Επιστημονικός Υπολογισμός Ι
Εργαστηριακή Άσκηση

Ζάκκας Κωνσταντίνος
ΑΜ: 5292

Προκαταρκτικά: Περιγραφή των χαρακτηριστικών του συστήματος που χρησιμοποιήθηκε (υλικό και λογισμικό):

1)

- **Επεξεργαστής:** Intel® Core™ i5-5200U CPU @ 2.20GHz × 4
- **Κρυφή μνήμη:** Η κρυφή μνήμη είναι 3072 KB και αποτελείται από τρία επίπεδα L1 που χωρίζεται κρυφή μνήμη εντολών με μνήμη και δεδομένων με χωρητικότητα 16kb και 16 kb αντίστοιχα, L2 με χωρητικότητα 128kb και L3 με χωρητικότητα 3Mb.
- Υποστηρίζει FMA

2) **Έκδοση MATLAB**: 2015a

1 ΜΕΡΟΣ Α

2.
(α')

```
for n=[128 512]
for t=[1 2 10]
A=rand(n);
rank=SMW(A,n,t);
disp(rank);
end
end
```

n	t	rank
128	1	1
128	2	2
128	10	10
512	1	1
512	2	2
512	10	10

(β')

```
for n=[128 512]
for t=[1 2 10]
A=tril(randn(n));
rank=SMW(A,n,t);
disp(rank);
end
end
```

n	**t**	**rank**
128	1	3
128	2	3
128	10	7
512	1	11
512	2	11
512	10	11

Οι τιμές του rank στον παραπάνω πίνακα δεν ήταν ίσες με τον χρόνο οπότε δεν επιβεβαιώνουν την σχέση ενώ δεν ήταν σταθερές όπως στον προηγούμενο πίνακα αλλά σε διαφορετικές εκτελέσεις πήραμε διαφορετικές τιμές για το rank με μια διαφορά έως συν/πλην 3.

(γ') Ένα παράδειγμα είναι το προηγούμενο μητρώο A=tril(rand(n)) όπου δεν επιβεβαιώθηκε η σχέση [1]. Ένα άλλο παράδειγμα είναι το μητρώο A=triu(rand(n)) που δίνει τιμές:

n	**t**	**rank**
128	1	1
128	2	1
128	10	1
512	1	1
512	2	1
512	10	1

Αντίστοιχα το μητρώο A=gep(rand(n)) δίνει:

n	**t**	**rank**
128	1	5
128	2	5
128	10	10
512	1	12
512	2	18
512	10	10

Για τις τιμές των παραπάνω πινάκων εκτελέστηκε ο παρακάτω κώδικας:

```
% Για το μητρωο A=gep(rand(n))
for n=[128 512]
for t=[1 2 10]
A=gep(randn(n));
rank=SMW(A,n,t);
disp(rank);
end
end
% Για το μητρωο A=triu(rand(n)))
for n=[128 512]
for t=[1 2 10]
A=triu(rand(n));
rank=SMW(A,n,t);
disp(rank);
end
end
```

(δ')

```
for n=[128 512]
for t=[1 2 10]
A=gfpp(n);
rank=SMW(A,n,t);
disp(rank);
end
end
```

n	t	rank
128	1	2
128	2	3
128	10	11
512	1	1
512	2	1
512	10	1

Για n=128 οι τιμές είναι αρκετά κοντά στις επιθυμητές ενώ για n=512 οι τιμές έχουν μεγάλη διαφορά με τις επιθυμητές.

2 ΜΕΡΟΣ Β

1. Τροποποιήσαμε τον κώδικα της επαναληπτικής εκλέπτυνσης ώστε να λύνει το σύστημα (A + W H^T)x = b όπου W, H τάξης 1 χρησιμοποιώντας αποκλειστικά αριθμητική διπλής ακρίβειας.

```
function [k,err]=itref_double(A,W,H,b)
%
A=double((A + W *H_T)^(-1));
b=double(b);
x=double(A*b);
k=1;
% Αρχικοποιηση της συνθηκης της while ωστε να
% τρεξει τουλαχιστον μια φορα
err=10e-5;
while err>10e-6 %repeat
x1=x;
r=double(b-A*x);
z=double((A^(-1))*r);
x=double(x+z);
err=norm(x-x1,inf)/norm(x1,inf);
k=k+1;
% Αν το σφαλμα δεν γινεται μικροτερο απο την επιθυμητη τιμη
% επιστρεφει το αναλογο μηνυμα
if k>realmax
    disp('Αστοχία επαρκούς σύγκλισης');
end %until
end
```

3. Για το παρακάτω μητρώο A γίνεται χρήση της παραπάνω μεθόδου ώστε να μελετηθεί αν συγκλίνει και πόσο γρήγορα σε χρόνο και επαναλήψεις εκλέπτυνσης, για όλους τους συνδυασμούς των τιμών του h και του n.

```
for n=[10 100 1000]
for h=2:4
W=randn(n,1);
W=W/norm(W);
H=W(n:-1:1);
A=toeplitz([h,-1,zeros(1,n-2)]);
A_1=(A+W*H_T);
[k,err]=itref_double(A,W,H,b);
tm=@() itref_double(A,W,H,b);
t=timeit(tm);
```

```
end
end
```

n	h	k(επαναλήψεις)	Χρόνος σύγκλισης
10	2	2	5.859092*(10^-5)
	3	2	5.686713*(10^-5)
	4	2	5.758459*(10^-5)
100	2	2	6.404890*(10^-4)
	3	2	5.774890*(10^-4)
	4	2	5.784890*(10^-4)
1000	2	2	1.449939*(10^-1)
	3	2	1.380869*(10^-1)
	4	2	1.444259*(10^-1)

Παρατηρούμε ότι όσο μεγαλώνει το n μεγαλώνει και ο χρόνος που χρειάζεται για την εκτέλεση της συνάρτησης που δημιουργήσαμε ενώ για το h, η τιμή για h=3 είναι μικρότερη από ότι για h=2 και το ίδιο συμβαίνει και για h=4 μόνο που γίνεται μεγαλύτερη από την τιμή για h=3.

