Επιστημονικός Υπολογισμός 1

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Πανεπιστήμιο Πατρών

3η Εργαστηριακή Άσκηση

Created with LaTeX

Δούκας Ιωάννης 5509

Περιεχόμενα

Χαρακτηριστικά Υπολογιστικού Συστήματος

Τα χαρακτηριστικά του υπολογιστικού συστήματος στο οποίο υλοποιήθηκαν οι ασκήσεις στο περιβάλλον matlab περιγράφονται στον παρακάτω πίνακα.

Model Name:	MacBook Pro 9.2
Processor:	Intel(R) Core(TM) i5-3210M 2.50GHz (dual core)
L2 Cache (per Core):	256 KB
L3 Cache:	3 MB
Merory (Ram):	4 GB
Operating System:	OS X 10.9.5
Matlab Version:	8.3.0.532 (R2014a) 64-bit
Διακριτότητα Χρονομετρητή:	2.9254e-07
Αποτέλεσμα LU από εντολή bench:	0.1141
mex -setup	Xcode with Clang

Παρατήρηση: Η διακριτότητα του χρόνου βρέθηκε εκτελώντας το παρακάτω script.

```
nmax = 100:100:100000;
t=0;
a = 0; b=0; c=0;
for i=0:length(nmax)
    tic;
    a = b+c;
    t = t + toc;
end
t = t/length(nmax)
```

Matlab

1 Συναρτήσεις LAPACK

Σκοπός του ερωτήματος, σε επίπεδο συστημάτων, είναι η εξοικείωση με τη δυνατότητα που δίνεται στη matlab για να αναπτύσσει προγράμματα τα οποία αξιοποιούν συνδυαστικά τα πλεονεκτήματα της matlab και τις μεγάλες δυνατότητες σε επίδοση που παρέχουν υπάρχουσες βιβλιοθήκες καθώς και κώδικες που έχουν αναπτυχθεί σε γλώσσα υψηλού επιπέδου (πχ C) και που μετασχηματίζονται σε εκτελέσιμη μορφή από μεταφραστές υψηλών δυνατοτήτων. Ειδικότερα, θα εξετάσουμε την κλήση των συναρτήσεων της βιβλιοθήκης LAPACK μέσω του περιβάλλοντος matlab χρησιμοποιώντας τη διεπαφή MEX.

