# Δεύτερη Άσκηση στον Επιστημονικό Υπολογισμό

Κων/νος Αραβανής ΑΜ: 3628
mail: arabanis@ceid.upatras.gr

24 Νοεμβρίου 2008

1. Αρχικά θα περιγράψουμε το σύστημα στο οποίο τρέξαμε τα πειράματα μας: Το σύστημα μας είχε το λειτουργικό Linux/Debian Lenny και ο επεξεργαστής του ήταν ένας Intel Core Duo T2300 στα 1,66GHz. Είχαμε μία ιεραρχεία μνήμης δύο επιπέδων με το πρώτο επίπεδο να αποτελείτε από δύο κρυφές μνήμες, η μία για data (L1 D-Cache) και η άλλη για inst. (L1 D-Cache) και οι δύο των 32KBytes και με τα ακόλουθα χαρακτηριστικά: 8 way set-associative, 64-byte line size. Η κύρια μνήμη (L2 Cache) μας είχε μέγεθος 2048KBytes και ακριβώς τα ίδια χαρακτηριστικά με τις κρυφές. Όσον αφορά το λογισμικό στο οποίο εκτελέστηκαν τα προγράμματα μας ήταν η Matlab 7.6.0(R2008a). Να σημειωθεί ότι η αναφορά έγινε σε LaTeX.

2. (α′) Στο ερώτημα αυτό φτιάξαμε μία βοηθητική συνάρτηση .m (y = my_horner( A, x )) η οποία σαν είσοδο παίρνει τους συντελεστές της δυναμομορφής καθώς και το κατάλληλο x και επιστρέφει την τιμή του πολυωνύμου με τη βοήθεια του αλγορίθμου Horner. Η συνάρτηση αυτή υλοποιήθηκε έτσι ώστε να κάθε εδιάμεσο αποτέλεσμα του Horner να αποθηκεεται πρώτα, έτσι ώστε να μην χρησιμοποιήται η αριθμητική εκτεταμένης ακρίβειας (80 bits) του Pentium.

   Ο κώδικας παρουσιάζεται παρακάτω (my_horner.m):

```
function y = my_horner( A, x )
%horner( A, x ): Sinartisi stin opoia dinoume san eisodous tous sintelestes
%                enos poluonimou se ena dianusma A kai mia timi x kai i
%                my_horner epistrefei tin timi tou polionimou. Na tonisoume
%                oti sto A(1) tha prepei na vrisketai o sintelestis
%                a_(n-1)kai sto A(n) o a_0

n = length(A);
y = A(1);
single(y);
for i = 2:1:n
    y = single(y * x);
    y = single(y + A(i));
end
end
```

Στο σημείο αυτό καλούμαστε να αποδείξουμε την πίσω ευστάθεια του Horner.
Για ευκολία παραθέτουμε τον Horner όπως ακριβώς είναι:

$s_n = a_n$
$fork = n-1 : -1 : 0$
$s_k = x * s_{k+1} + a_k$
$end$
Οι υπολογισμένες τιμές στο μοντέλο αριθμητικής κινητής υποδιαστολής είναι:

$s_{n-1} = (x * s_n < 1 > + a_{n-1}) < 1 >$
$= x * a_n < 2 > + a_{n-1} < 1 > s_{n-2} = (x * s_{n-1} < 1 > + a_{n-2}) < 1 >$
...
$s_0 = a_0 < 1 > a_1 * x < 3 > + ... a_{n-1} * x^{n-1} < 2n-1 > + a_n * x^n < 2n >$
$= (1+\theta_1) * a_0 + ... (1+\theta_{2n}) * a_n * x^n$

άρα
$s_0 = f_{prog}(a_0, ..., a_n, x)$
$= f(a_0(1+\theta_1), ..., a_n(1+\theta_{2n}), x)$

πράγμα που αποδεικνύει την πίσω ευστάθεια του Horner.