3 ΜΕΡΟΣ Γ

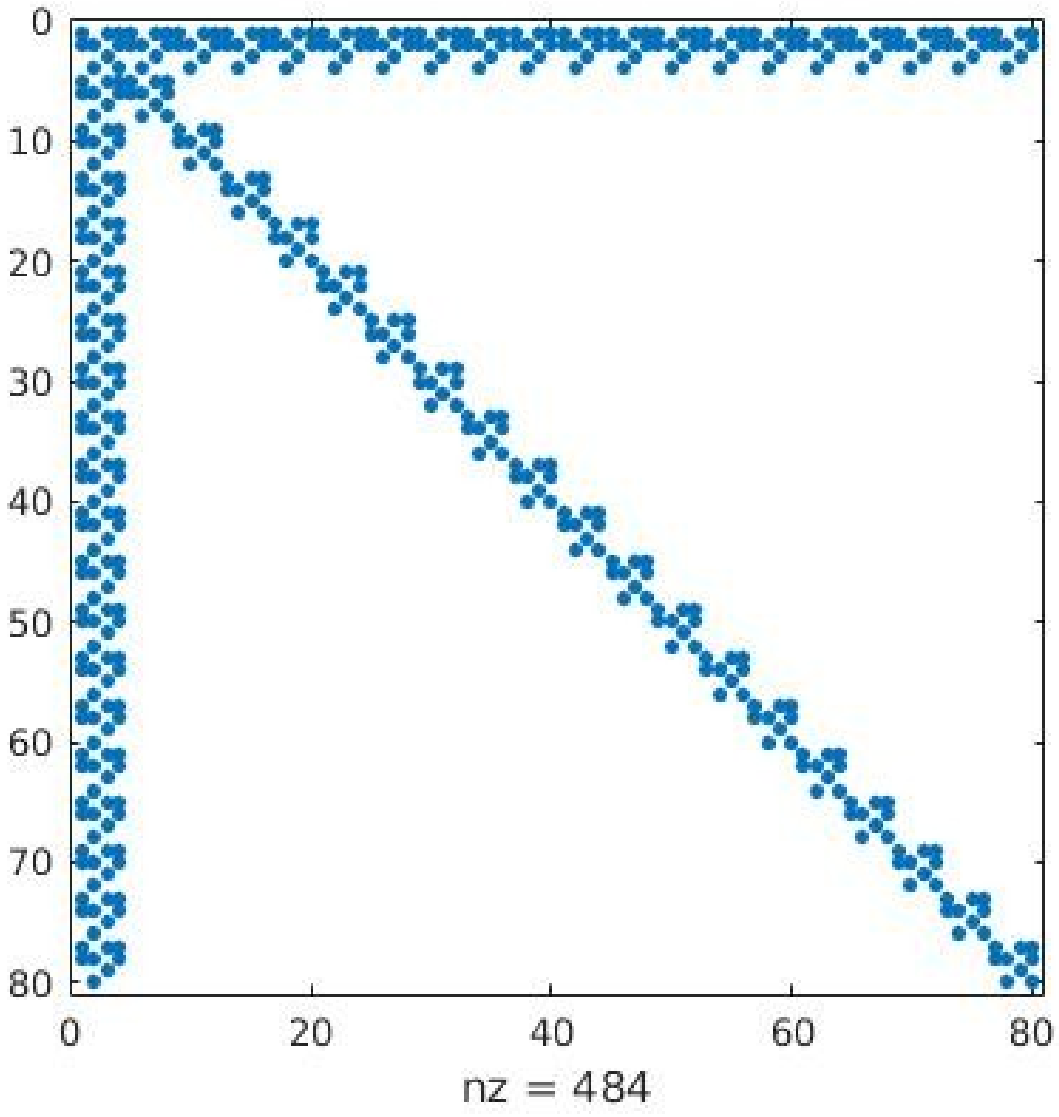
1. Παρακάτω παρατίθεται ο κώδικας που κατασκεάστηκε για να δημιουργηθεί το αρχικό ΒΔ μητρώο.

```
myHash=9397;
rng(myHash);
s=0;
for n=[125 250]
for m=[8 16]
    s=s+1;
    if s>2
        m=m/2;
    end
    T=full(sprand(m,m,0.6));
     A=arrowNW(T,n);
end
end
```

Η συνάρτηση arrowNW(T,n) θα παίρνει για είσοδο το παραπάνω T και την τιμή του n και θα κατασκευάζει μητρώα $A \in R^{mn \times mn}$ με μορφή ΒΔ βέλους βάσει του T και ενός μητρώου D = T + mI όπου I το ταυτοτικό μεγέθους m.
Ο κώδικας της συνάρτησης arrowNW:

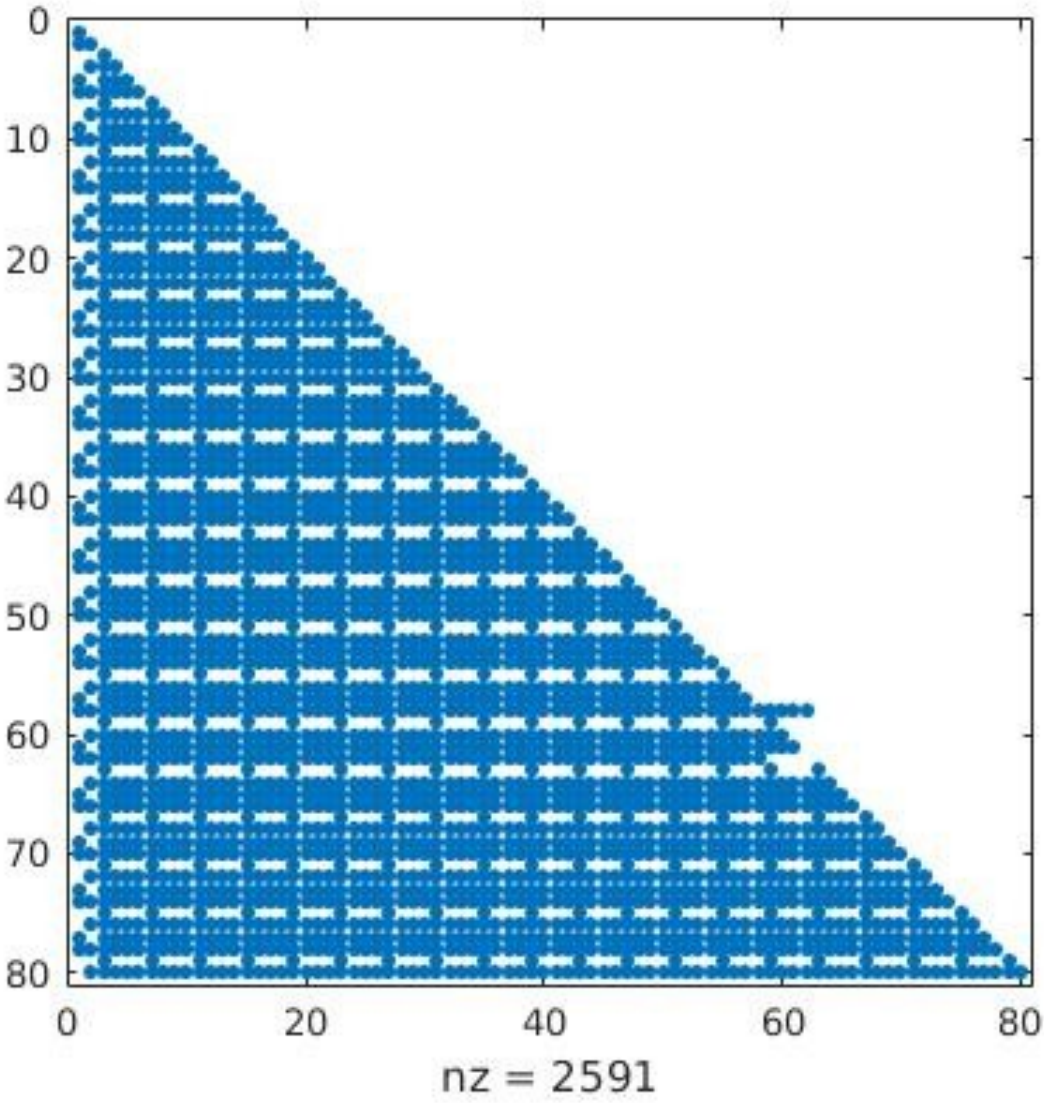
```
function A = arrowNW(T,n)
%
m=length(T);
A=eye(n);
I=0;
for u=1:m
    I(u,u)=1;
end
D=T+m*I;
A=kron(A,D);
for i=2:n
    A1(1,i)=1;
    A1(i,i)=0;
    A2(i,1)=1;
    A2(i,i)=0;
end
A1=kron(A1,T);
A2=kron(A2,T);
A=A+A1+A2;
```

2. Με χρήση της εντολής spy παρακάτω απεικονίζεται σε σχήμα το μητρώο A=arrowNW(T,n) για m=4 και n=20.

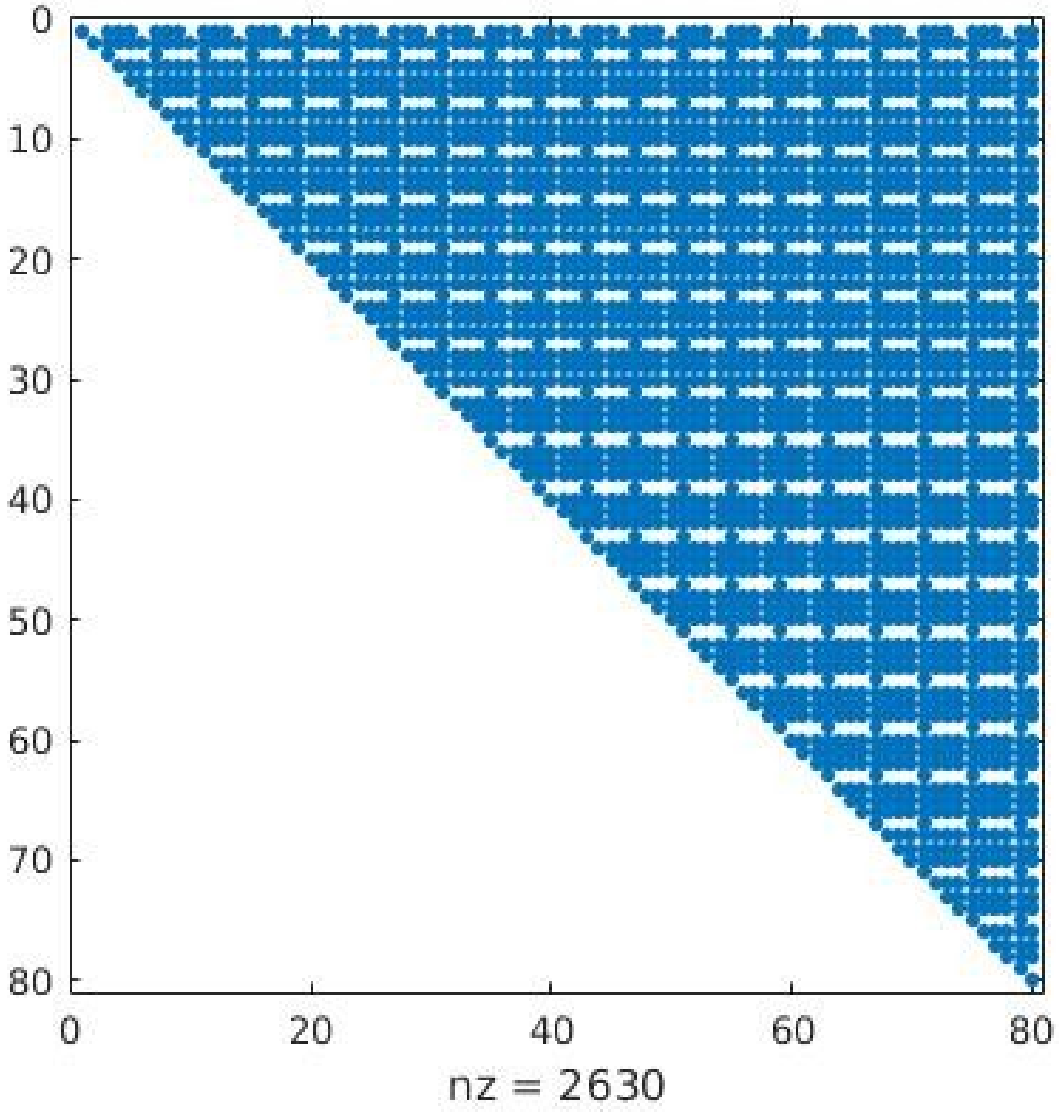


3. Χρησιμοποιώντας την συνάρτηση lu της MATLAB για το μητρώο A που προέκυψε παραπάνω παίρνουμε τα αντίστοιχα L και U. Παρακάτω είναι η απεικόνιση τους με χρήση ξανά της συνάρτησης spy.

