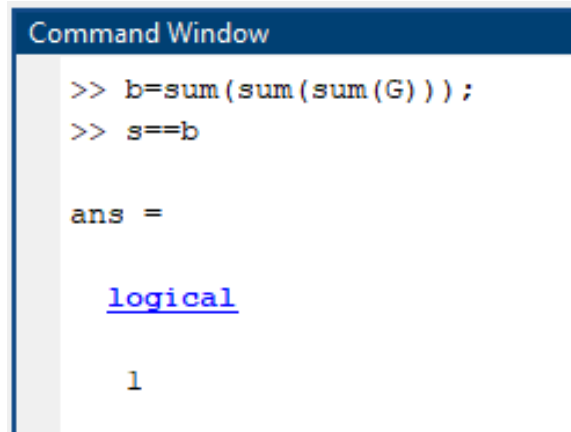
s =

    146

fx >> |

```

Σχήμα 10: Εκτέλεση 3.3



```

Command Window

>> b=sum(sum(sum(G)));
>> s==b

ans =

    logical

     1

```

Σχήμα 11: Επαλήθευση 3.3

4 Επαναληπτικές μέθοδοι

4.1 Ειδικά μητρώα

Στο Σχήμα 13 παρατηρούμε τα αποτελέσματα του κώδικα στην Λίστα 4.1. Το σημείο σύγκλισης της πρώτης δοθέν συνάρτησης είναι το 90 και της 2ης το 11. Παρόλα αυτά όπως φαίνεται και στο Σχήμα 13 ο δείκτης κατάστασης της 2ης συνάρτησης είναι μεγαλύτερος, επομένως σαν συνέχεια έχουμε το ότι η αλγοριθμική υλοποίηση της 2ης συνάρτησης είναι πολύ καλύτερη από αυτής της 1ης.

Listing 6: Ερώτημα 5.1

```

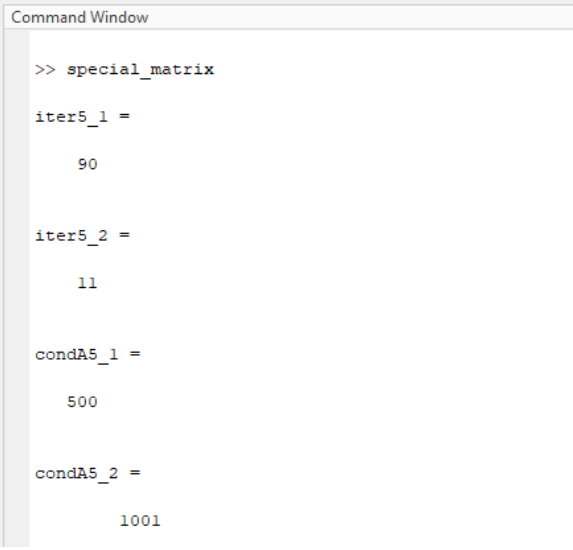
1 % Author: ALKISTIS-AIKATERINI SIGOUREU, AM 1059661, Date: 23/02/2021
2 n=500;
3 A5_1=spdiags((1:n)',0,n,n);
4 A5_2=spdiags([linspace(1,2,n/2)';linspace(1000,1001,n/2)'],0,n,n);
5
6 xsol=ones(n,1);
7
8 b5_1=A5_1*xsol;
9 b5_2=A5_2*xsol;
10
11 tol=1e-6;
12 maxIt=0:4*n;
13
14 for i=maxIt

```