1.1 Επίλυση Χρησιμοποιώντας LAPACK

Σε αυτό το σημείο θα ασχοληθούμε με ένα από τα κυριότερα προβλήματα της γραμμικής άλγεβρας το οποίο είναι η επίλυση γραμμικών συστημάτων $Ax = b$. Συγκεκριμένα αυτό το οποίο θα κάνουμε είναι χρησιμοποιώντας την υπορουτίνα dgesv της LAPACK, την οποία χρησιμοποιεί και η matlab, να λύσουμε γραμμικά συστήματα, για διαφόρους τύπους μητρώων, και να συγκρίνουμε το εμπρός σφάλμα της LAPACK με αυτό της ενδογενούς συνάρτησης mldivide της matlab. Τα μητρώα θα είναι όλα 20×20 και συγκεκριμένα το δύο πρώτα θα είναι τυχαία με δείκτες κατάστασης 10^4 και 10^10 αντίστοιχα, το τρίτο είναι τυχαίο κάτω τριγωνικό μητρώο, το τέταρτο είναι ένα μητρώο από την συνάρτηση gfpp. Για την σύγκριση των σφαλμάτων εκτελέσαμε το παρακάτω script, όπου για τις συνάρτησεις dgesv και dgesvx φαίνονται οι κώδικας αλληλεπίδρασης με την matlab σε C.

```
clc
n=20;
% matrix with condition number 10^4
fprintf('+----------------------------------------+\n');
fprintf(' Matrix with condition number 10^4 \n');
condNum = 10^4;
A = newMatrix(n,condNum);
x = ones(n,1);
b = A*x;
x_dgesv = dgesv(A,b);
x_dgesvx = dgesvx(A,b);
fprintf('\t');
forward_error_dgesv = norm(x_dgesv - x, inf)/norm(x, inf)
forward_error_dgesvx = norm(x_dgesvx - x, inf)/norm(x, inf)
x_mldivide = A\b;
forward_error_mldivide = norm(x_mldivide - x, inf)/norm(x, inf)

fprintf('+----------------------------------------+\n');
fprintf(' Matrx with condition number 10^10');
condNum = 10^10;
A = newMatrix(n,condNum);
x = ones(n,1);
b = A*x;
```

```
x_dgesv = dgesv(A,b);
x_dgesvx = dgesvx(A,b);
forward_error_dgesv = norm(x_dgesv - x, inf)/norm(x, inf)
forward_error_dgesvx = norm(x_dgesvx - x, inf)/norm(x, inf)
x_mldivide = A\b;
forward_error_mldivide = norm(x_mldivide - x, inf)/norm(x, inf)

fprintf('+---------------------------------------+\n');
fprintf(' Random Lower trangular\n');
A = tril ( rand(n) );
x = ones(n,1);
b = A*x;
x_dgesv = dgesv(A,b);
x_dgesvx = dgesvx(A,b);
forward_error_dgesv = norm(x_dgesv - x, inf)/norm(x, inf)
forward_error_dgesvx = norm(x_dgesvx - x, inf)/norm(x, inf)
x_mldivide = A\b;
forward_error_mldivide = norm(x_mldivide - x, inf)/norm(x, inf)

fprintf('+---------------------------------------+\n');
fprintf(' GFPP Matrx \n');
A = triu ( rand(n-1) );
A = gfpp(A);
x = ones(n,1);
b = A*x;
x_dgesv = dgesv(A,b);
x_dgesvx = dgesvx(A,b);
forward_error_dgesv = norm(x_dgesv - x, inf)/norm(x, inf)
forward_error_dgesvx = norm(x_dgesvx - x, inf)/norm(x, inf)
x_mldivide = A\b;
forward_error_mldivide = norm(x_mldivide - x, inf)/norm(x, inf)
```

script test_a.m

```
/*=========================================================*
 * compile with mex -v -g -largeArrayDims dgesv.c -llapack *
 *=========================================================*/
#include "mex.h"

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const ...
    mxArray *prhs[] )
{

    /* mex interface to LAPACK functions dsysvx and zhesvx */

    int m, n, bnrhs, lda, *ipiv, ldb, info;
    double *A,*b, *AT, *bT, *ret;
    int i;

    if (nrhs != 2) {
      mexErrMsgTxt("Expect 2 input arguments and return 1 ...
          output argument");
```

```
    }

    A = mxGetPr(prhs[0]);
    b = mxGetPr(prhs[1]);
    m = mxGetM(prhs[0]);
    n = mxGetN(prhs[0]);
    bnrhs = mxGetN(prhs[1]);

    AT = (double *)mxCalloc(m*n, sizeof(double));
    bT = (double *)mxCalloc(m*bnrhs, sizeof(double));
    ipiv = (int *)mxCalloc(m, sizeof(int));

    for(i=0;i<m*bnrhs;i++) {
        bT[i] = b[i];
    }

    for(i=0;i<m*n;i++) AT[i] = A[i];
    lda = m;
    ldb = m;
    dgesv(&m, &bnrhs, AT, &lda, ipiv, bT, &ldb, &info);

    plhs[0] = mxCreateDoubleMatrix(m, bnrhs,mxREAL);
    ret = mxGetPr(plhs[0]);
    for(i=0;i<m*bnrhs;i++) ret[i] = bT[i];

    mxFree(AT);
    mxFree(bT);
    mxFree(ipiv);
}
```