(β′) Στο ερώτημα αυτό κληθήκαμε να φτιάξουμε μία συνάρτηση η οποία να υπολογίζει τον δείκτη κατάστασης των δύο δοθέντων συναρτήσεων, των πολυωνύμων Willkinson και Grcar καθώς και των δύο προβλημάτων πολλαπλασιασμού μητρώου με διάνυσμα.Από την θεωρία γνωρίζουμε ότι τον δείκτη κατάστασης

i. μίας συνάρτησης μας τον επιστρέφει η λυση του τύπου $x * f'(x)/f(x)$
ii. ενός πολυωνύμου η $\Sigma(|a_j * x^j|/|p(x)|)$ όταν το $x$ πρόκειται για κάποια τυχαία τιμή της συνάρτησης και η $(1/(x * p'(x))) * \Sigma|a_j * x^j|$ όταν το $x$ πρόκειται για ρίζα της συνάρτησης (Και στα δύο για $j = 1:1:n-1$).
iii. ενός μητρώου $W$ (Vandermonde) μπορεί έυκολα να βρεθεί από την $cond(W)$ της Matlab.

Ο κώδικας που γράφτηκε ήταν ο ακόλουθως (my_cond.m):

```
function condition = my_cond( what, x )
%my_cond( what, x ): Sinartisi pou epistrefei tin deikti katastasi ton 6
%                    provlimaton pou perigrafontai apo tin ekfonisi. An
%                    thesete what = 1 tote epistrefete o deiktis
%                    katastaseis tis f1 gia to analogo x pou thesate an
%                    valete what = 2 tis f2, what = 3 tis Willkinson, what
%                    = 4 tis Grcar, kai outo kathekseis... mexri to 6.

if what==1
    f1 = ( 1 - cos( x ) ) / x^2;
    %i paragogos tou f1 einai i f1_par
```

```
    f1_par = ( x * sin( x ) - 2 * ( 1 - cos( x ) ) ) / x^3;
    condition=abs( x * f1_par / f1 );
elseif what==2
    f2 = 2 * ( sin( x / 2 ) )^2 / x^2;
    %i paragogos tou f2 einai i f2_par
    f2_par = ( 2 * sin ( x / 2 ) * cos( x / 2 ) * x - 4 * ( sin( x / 2 ) )^2 ) / x^3;
    condition= abs(x * f2_par / f2);
elseif what==3
    n = 13;
    p_willkinson = poly( 1:n-1 );
    if is_root(p_willkinson,x)==1
        p_par_willkinson = polyder( p_willkinson );
        condition = 0;
        for i = 1:1:n-1
            condition = condition + abs(p_willkinson(i) * x^i);
        end
        condition = condition / abs( x * polyval(p_par_willkinson, x) );
    else
        condition = 0;
        p_willkinson_x_abs = abs( polyval( p_willkinson, x ) );
        for i = 1:1:n
            condition = condition + abs(p_willkinson(i) * x^i) / p_willkinson_x_abs;
        end
    end
elseif what==4
    n = 13;
    p_grcar = poly(eig(gallery('grcar',12)));
    if is_root(p_grcar,x)==1
        p_par_grcar = polyder( p_grcar );
        condition = 0;
        for i = 1:1:n-1
            condition = condition + abs(p_grcar(i) * x^i);
        end
        condition = condition / abs( x * polyval(p_par_grcar, x) );
    else
        condition = 0;
        p_grcar_x_abs = abs( polyval( p_grcar, x ) );
        for i = 1:1:n
            condition = condition + abs(p_grcar(i) * x^i) / p_grcar_x_abs;
        end
    end
elseif what==5
    W=[ones(9,1),x',(x.^2)',(x.^3)',(x.^4)',(x.^5)',(x.^6)',(x.^7)',(x.^8)'];
    condition=cond(W);
elseif what==6
    W=[ones(9,1),x',(x.^2)',(x.^3)',(x.^4)',(x.^5)',(x.^6)',(x.^7)',(x.^8)'];
    condition=cond(W);
else
    'I timi tou what den itan orthi. Gia tin akriveia:'
    help my_cond
end
end
```