L:



U:



4. Για να ελέγξουμε το γέμισμα που δημιουργείται συγκρίνουμε το πλήθος των μηδενικών του A με τα μη μηδενικά των L και U μείονν mn

nnz(A)=504
nnz(L)=2591
nnz(U)=2630

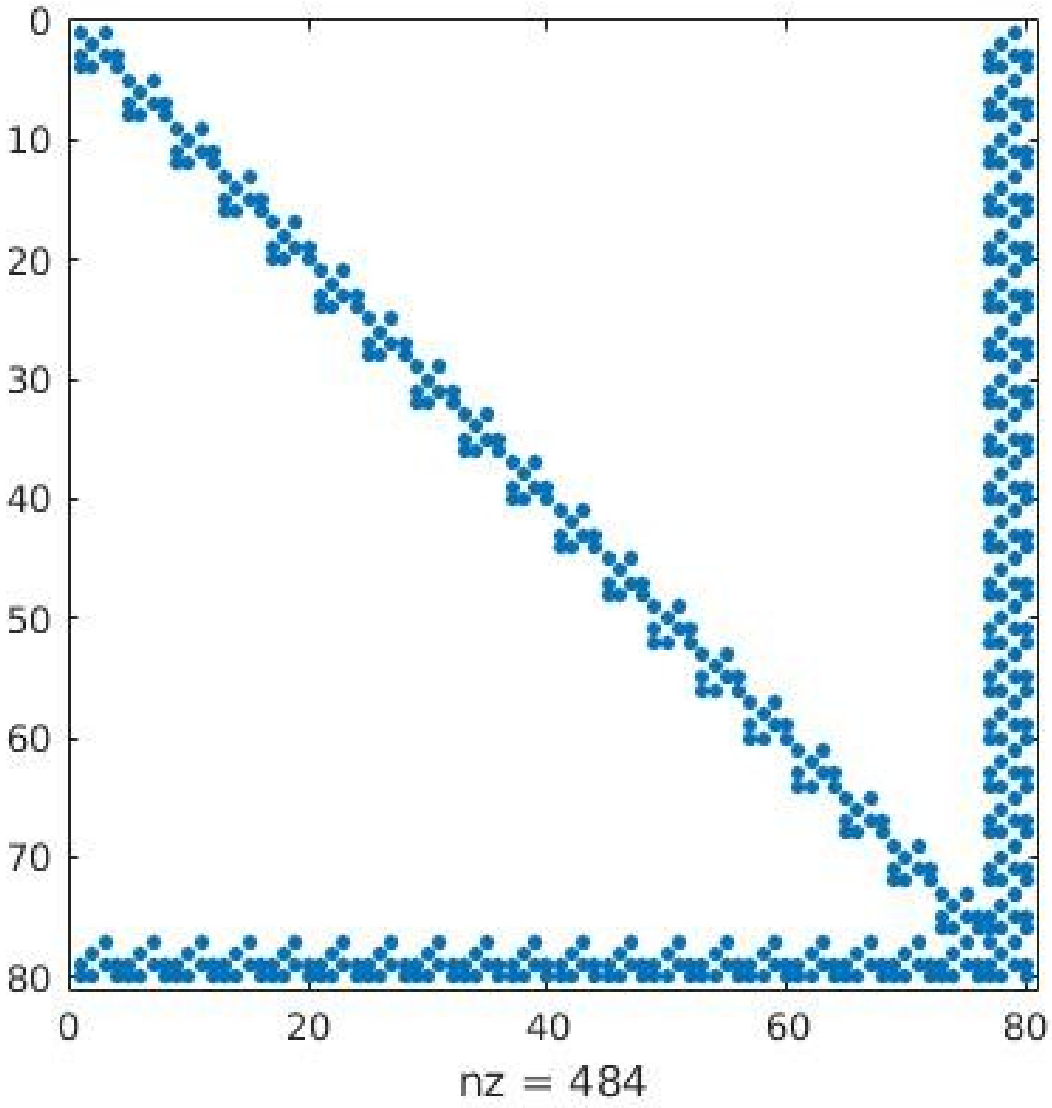
το ποσοτικοποιούμε ως:

fill_A=(nnz(L) + nnz(U) – m*n)/nnz(A)=10.5179

5. Όπως έχουμε δει στις διαλέξεις του μαθήματος η επίλυση συστήματος με μητρώο που είναι σε μορφή βέλους με ”βορειοδυτική” (ΒΔ) φορά έχει το μειονέκτημα ότι οδηγεί σε πυκνούς παράγοντες L, U που μπορεί να είναι μεγάλο μειονέκτημα. Το πρόβλημα λύνεται με επαναρίθμηση ώστε το μητρώο να έχει δομή βέλους με ”νοτιανατολική” (ΝΑ) φορά. Σε αυτο το ερώτημα δημιουργούμε ένα μητρώο μετάθεσης W τέτοιο ώστε B=W*A*W_T να έχει μικρότερο γέμισμα στους αντίστοιχους παράγοντες LU. Το W που θα δημιουργήσουμε και το ανάστροφό του W_T όταν πολλαπλασιαστούς με το μητρώο A που δημιουργήσαμε στο πρώτο ερώτημα, θα το μετατρέψει από μητρώο μορφής ΒΔ βέλους σε μητρώο μορφής ΝΑ βέλους.