dgesv.c

```
/*=================================================================*
 * compile with mex -v -g -largeArrayDims dgesvx.c -llapack *
 *=================================================================*/
#include "mex.h"
#include "lapack.h"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const ...
    mxArray *prhs[])
{
    size_t n;
    int ione, info;
    char equed = 'N';
    char fact = 'N';
    char transa = 'N';
    int *ipiv, *iwork;
    double *r, *c, *X, *work, *af;
    double rcond, ferr, berr;
    int dim;
    double rcondition,forwarderror,berror;
```

```
19
20      size_t m, bnrhs, lda, ldb;
21      double *A,*b, *A2, *b2, *ret;
22      int i;
23
24      m = mxGetM(prhs[0]);
25      n = mxGetN(prhs[0]);
26      bnrhs = mxGetN(prhs[1]);
27
28      dim = n;
29      ione = 1;
30      af = (double *) mxCalloc( dim*dim , sizeof( double ) );
31      r = (double *) mxCalloc( dim , sizeof( double ) );
32      c = (double *) mxCalloc( dim , sizeof( double ) );
33      X = (double *) mxCalloc( dim , sizeof( double ) );
34      work = (double *) mxCalloc( 4* dim , sizeof( double ) );
35      ipiv = (int *) mxCalloc( dim , sizeof( int ) );
36      iwork = (int *) mxCalloc( dim , sizeof( int ) );
37
38      A = mxGetPr(prhs[0]);
39      b = mxGetPr(prhs[1]);
40
41      A2 = (double *)mxCalloc(m*n,sizeof(double));
42      b2 = (double *)mxCalloc(m*bnrhs,sizeof(double));
43
44      for(i=0;i<m*bnrhs;i++)
45      {
46          b2[i] = b[i];
47      }
48
49      for(i=0;i<m*n;i++)    A2[i] = A[i];
50
51      lda = m;
52      ldb = m;
53      dgesvx( &fact, &transa, &n, &ione, A2, &lda, af, &n, ...
            ipiv, &equed, r, c, b2, &ldb, X, &n, &rcond, &ferr, ...
            &berr, work, iwork, &info );
54
55      printf("Condition = %.20f\n",rcond );
56      printf("Backward error = %.20f\n",berr );
57      printf("Forward error = %.20f\n",ferr );
58      printf("Sintelestis afksisis = %.20f\n",work[1]);
59
60      plhs[0] = mxCreateDoubleMatrix(m,bnrhs,mxREAL);
61
62      ret = mxGetPr(plhs[0]);
63
64      for(i=0;i<m*bnrhs;i++)
65      {
66          ret[i] = X[i];
67      }
68
69      mxFree(af);
70      mxFree(r);
```

```
71      mxFree(c);
72      mxFree(X);
73      mxFree(work);
74      mxFree(ipiv);
75      mxFree(iwork);
76  }
```

dgesvx.c

Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα εκτέλεσης του script. Πα-

Εμπρός Σχετικό σφάλμα	dgesv	dgesvx	mldivide
τυχαίο με δείκτη κατάστασης 10^4	1.7546e-12	2.7660e-12	1.2803e-12
τυχαίο με δείκτη κατάστασης 10^{10}	4.2675e-12	6.9389e-13	2.8894e-12
τυχαίο κάτω τριγωνικό	3.2660e-11	1.2412e-13	1.0223e-11
gfpp	1.1708e-09	8.5931e-14	1.1775e-09

Πίνακας 1: Αποτελέσματα scipt test_a.m

ρατηρούμε ότι για κάθε τύπο μητρώου τα σφάλματα είναι ίδιας τάξης και για τις τρεις μεθόδους, εκτός από το μητρώο gfpp στο οποίο η dgesvx δείχνει να ανταποκρίνεται πολύ καλύτερα. Επίσης επισημαίνουμε και κάποιους από τους δείκτες τους οποίους επιστρέφει η dgesvx στους πίνακες 2 και 3.

	τυχαίο με δείκτη κατάστασης 10^4	τυχαίο με δείκτη κατάστασης 10^{10}
Condition	0.00003910906904724390	0.00004788914332393015
Backward error	0.00000000000000010684	0.00000000000000009405
Forward error	0.00000000020308151697	0.00000000019130438866
Sintelestis afksisis	0.00000000004402015265	0.00000000006788869874

Πίνακας 2:

	τυχαίο κάτω τριγωνικό	GFPP Matrx
Condition	0.00000161739441085323	0.00000955914367171939
Backward error	0.00000000000000005650	0.00000000000000014832
Forward error	0.00000000035552726294	0.00000000027208173566
Sintelestis afksisis	0.00000000000000233024	0.00000000000000511464

Πίνακας 3:

2 Αραιά Μητρώα, διαφορικές εξισώσεις και η CG

Στόχος αυτού του ερωτήματος είναι μία πρώτη επαφή με μία σύγχρονη μέθοδο επίλυσης μεγάλων και αραιών γραμμικών συστημάτων, τη μέθοδο συζυγών κλήσεων Conjugate Gradient (CG), και να τη συγκρίνετε με την επαναληπτική μέθοδο Jacobi για διάφορα μητρώα, ένα από τα οποία αφορά στην αριθμητική επίλυση διαφορικής εξίσωσης. Επιπλέον, έχουμε την ευκαιρία να χρησιμοποιήσετε την αραιή αναπαράσταση της matlab καθώς τα μητρώα ελέγχου είναι μητρώα ζώνης. Στο πρόβλημα αυτό συμβολίζουμε με το διάνυσμα των αγνώστων με U και το δεξιό μέλος με F.

Τα παραπάνω τα εξετάσαμε για δύο διαφορετικά μητρώα (μεγέθους 512×512). Το πρώτο μητρώο έχει προέλθει από A=hess(Q'*L*Q); όπου Q=orth(rand(n)) και το διαγώνιο μητρώο L = diag([0.7*ones(1,n/4), 1.2*ones(1,n/4), 2*ones(1,n/2)]); και το δεξιό μέλος F έχει κατασκευαστεί έτσι ώστε το σύστημα να έχει σαν λύση το διάνυσμα ones(n,1).

Το δεύτερο μητρώο έχει προέλθει από την διακριτοποίηση της διαφορικής εξίσωσης

$$-\frac{d^2u}{du^2}(x) + u(x) = 3e^x(3\sin(3x) - 2\cos(3x))$$

Το χωρίο το διακριτοποιούμε ως $\Omega_h = \{x_j = jh | j = 0 : n+1\}$ με $h = \frac{\pi}{6(n+1)}$. Από την διακριτοποίηση της διαφορικής επειδή είναι δευτέρας τάξης προκύπτει το τριδιαγώνιο μητρώο $A = \text{trid}_n[\gamma_j, \alpha_j, \beta_j]$ με $\alpha_j = \frac{2}{h^2} + c_j = \frac{2}{h^2} + 1$, $\beta_j = -\frac{1}{h^2}$ και $\gamma_j = -\frac{1}{h^2}$ και κατασκευάζεται από το παρακάτω script.

```
function A = dif_A(n, x_max, x_min)
    h = (x_max - x_min) / (n+1);
    a = 2/(h^2) + 1;
    g = -1/(h^2);
    b = -1/(h^2) ;
    A = trid(g, a, b, n);
end

function A = trid(g, a, b, n)
    A = zeros(n,n);
    A(1,1) = a;
    A(1,2) = b;
    for j=2:n-1
        A(j, j-1) = g;
        A(j, j)   = a;
        A(j, j+1) = b;
    end
    A(n, n-1) = g;
    A(n, n)   = a;
end
```

κατασκευή μητρώου A

Το δεξί μέλος του συστήματος, το διάνυσμα F, για $j = 1$ έχουμε: $F_1 = d_1 + \frac{u_0}{h^2} + b_1\frac{u_0}{2h} = 3e^h(3\sin(3h) - 2\cos(3h))$, για $j = 2, ..., n-1$ έχουμε: $F(j) = 3e^{jh} * (3\sin(3jh) - 2\cos(3jh))$ και για $j = n$ έχουμε: $F(n) = 3ne^{nh}(3\sin(3nh) - 2\cos(3nh)) - (e^{\pi/6})/(h^2)$.

```
function F = dif_F(n, x_max, x_min)
    h = (x_max - x_min) / (n+1);
    F = zeros(n,1);
    j=1;
    F(1) = 3*eps(j*h)*(3*sin(3*j*h) - 2*cos(3*j*h)) ;
    for j=2:n-1
        F(j) = 3*eps(j*h)*(3*sin(3*j*h) - 2*cos(3*j*h));
    end
    j=n;
    F(j) = 3*eps(j*h)*(3*sin(3*j*h) - 2*cos(3*j*h)) - ( ...
        exp(pi/6) ) / (h^2) ;
end
```