(is_root.m)

```
function y = is_root(A,x)
%is_root(A,x): i is_root pairnei san orismata tous sintelestes enos
%              polionimou(A) kai kapoio x kai epistrefei 1 an to x einai
%              riza tou polionimou i 0 an den einai.

y=0;
B=single(roots(A));
n =length(B);
for i = 1:1:n
    if B(i)==x
```

```
        y=1;
    end
end
end
```

Ο υπολογισμός του δείκτη κατάστασης γίνεται με διπλή ακρίβεια γιατι η Matlab σαν προεπιλογή της εκτελέι της πράξεις με διπλή ακρίβεια εκτός και αν κάποια πράξη δηλωθεί single() πράγμα το οποίο δεν γίνεται στον κώδικα μας.

(γ′) Στο ερώτημα αυτό μας ζητήθηκε να γράψουμε συνάρτησεις οι οποίες να υπολογίζουν τις τιμές των $f_1, f_2, Willkinson, Grcar, mtimes$ σε μονή και διπλή ακρίβεια. Για το λόγω αυτό κάναμε τις ακόλουθες τρείς συναρτήσεις:

(my_val_f.m):

```
function y=my_val_f(what,x,p)
%my_val_f(what,x,p): Sinartisi pou epistrefei tin timi ton sinartiseon f1,
%                    f2 pou perigrafontai apo tin ekfonisi. An thesete
%                    what = 1
%                    tote epistrefete i timi tis f1 gia to analogo x pou
%                    thesate, eno an valete
%                    what = 2
%                    tis f2.
%                    To p afora tin percission an p = 1 tote tha einai monis
%                    kai an p = 2 diplis

if p==2
    if what==1
        y = ( 1 - cos( x ) ) / x^2;
    elseif what==2
        y = 2 * ( sin( x / 2 ) )^2 / x^2;
    else
        'I timi tou what den itan orthi. Gia tin akriveia:'
        help my_val_p
    end
elseif p==1
    if what==1
        y = single(( 1 - cos( x ) ) / x^2);
    elseif what==2
        y = single(2 * ( sin( x / 2 ) )^2 / x^2);
    else
        'I timi tou what den itan orthi. Gia tin akriveia:'
        help my_val_p
    end
else
    'I timi tou p den itan orthi. Gia tin akriveia:'
    help my_val_p
end
end
```

(my_val_p.m):

```
function y=my_val_p(what,x,p)
%my_val_p(what,x,p): Sinartisi pou epistrefei tin timi ton polionimon
%                    Willkinson, Grcar pou perigrafontai apo tin ekfonisi.
%                    An thesete
%                    what = 1
%                    tote epistrefete i timi tou protou gia to analogo x pou
%                    thesate, eno an valete
%                    what = 2
%                    tou Grcar.
```

```
%                   To p afora tin percission an p = 1 tote tha einai monis
%                   kai an p = 2 diplis

if p==2
    if what==1
        y=polyval(poly(1:12),x);
    elseif what==2
        y = polyval(poly(eig(gallery('grcar',12))),x);
    else
        'I timi tou what den itan orthi. Gia tin akriveia:'
        help my_val_p
    end
elseif p==1
    if what==1
        y=my_horner(poly(1:12),x);
    elseif what==2
        y = my_horner(poly(eig(gallery('grcar',12))),x);
    else
        'I timi tou what den itan orthi. Gia tin akriveia:'
        help my_val_p
    end
else
    'I timi tou p den itan orthi. Gia tin akriveia:'
    help my_val_p
end
end
```