```

15     [x5_1,flag5_1,relres5_1,iter5_1,resvec5_1]= pcg(A5_1,b5_1,tol,i);
16     res5_1(i+1)=relres5_1;
17     [x5_2,flag5_2,relres5_2,iter5_2,resvec5_2]= pcg(A5_2,b5_2,tol,i);
18     res5_2(i+1)=relres5_2;
19 end
20
21 semilogy(maxIt,res5_1,maxIt,res5_2)
22 legend('A5_1','A5_2')
23 caption = sprintf('A5_1 error =%f , A5_2 error =%f ', relres5_1 , relres5_2);
24 annotation('textbox','String',caption,'FitBoxToText','on');
25 xlabel('Iterations')
26 ylabel('Error Value')
27 title('Results 5_1')
28 xlim([0 max(iter5_1,iter5_2)+10]);
29
30 [x5_1,flag5_1,relres5_1,iter5_1,resvec5_1]= pcg(A5_1,b5_1,tol,4*n);
31 [x5_2,flag5_2,relres5_2,iter5_2,resvec5_2]= pcg(A5_2,b5_2,tol,4*n);
32 iter5_1
33 iter5_2
34
35 cond(A5_1)
36 cond(A5_2)

```



```

Command Window

>> special_matrix

iter5_1 =

    90

iter5_2 =

    11

condA5_1 =

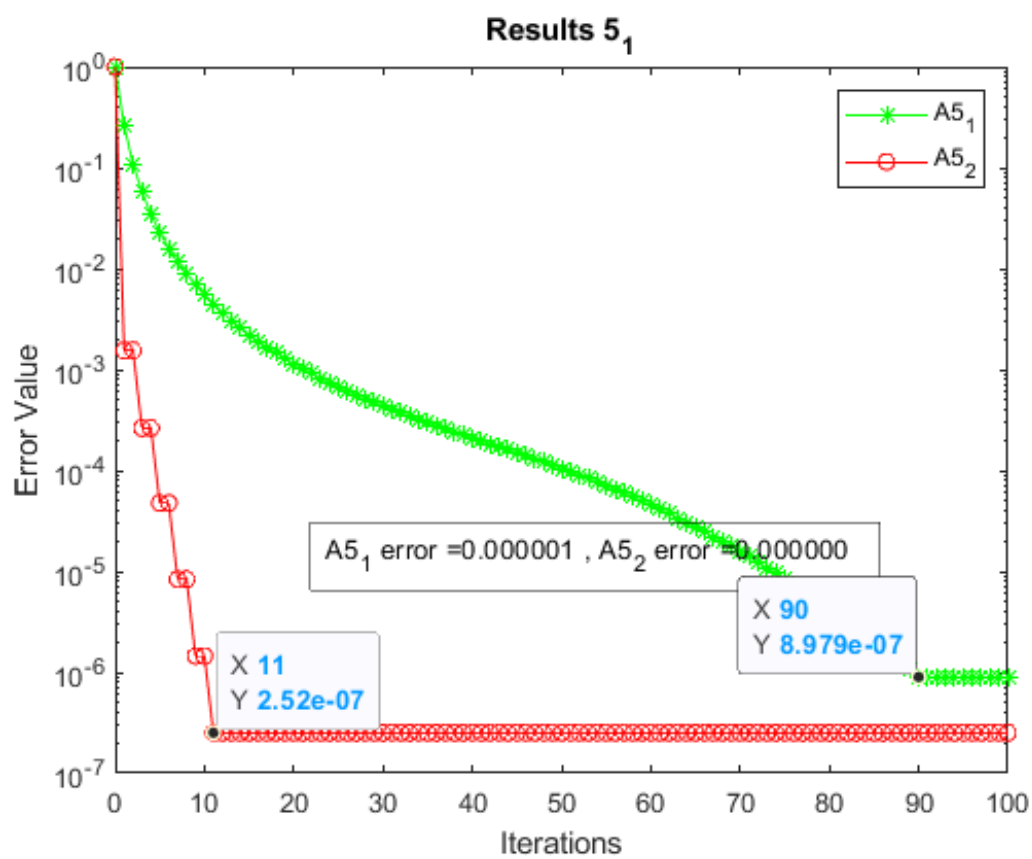
    500

condA5_2 =

   1001

```

Σχήμα 12: Αποτελέσματα της 5.1



Σχήμα 13: Γραφική αναπαράσταση των αποτελεσμάτων