κατασκευή μητρώου F

Για τα μητρώα αυτά, και για τις δύο μεθόδους, εκτελούμε ένα script από το οποίο τυπώνουμε γραφικές παραστάσεις για (i) το σχετικό κατάλοιπο, (ii) το σχετικό εμπρός σφάλμα που οφείλεται στην χρήση της επαναληπτικής μεθόδου και (iii) το ολικό σχετικό εμπρός σφάλμα. Επίσης συγκρίνουμε τους χρόνους εκτέλεσης της κάθε μεθόδου για τους δύο τύπους μητρώου.

```
clc
n=512;

maxit = n;
tol = 10^(-8);
% create matrix (a)
fprintf('------------ Matrix (a) ------------\n');
x = ones(n,1);
Q = orth(rand(n));
L = diag( [0.7*ones(1,n/4), 1.2*ones(1,n/4), 2*ones(1,n/2)] );
A = hess( Q'* L * Q );
A = sparse(A);
b = A*x;
% solve Ax=b
fprintf('------------ Solve (a) ------------\n');
[x_j_a, error, iter_j_a] = jacobi(A, zeros(n,1), b, maxit, ...
    tol);
[x_cg_a, w, q, iter_cg_a ] = pcg(A, b, tol, maxit );
x_ml = A \ b;
f = @() jacobi(A, zeros(n,1), b, maxit, tol);
t_jac_a = timeit(f);
f = @() pcg(A, b, tol, maxit );
t_cg_a = timeit(f);
```

```
% sfalmata_a
fprintf('------------ Sfalmata (a) ------------\n');
figure
hold on
xlabel('sfalma');
ylabel('epanalipseis');
title('a erwtima');
% sxetika kataloipa
fprintf('------------ Sxetika kataloipa (a) ------------\n');
kat_j_a = norm(b - A*x_j_a ,2) / norm(b ,2)
plot(kat_j_a, iter_j_a, 'm*' );
kat_cg_a = norm(b - A*x_cg_a ,2) / norm(b ,2)
plot(kat_cg_a, iter_cg_a, 'g*' );
% sxetiko empros sfalma
fprintf('------------ Sxetiko embros sfalma (a) ...
    ------------\n');
sxe_j_a = norm(x_ml - x_j_a ,inf) / norm(x_ml ,inf)
plot(sxe_j_a, iter_j_a, 'c*' );
sxe_cg_a = norm(x_ml - x_cg_a ,inf) / norm(x_ml ,inf)
plot(sxe_cg_a, iter_cg_a, 'b*' );
% oliko sxetiko sfalma
fprintf('------------ Oliko sxetiko sfalma (a) ...
    ------------\n');
osx_j_a = norm(x - x_j_a ,inf) / norm(x ,inf)
plot(osx_j_a, iter_j_a, 'k*' );
osx_cg_a = norm(x - x_cg_a ,inf) / norm(x ,inf)
plot(osx_cg_a, iter_cg_a, 'r*' );
legend('sxetiko kataloipo Jacobi (a)', 'sxetiko kataloipo CG ...
    (a)', ...
    'empros sxetiko sfalma Jacobi (a)', 'empros sxetiko ...
        sfalma CG (a)', ...
    'oliko sxetiko sfalma Jacobi (a)', 'oliko sxetiko sfalma ...
       CG (a)');
hold off

figure
hold on
xlabel('sfalma');
ylabel('epanalipseis');
title('b erwtima');
fprintf('------------ Matrix (b) ------------\n');
% create matrix (b)
x_max = pi/6;
x_min = 0;
% u read solution
u = zeros(n,1);
for j=1:n
    u(j) = exp(j) * sin(3*j);
end
% solve AU=F
fprintf('------------ Solve (b) ------------\n');
A = dif_A(n, x_max, x_min);
F = dif_F(n, x_max, x_min);
[x_j_b, error, iter_j_b] = jacobi(A, zeros(n,1), F, maxit, ...
```