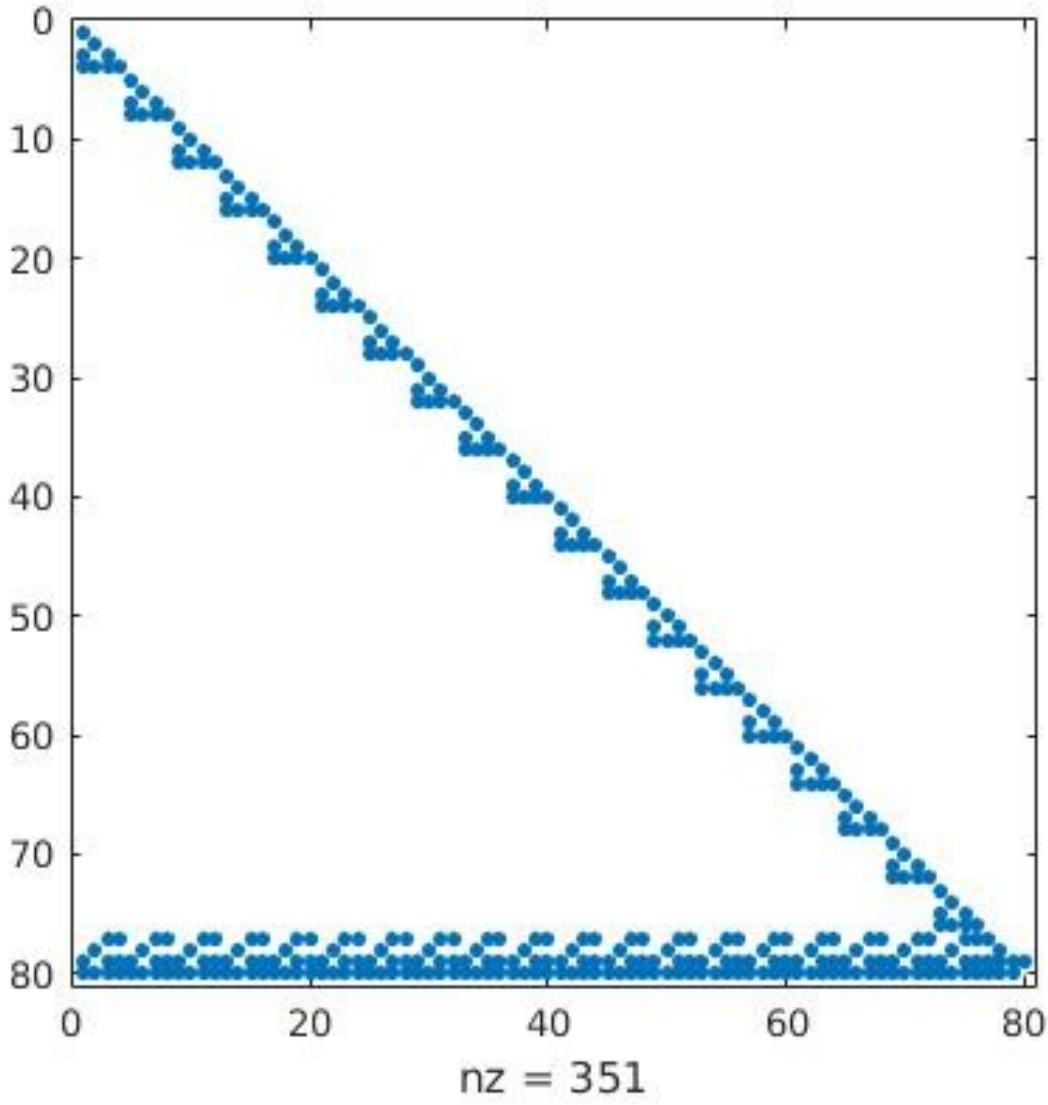
```
% Δημιουργια του μητρωου μεταθεσης W
k=1;
for i=length(A):-1:1
W(k,i)=1;
k=k+1;
end
% Το αναστροφο του W
W_T=reshape(W,length(W),[])';
% Πολλαπλασιασμος των 2 προηγουμενων μητρωων με το A ωστε να
% ερθει στη μορφη ΝΑ βελους
B=W*A*W_T;
```

Με χρήση της εντολής spy παρακάτω απεικονίζεται σε σχήμα το μητρώο:

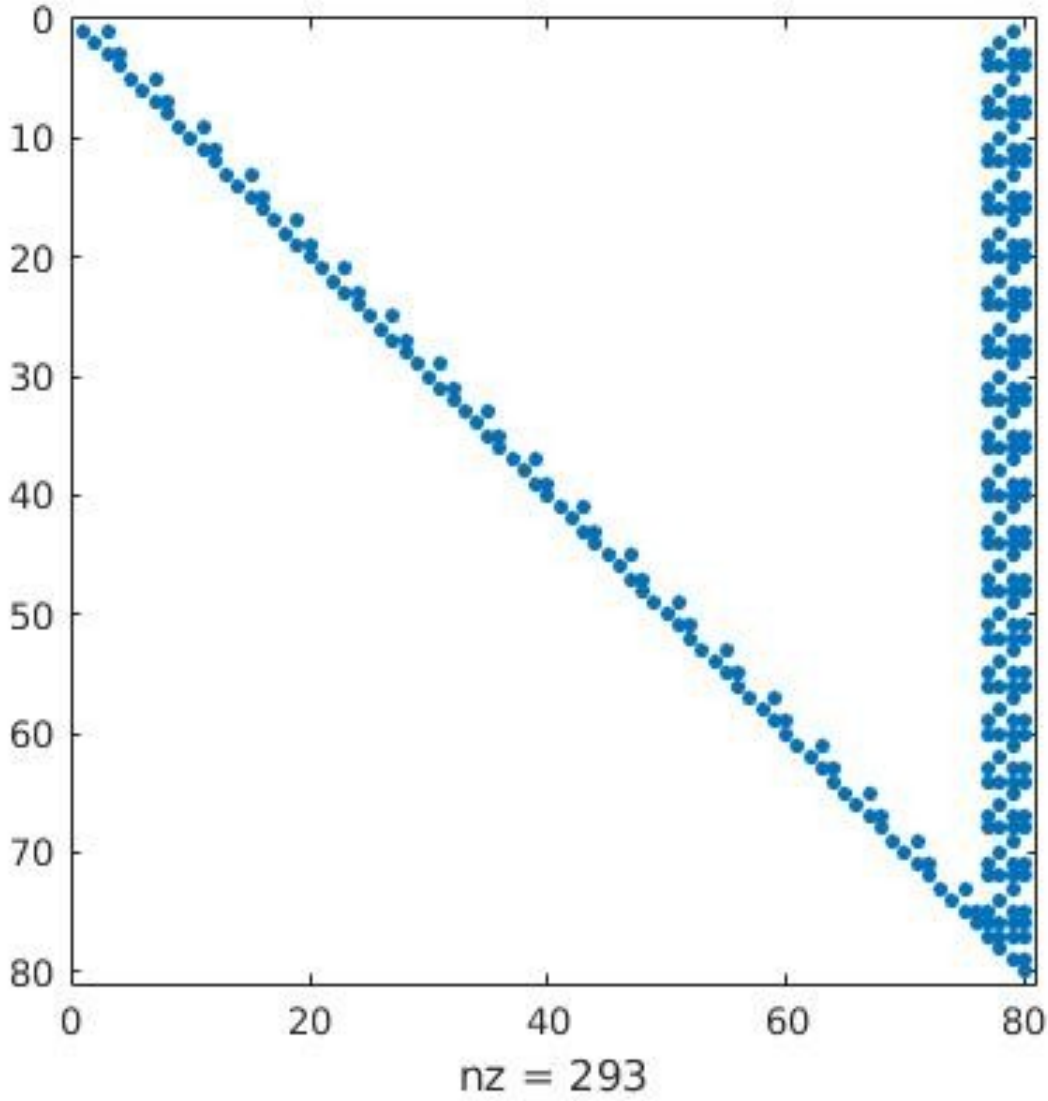


Χρησιμοποιώντας την συνάρτηση lu της MATLAB για το μητρώο B που προέκυψε παραπάνω παίρνουμε τα αντίστοιχα L και U. Παρακάτω είναι η απεικόνιση τους με χρήση ξανά της συνάρτησης spy.

L:



U:



Για να ελέγξουμε το γέμισμα που δημιουργείται συγκρίνουμε το πλήθος των μηδενικών του A με τα μη μηδενικά των L και U μείονν mn

nnz(B)=484
nnz(L)=351
nnz(U)=293

το ποσοτικοποιούμε ως:

fill_B=(nnz(L) + nnz(U) – m*n)/nnz(B)=1.4959
Είναι φανερό ότι το γέμισμα έχει μειωθεί σημαντικά, όπως περιμέναμε και με βάση τη θεωρία, καθώς ειναι σχεδόν το 1/8 σε σχέση με πριν. Συγκεριμένα έχει μειωθεί κατά:
(10.5179-1.4959)/10.5179= 0.8705 δηλαδή 87.05%

Επίσης οι παράγοντες LU έχουν σημαντικά μικρότερο γέμισμα και συγκεκριμένα το γέμισμα του L μειώθηκε (2591-351)/2591= 0.8645 δηλαδή κατά ένα 86.45% ενώ το γέμισμα του U (2630-293)/2630=0.8886 δηλαδή κατά ένα 88.86%

6. Για τον υπολογισμό του χρόνου επίλυσης του Ax=b και του WAW^TWx=Wb γίνεται χρήση της συνάρτησης timeit όπως παρατίθεται παρακάτω:

```
s=0;
for n=[125 250]
for m=[8 16]
   s=s+1;
   if s>2
      m=m/2;
   end
   T=full(sprand(m,m,0.6));
A=arrow(T,n);
W=0;
W_T=0;
tim=0;
tim2=0;
% Δημιουργια του μητρωου μεταθεσης W
k=1;
for i=length(A):-1:1
W(k,i)=1;
k=k+1;
end
% Το αναστροφο του W
W_T=reshape(W,length(W),[])';
e=ones(m*n,1);
b=A*e;
B=W*A*W_T;
t1=@() newt(A,b);
t2=@() newt(B*W,W*b);
% Υπολογισμος χρονου επιλυσης και σχετικου σφαλματος για τις δυο                 %
περιπτωσεις
x1=newt(A,b);
x2=newt(B*W,W*b);
t1=@() newt(A,b);
t2=@() newt(B*W,W*b);
tim=timeit(t1);
f_1=(norm(e-x1,inf))/norm(e,inf);
fptintf('Για την επιλυση του Ax=b με n=%d και m=%d ο χρονος ειναι:%d και το
σχετικο σφαλμα:%d\n\n', n,m,tim1,f_1);
tim2=timeit(t2);
f_2=(norm(e-x2,inf))/norm(e,inf);
```

```
fptintf('Για την επιλυση του W*A*W_T*W*x=W*b με n=%d και m=%d ο χρονος
ειναι:%d και το σχετικο σφαλμα ειναι:%d\n\n', n,m,tim2,f_2);
end
end
```

n	m	Χρόνος επίλυσης Ax=b	Χρόνος επίλυσης W AW^T W x = W b	σχετικό σφάλμα Ax=b	σχετικό σφάλμα W AW^TW x = W b
125	8	0.041744	0.077859	8.713030*(10^(-12))	1.612932*(10^(-11))
125	16	0.27198	0.532478	1.411338*(10^(-11))	1.812350*(10^(-11))
250	4	0.037352	0.074290	5.772272*(10^(-12))	6.355694*(10^(-12))
250	8	0.268580	0.535429	4.818534*(10^(-11))	7.074141*(10^(-11))

Από τις παραπάνω μετρήσεις παρατηρούμε ότι παρόλο που το μητρώο για τον υπολογισμό του W*A*W^T*x=W*b, δηλαδή το B, είναι πιο αραιό από το A όπως έχουμε δείξει σε προηγούμενο ερώτημα, ο χρόνος εκτέλεσης όπως και το σχετικό σφάλμα του υπολογισμού είναι μεγαλύτερα από τον χρόνο εκτέλεσης και το σχετικό σφάλμα του υπολογισμού A*x=b για το A.