(my_val_W.m):

```
function y=my_val_W(x,p,b)
%my_val_W(x,p,b): Sinartisi pou epistrefei tin timi mtimes(W,b) gia
%                  ta analoga x kai b
%                  To p afora tin percission an p = 1 tote tha einai monis
%                  kai an p = 2 diplis

if p==2
    W=[ones(9,1),x',(x.^2)',(x.^3)',(x.^4)',(x.^5)',(x.^6)',(x.^7)',(x.^8)'];
    y=mtimes(W,b);
elseif p==1
    W=single([ones(9,1),x',(x.^2)',(x.^3)',(x.^4)',(x.^5)',(x.^6)',(x.^7)',(x.^8)']);
    y=single(mtimes(W,b));
else
    'I timi tou p den itan orthi. Gia tin akriveia:'
    help my_val_W
end
end
```

i. $my\_val\_f$:Συνάρτηση που επιστρέφει τη τιμή $f_1(x)$ ή την $f_2(x)$ αφού δοθεί σαν είσοδος το $x$. Το αποτέλεσμα που επιστρέφεται μπορεί να έχει προκύψει είτε απο πράξεις με μονή είτε με διπλή ακρίβεια.

ii. $my\_val\_p$:Συνάρτηση που επιστρέφει τη τιμή του πολυωνύμου $Wilkinson$ ή του $Grcar$ για δεδομένο $x$. Το αποτέλεσμα που επιστρέφεται μπορεί να έχει προκύψει είτε απο πράξεις με μονή είτε με διπλή ακρίβεια.

iii. $my\_val\_W$:Συνάρτηση που επιστρέφει τη τιμή $mtimes(W, b)$ αφού δοθεί σαν είσοδος το $b$ και το $x$ που υπολογίζει το $W$. Το αποτέλεσμα που επιστρέφεται μπορεί να έχει προκύψει είτε απο πράξεις με μονή είτε με διπλή ακρίβεια.

3. Στο ερώτημα αυτό υλοποιήσαμε κώδικα που να βρίσκει το εμπρός σφάλμα των $f_1$ και $f_2$ για $x = 10.^[-10:1:1]$ και για των πολυωνύμων Willkinson και Grcar για $x = [-1, 0.5, 1, 2]$. Ο κώδικας μας αυτό που κάνει στην ουσία είνα να βρίσκει το σχετικό σφαλμα $(||f_{prog}(x*) - f(x)||/||f(x)||)$ εκτός και αν ο παρανομαστής του κλάσματος μηδενιζεται και τότε βρίσκει το απόλυτο $(||f_{prog}(x*) - f(x)||)$.

   Ο κώδικας που γράψαμε φαίνεται παρακάτω(ask2_3.m):

```
X1=10.^[-10:1:1];
X2=[-1,0.5,1,2];

n1=length(X1);
n2=length(X2);

sfalma_f1=zeros(n1,1);
sfalma_f2=zeros(n1,1);
sfalma_willkinson=zeros(n2,1);
sfalma_grcar=zeros(n2,1);

for i=1:1:n1
    if my_val_f(1,X1(i),2)==0
        %apoluto sfalma
        sfalma_f1(i,1)=abs( my_val_f(1,X1(i),1) - my_val_f(1,X1(i),2) );
    else
        %sxetiko sfalma
        sfalma_f1(i,1)=abs( my_val_f(1,X1(i),1) - my_val_f(1,X1(i),2) ) / abs( my_val_f(1,X1(i),2) );
    end

    if my_val_f(2,X1(i),2)==0
        %apoluto sfalma
        sfalma_f2(i,1)=abs( my_val_f(2,X1(i),1) - my_val_f(2,X1(i),2) );
    else
        %sxetiko sfalma
        sfalma_f2(i,1)=abs( my_val_f(2,X1(i),1) - my_val_f(2,X1(i),2) ) / abs( my_val_f(2,X1(i),2) );
    end
end

for i=1:1:n2
    if my_val_p(1,X2(i),2)==0
        %apoluto sfalma
        sfalma_willkinson(i,1)=abs( my_val_p(1,X2(i),1) - my_val_p(1,X2(i),2) );
    else
        %sxetiko sfalma
        sfalma_willkinson(i,1)=abs( my_val_p(1,X2(i),1) - my_val_p(1,X2(i),2) ) / abs( my_val_p(1,X2(i),2) );
    end

    if my_val_p(2,X2(i),2)==0
        %apoluto sfalma
        sfalma_grcar(i,1)=abs( my_val_p(2,X2(i),1) - my_val_p(2,X2(i),2) );
    else
        %sxetiko sfalma
        sfalma_grcar(i,1)=abs( my_val_p(2,X2(i),1) - my_val_p(2,X2(i),2) ) / abs( my_val_p(2,X2(i),2) );
    end
end

sfalma_f1
sfalma_f2
sfalma_willkinson
sfalma_grcar
```