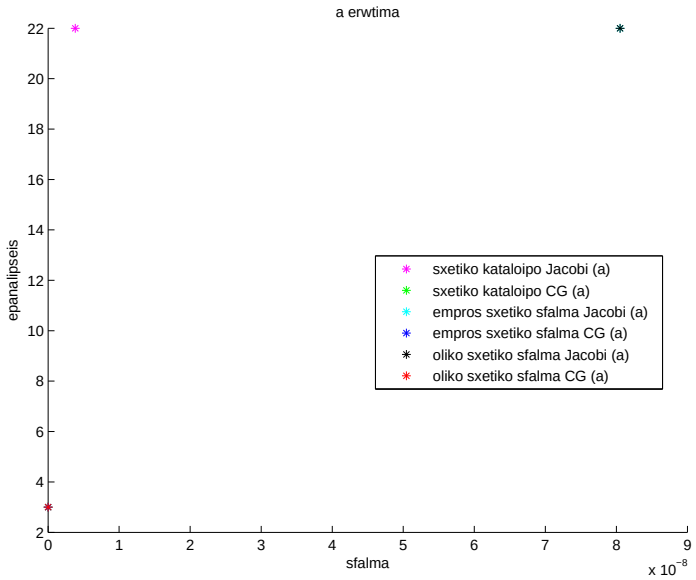
```
    tol);
[x_cg_b, w, q, iter_cg_b ] = pcg(A, F, tol, maxit );
U = A\F;
fprintf('------------ Sfalmata (b) ------------\n');
% sxetika kataloipa
fprintf('------------ Sxetika kataloipa (b) ------------\n');
kat_j_b = norm(F - A*x_j_b ,2) / norm(F ,2)
plot(kat_j_b, iter_j_b, 'm*' );
kat_cg_b = norm(F - A*x_cg_b ,2) / norm(F ,2)
plot(kat_cg_b, iter_cg_b, 'g*' );
% sxetiko empros sfalma
fprintf('------------ Sxetiko embros sfalma (b) ...
    ------------\n');
sxe_j_b = norm(U - x_j_b ,inf) / norm(U ,inf)
plot(sxe_j_b, iter_j_b, 'c*' );
sxe_cg_b = norm(U - x_cg_b ,inf) / norm(U ,inf)
plot(sxe_cg_b, iter_cg_b, 'b*' );
% oliko sxetiko sfalma
fprintf('------------ Oliko sxetiko sfalma (b) ...
    ------------\n');
osx_j_b = norm(u - x_j_b ,inf) / norm(u ,inf)
plot(osx_j_b, iter_j_b, 'k*' );
osx_cg_b = norm(u - x_cg_b ,inf) / norm(u ,inf)
plot(osx_cg_b, iter_cg_b, 'r*' );

f = @() jacobi(A, zeros(n,1), F, maxit, tol);
t_jac_b = timeit(f);
f = @() pcg(A, F, tol, maxit );
t_cg_b = timeit(f);

legend('sxetiko kataloipo Jacobi (b)', 'sxetiko kataloipo CG ...
    (b)', ...
    'empros sxetiko sfalma Jacobi (b)', 'empros sxetiko ...
        sfalma CG (b)', ...
    'oliko sxetiko sfalma Jacobi (b)', 'oliko sxetiko sfalma ...
        CG (b)', ...
    'Location', 'best');
hold off

figure
hold on
xlabel('xronos ektelesis');
ylabel('epanalipseis');
title('xronometrisi');
plot(t_jac_a, iter_j_a, 'r*');
plot(t_jac_b, iter_j_b, 'c*');
plot(t_cg_a, iter_cg_a, 'k*');
plot(t_cg_b, iter_cg_b, 'b*');
legend('xronos Jacobi (a)', 'xronos Jacobi (b)', 'xronos CG ...
    (a)', 'xronos CG (b)', 'Location', 'best');
hold off
```