7. Μετατρέπουμε τα μητρώα A και B σε αραιά με την χρήση της εντολής sparse και ξανατρέχουμε όλες τις χρονομετρήσεις.

n	m	Χρόνος επίλυσης Ax=b	Χρόνος επίλυσης W AW^T W x = W b	σχετικό σφάλμα Ax=b	σχετικό σφάλμα W AW^T W x = W b
125	8	0.002090	0.047707	4.937717*(10^(-12))	1.612932*(10^(-11))
125	16	0.007429	0.357559	2.633449*(10^(-13))	1.812350*(10^(-11))
250	4	0.002067	0.046499	2.328582*(10^(-12))	6.355694*(10^(-12))
250	8	0.005339	0.318709	7.555401*(10^(-12))	7.074141*(10^(-11))

Παρατηρούμε ότι ο χρόνος επίλυσης για τα αραιά μητρώα είναι μικρότερος που είναι αναμενόμενο αφού με την αραίωση που τους έγινε, η επίλυσή τους πλέον είναι πιο απλή οπότε και πιο γρήγορη.

4 ΜΕΡΟΣ Δ

1. Με χρήση της polyfit βρέθηκαν οι δύο παρακάτω συναρτήσεις για τις δύο εντολές του ερωτήματος σύμφωνα με τις χρονομετρήσεις που πραγματοποιήθηκαν.

T_X(n)=2.8802*(10^(-11))*n^3+ (1.3941*(10^8))*n^2 +(-2.6952)*(10^(-7))*n +1.693**(10^(-4))

T_QR(n)=6.4229*(10^(-11))*n^3+2.2426*(10^(-8)) *n^2 + 3.4424*(10^(-6))*n -2.2041*(10^(-4))

Ο κώδικας που εκτελέστηκε είναι ο παρακάτω:

```
i=1;
for n=[200:200:1400]
A=randn(n);
%  Υπολογισμος του χρονου εκτελεσης
X_t=@() qr(A);
t1(i)=timeit(X_t);
QR_t=@() qr(A);
t2(i)=timeit(QR_t,2);
n1(i)=n;
i=i+1;
end
p1=polyfit(n1,t1,3);
p2=polyfit(n1,t2,3);
% Μοντελοποιηση χρονου εκτελεσης
i=1;
for n=[200:200:1400]
T_X(i)=p1(1)*(n^3)+p1(2)*(n^2)+p1(3)*n+p1(4);
T_QR(i)=p2(1)*(n^3)+p2(2)*(n^2)+p2(3)*n+p2(4);
i=i+1;
end
```

2. Οι τιμές που υπολογίζονται με τις συναρτήσεις του πρώτου ερωτήματος για τα επιθυμητά n είναι οι παρακάτω:

n=200:200:1400

n	200	400	600	800	1000	1200	1400
T_X	0.0009	0.0041	0.0112	0.0236	0.0426	0.0697	0.1061
T_QR	0.0019	0.0089	0.0238	0.0498	0.0899	0.1472	0.2248

n=250:200:1750

n	250	450	650	850	1050	1250	1450	1650
T_X_2	0.0014	0.0055	0.0138	0.0277	0.0486	0.0779	0.1169	0.1671
T_QR_2	0.0030	0.0117	0.0291	0.0584	0.1025	0.1646	0.2477	0.3550

3. α) Με χρήση της συνάρτησης timeit οι χρόνοι που προέκυψαν για n=200:200:1400 είναι:

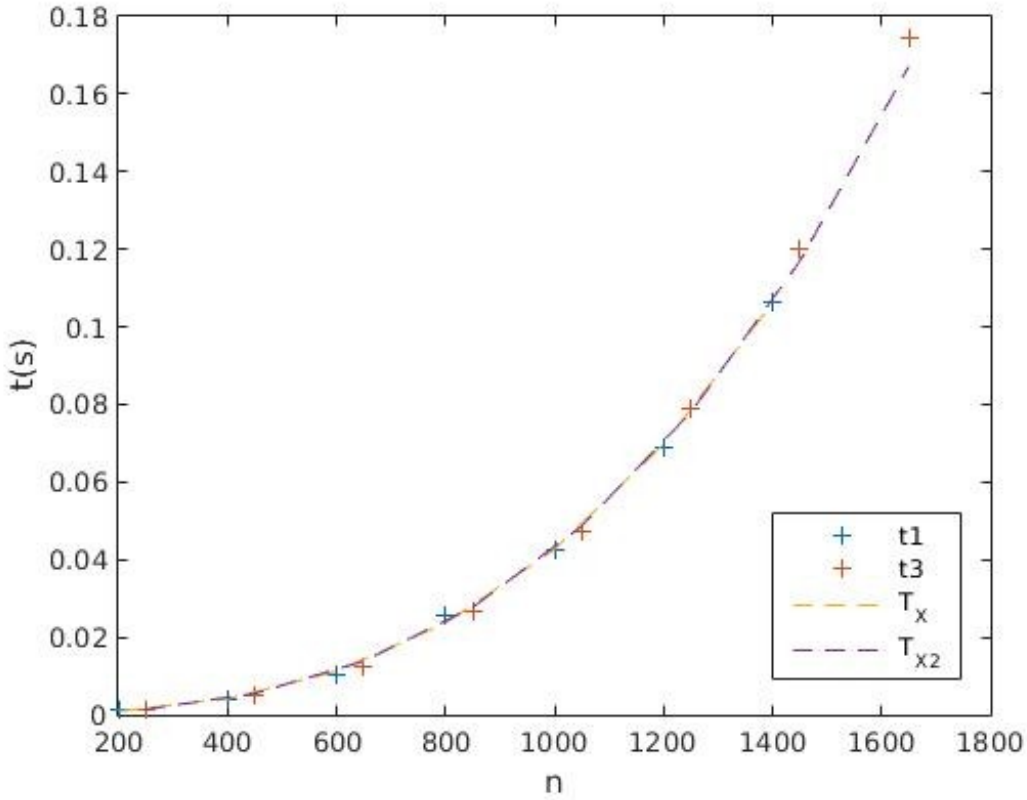
n	200	400	600	800	1000	1200	1400
t1	0.0012	0.0038	0.0103	0.0254	0.0423	0.0689	0.1065
t2	0.0020	0.0087	0.0237	0.0499	0.0901	0.1468	0.2249

β) Αντίστοιχα για n=250:200:1750:

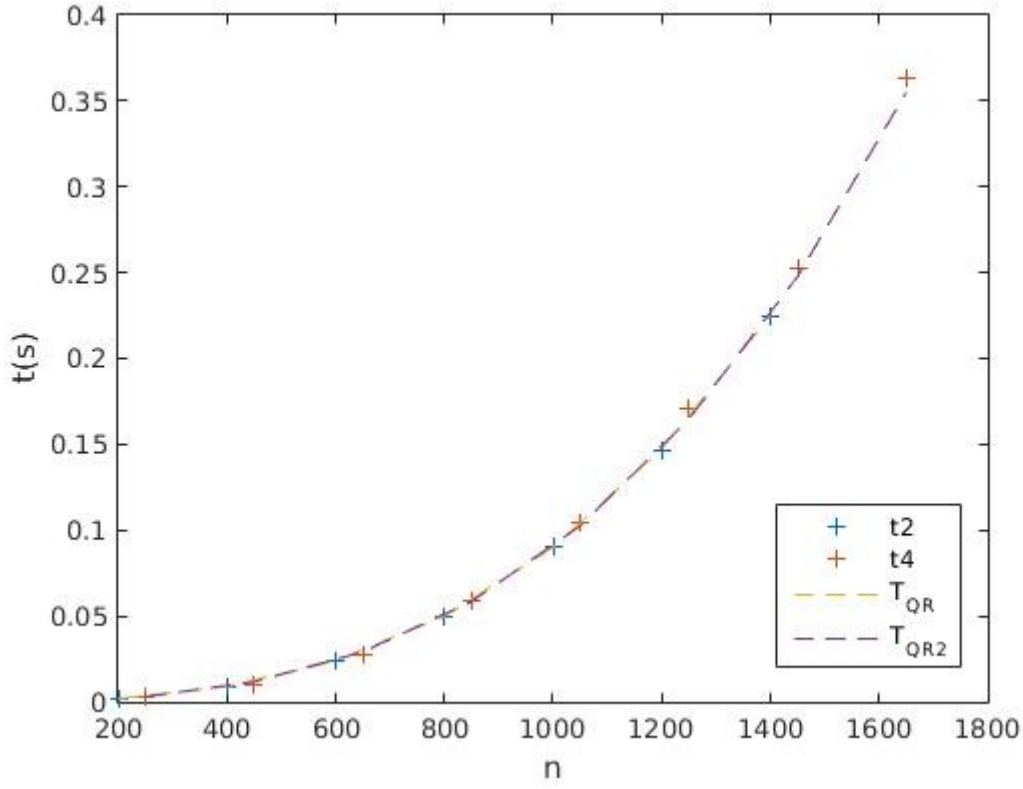
```
i=1;
for n=[250:200:1750]
A=randn(n);
% Υπολογισμος χρονου εκτελεσης
X_t=@() qr(A);
t3(i)=timeit(X_t);
QR_t=@() qr(A);
t4(i)=timeit(QR_t,2);
n1(i)=n;
i=i+1;
end
```

n	250	450	650	850	1050	1250	1450	1650
t3	0.0015	0.0049	0.0124	0.0268	0.0475	0.0788	0.1199	0.1745
t4	0.0025	0.0103	0.0274	0.0594	0.1038	0.1711	0.2529	0.3629

Η γραφική παράσταση για την εντολή X=qr(A):



Η γραφική παράσταση για την εντολή [Q,R]=qr(A):



4. Θα επαναλάβουμε το 1ο ερώτημα ζητώντας από την polyfit να κατασκευάσει πολυώνυμα 2ου και 4ου βαθμού

```
i=1;
for n=[200:200:1400]
A=randn(n);
%  Υπολογισμος του χρονου εκτελεσης
X_t=@() qr(A);
t5(i)=timeit(X_t);
QR_t=@() qr(A);
t6(i)=timeit(QR_t,2);
n1(i)=n;
i=i+1;
end
p3=polyfit(n1,t5,2);
p4=polyfit(n1,t6,2);
p5=polyfit(n1,t5,4);
p6=polyfit(n1,t6,4);
% Μοντελοποιηση χρονου εκτελεσης
i=1;
for n=[200:200:1400]
T_X_3(i)=p3(1)*(n^2)+p3(2)*n+p3(3);
T_QR_3(i)=p4(1)*(n^2)+p4(2)*n+p4(3);
T_X_4(i)=p5(1)*(n^4)+p5(2)*(n^3)+p5(3)*(n^2)+p5(4)*n+p5(5);
T_QR_4(i)=p6(1)*(n^4)+p6(2)*(n^3)+p6(3)*(n^2)+p6(4)*n+p6(5);
i=i+1;
end
```

Παρακάτω παρατίθενται οι τιμές του χρόνου των επιθυμητών συναρτήσεων για n=200:200:1400 με χρήση της timeit και οι αντίστοιχες τιμές που δίνουν τα πολυώνυμα 2ου, 3ου και 4ου βαθμού που προκύπτουν με χρήση της polyfit.

Για την συνάρτηση X=qr(A):

n	200	400	600	800	1000	1200	1400
t5	0.0008	0.0039	0.0106	0.0272	0.0538	0.0818	0.1105
2ου βαθμού	0.0019	0.0026	0.0133	0.0291	0.0510	0.0789	0.1129
3ου βαθμού	0.0017	0.0029	0.0114	0.0291	0.0527	0.0808	0.1110
4ου βαθμού	0.0009	0.0034	0.0112	0.0277	0.0529	0.0825	0.1103

Για την συνάρτηση [Q,R]=qr(A):

n	200	400	600	800	1000	1200	1400
t6	0.0016	0.0086	0.0248	0.0650	0.1308	0.1601	0.3630
2ου βαθμού	0.0046	0.0125	0.0182	0.0555	0.1046	0.2144	0.3329
3ου βαθμού	0.0043	0.0104	0.0296	0.0555	0.1176	0.1970	0.3503
4ου βαθμού	0.0032	0.0097	0.0274	0.0734	0.1219	0.1762	0.3592

Παρατηρούμε ότι όσο μεγαλώνει ο βαθμός τόσο πιο κοντά στις χρονομετρήσεις είναι οι τιμές πρόβλεψης των συναρτήσεων που είναι αναμενόμενο καθώς με την αύξηση του πολυωνύμου μεγαλώνει η ακρίβεια.