4. Στο ερώτημα αυτό υπολογίζουμε αριθμητικά το δείκτη κατάστασης για όλα τα προβηματα που τέθηκαν και σε όλα τα δεδομένα που δοθηκαν για το κάθε πρόβλημα.

   Ο κώδικας που χρησιμοποιήσαμε για αυτό είναι ο ακόλουθως(ask2_4.m):

```
%provlima 1
X1=10.^[-10:1:1];
n1=length(X1);
condition_f1=zeros(n1,1);
condition_f2=zeros(n1,1);

for i=1:1:n1
    %f1
    condition_f1(i,1)=my_cond(1,X1(i));
    %f2
    condition_f2(i,1)=my_cond(2,X1(i));
end

%ektiposi ton condition ton f1 kai f2 antistoixa
condition_f1
condition_f2

%provlima 2
X2=[-1,0.5,1,2];
n2=length(X2);
condition_Willkinson=zeros(n2,1);
condition_Grcar=zeros(n2,1);

for i=1:1:n2
    %Willkinson
    condition_Willkinson(i,1)=my_cond(3,X2(i));
    %Grcar
    condition_Grcar(i,1)=my_cond(4,X2(i));
end

%ektiposi ton condition ton Willkinson kai Grcar antistoixa
condition_Willkinson
condition_Grcar

%provlima 3
X3=-1+[0:8]/11;
X4=-cos([0:8]*pi/9);
condition_W1=my_cond(5,X3)
condition_W2=my_cond(6,X4)
```

   Οι τιμές που πήραμε μετά την εκτέλεση του παραπάνω κώδικα ήταν οι ακόλουθες:

| $x$ | $cond(f_1, x)$ | $cond(f_2, x)$ |
|---|---|---|
| $10^{-10}$ | $Inf$ | 0 |
| $10^{-9}$ | $Inf$ | 0 |
| $10^{-8}$ | $Inf$ | 0 |
| $10^{-7}$ | 0.0016 | 0.0000 |
| $10^{-6}$ | 0.0002 | 0.0000 |
| $10^{-5}$ | 0.0000 | 0.0000 |
| $10^{-4}$ | 0.0000 | 0.0000 |
| $10^{-3}$ | 0.0000 | 0.0000 |
| $10^{-2}$ | 0.0000 | 0.0000 |
| $10^{-1}$ | 0.0017 | 0.0017 |
| 1 | 0.1695 | 0.1695 |
| 10 | 4.9581 | 4.9581 |

| $x$ | $cond(P_{Wilkinson}, x)(*10^5)$ | $cond(P_{Grcar}, x)(*10^5)$ |
|---|---|---|
| $-1$ | 0.0000 | 0.0000 |
| 0.5 | 0.0000 | 0.0000 |
| 1 | 0.0014 | 0.0192 |
| 2 | 16.3797 | 8.3913 |

| $x$ | $cond(W)$ |
|---|---|
| $-1 + [0:8]/11$ | $6.0164 * 10^7$ |
| $-cos([0:8] * pi/9)$ | $1.0601 * 10^3$ |

Εδώ να πούμε ότι όπου στους παραπάνω πίνακες εμφανίζεται το 0.0000 δε πρόκειται για αριθμό ακριβώς 0 αλλα για μηδενικό που προέκυψε λόγω απαλοιφής κάποιον ψηφίων (χρησιμοποιήθηκε format short). Αντίθετα όπου εμφανίζεται το 0 πρόκειται για πραγματικό 0.