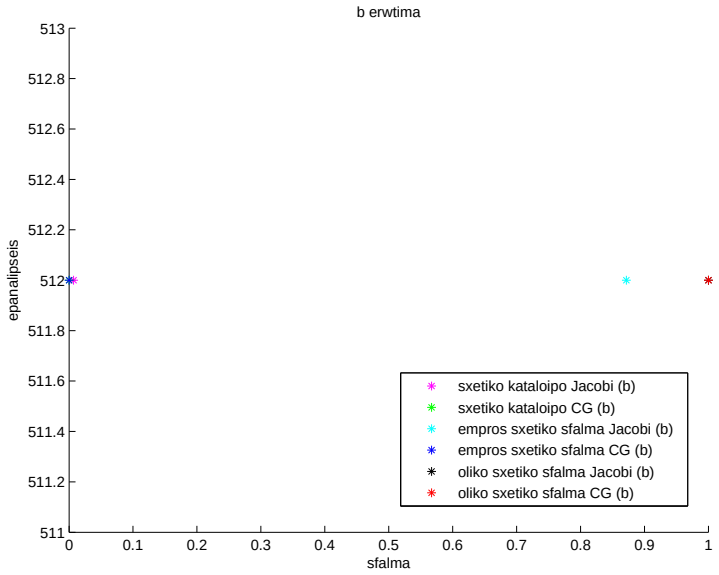
script εκτέλεσης ερωτήματος 2



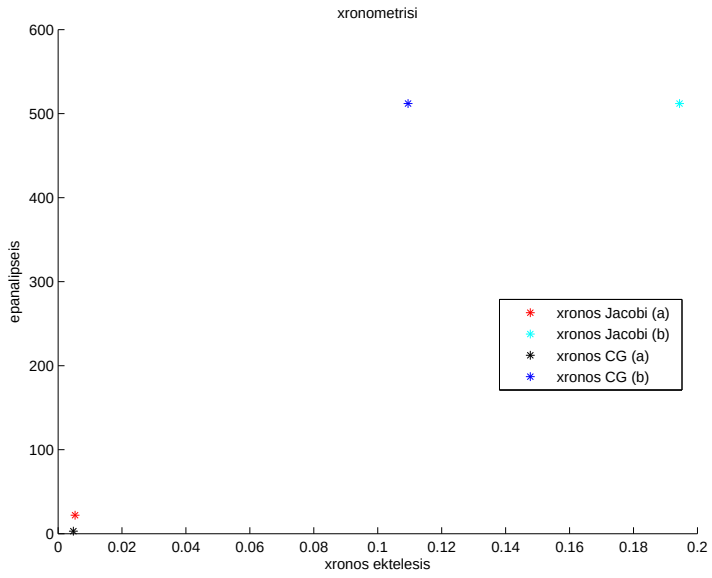
Σχήμα 2.1 : ερώτημα 3α

Όσο αναφορά τα σφάλματα των δύο μεθόδων, παρατηρούμε ότι για το μητρώο (α) την "χειρότερη" επίδοση την έχει η Jacobi καθώς έχει και πολύ μεγάλο ολικό σφάλμα και κάνει και περισσότερες επαναλήψεις σε σχέση με την CG. Επίσης βλέπουμε ότι σχεδόν όλο το σφάλμα της Jacobi οφείλεται στο εμπρός σχετικό. Η CG από την άλλη εμφανίζει πολύ μικρότερο σφάλμα και πάρα πολύ μικρό πλήθος επαναλήψεων. Όσο αναφορά το μητρώο (β) η CG δείχνει να ανταποκρίνεται πολύ καλύτερα ειδικά στο εμπρός σχετικό σφάλμα ενώ το σχετικό κατάλοιπο και των δύο μεθόδων είναι σχεδόν το ίδιο. Παρά όλα αυτά και οι δύο μέθοδοι παρουσιάζουν το ίδιο ολικό σφάλμα το οποίο είναι και αρκετά μεγάλο.

Σχετικά με τους χρόνους εκτέλεσης η CG εκτελείται πιο γρήγορα και από την Jacobi, ειδικά στην περίπτωση του τριδιαγώνιου μητρώου που προέρχεται από την διαφορική.



Σχήμα 2.2 : ερώτημα 3β



Σχήμα 2.3 : χρονομέτρηση CG και Jacobi

Βιβλιογραφία

[1] . Gilbert Strang. Εισαγωγή στη γραμμική άλγεβρα. Εκδόσεις Πανεπιστήμιο Πατρών 2012

[2] . Επιστημονικός υπολογισμός Ι. Ευστράτιος Γαλλόπουλος. 2013

[3] . *http://www.mathworks.com/help/matlab/*

[4] . Scientific Computing:An Introductory Survery. Michael T. Heath