5. Στο ερώτημα αυτό της άσκησης κληθήκαμε να υπολογίσουμε τις τιμές των πολυωνύμων Wilkinson και Grcar στα σημεία $[-1, 0.5, 1, 2]$ καθώς και τις ρίζες τους. Ακολούθως αφού εφαρμόσουμε στους συντελεστές τις ακόλουθες μεταβολές:

   $(\alpha')$ $|\Delta a_j| \approx (1.0e-07) randn(1,1)|a_j|$

   $(\beta')$ $|\Delta a_j| \approx (1.0e-05) randn(1,1)|a_j|$

   $(\gamma')$ $|\Delta a_j| \approx (1.0e-03) randn(1,1)|a_j|$

   Να ξαναυπολογισουμε τις παραάνω τιμές και τις ρίζες για τους νέους συντελεστές.

Οι κώδικες που γράψαμε για αυτό το ερώτημα ήταν οι ακόλουθοι:
(ask2_5.m):

```
%to X pou dinetai apo tin askisi
X=[-1;0.5;1;2];
%oi rizes tou Willkinson
X1_w=roots(poly(1:12));
%oi rizes tou grcar
X1_g=roots(poly(eig(gallery('grcar',12))));
%oi sintelestes ton poluonimon
A_w=poly(1:12);
A_g=poly(eig(gallery('grcar',12)));
%oi times ton poluonimon gia X
Y1_w=my_val_p_total(A_w,X);
Y1_w=my_val_p_total(A_g,X);

%upologismos ton timon me tis metavoles a
metavoli = 1.0e-07;
A_w=new_a(A_w,metavoli);
A_g=new_a(A_g,metavoli);
X2_w=roots(A_w);
X2_g=roots(A_g);
Y2_w=my_val_p_total(A_w,X);
Y2_g=my_val_p_total(A_g,X);

%upologismos ton timon me tis metavoles b
metavoli = 1.0e-05;
A_w=new_a(A_w,metavoli);
A_g=new_a(A_g,metavoli);
X3_w=roots(A_w);
X3_g=roots(A_g);
Y3_w=my_val_p_total(A_w,X);
Y3_g=my_val_p_total(A_g,X);

%upologismos ton timon me tis metavoles c
metavoli = 1.0e-03;
A_w=new_a(A_w,metavoli);
A_g=new_a(A_g,metavoli);
X4_w=roots(A_w);
X4_g=roots(A_g);
Y4_w=my_val_p_total(A_w,X);
Y4_g=my_val_p_total(A_g,X);

%ektiposi ton rizon Willkinson me tropo pou na mporoume na diaakrinoume tis
%metavoles tous me tin metavoli ton sinteleston
x_w=ones(n_willkinson,1);
plot(x_w,X1_w,'r+');
hold on;
plot(2.*x_w,X2_w,'x');
plot(3.*x_w,X3_w,'g.');
plot(4.*x_w,X4_w,'y*');
legend('xoris metavoli','1.0e-07','1.0e-05','1.0e-03');
hold;

%ektiposi ton rizon grcar me tropo pou na mporoume na diaakrinoume tis
%metavoles tous me tin metavoli ton sinteleston
figure;
x_g=ones(n_grcar,1);
plot(x_g,X1_g,'r+');
hold on;
plot(2.*x_g,X2_g,'x');
plot(3.*x_g,X3_g,'g.');
plot(4.*x_g,X4_g,'y*');
```

```
legend('xoris metavoli','1.0e-07','1.0e-05','1.0e-03');
hold;

%taksinomisi ton rizon me vasi tin euaisthisia tous stis metavoles pou ginan
%apo ti riza me ti megaliteri euaistisia pros ti riza me ti mikroteri
S1_w = sort_euaisthisia(X1_w,X2_w)
S2_w = sort_euaisthisia(X1_w,X3_w)
S3_w = sort_euaisthisia(X1_w,X4_w)

S1_g = sort_euaisthisia(X1_g,X2_g)
S2_g = sort_euaisthisia(X1_g,X3_g)
S3_g = sort_euaisthisia(X1_g,X4_g)
```

(my_val_p_total.m):

```
function Y = my_val_p_total(A,X)
n=length(X);

Y=zeros(n,1);

for i=1:1:n
    Y(i,1)=polyval(A,X(i,1));
end

end
```

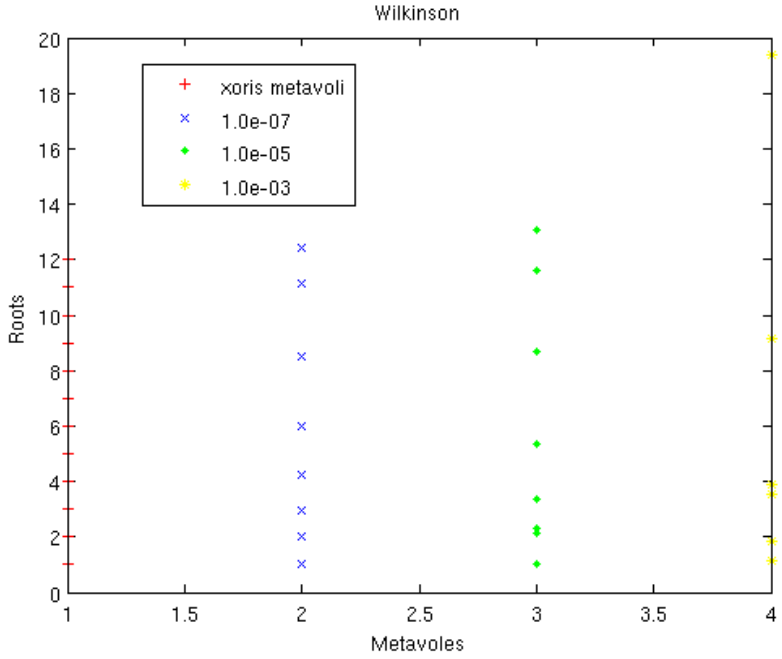
(new_a.m):

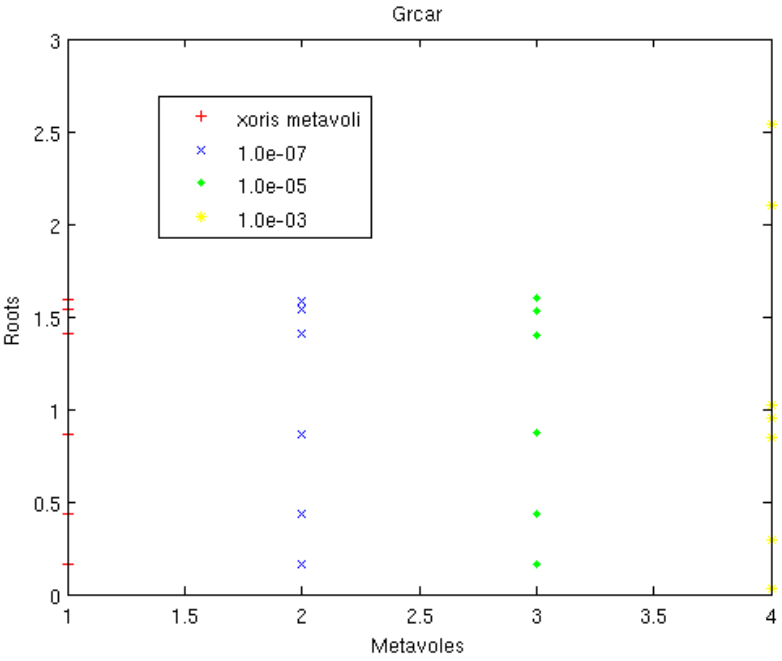
```
function A=new_a(A,metavoli)

n=length(A);

for i=1:1:n
    A(1,i)=abs(A(1,i)) * metavoli*randn(1,1) + A(1,i);
end

end
```

(sort_euaisthisia.m):

```
function A = sort_euaisthisia(X_r,X_r_changed)

n=length(X_r);
A=zeros(n,1);

Diafora=abs(X_r-X_r_changed);
[Diafora,Seira]=sort(Diafora,'descend');

for i=1:1:n
    A(i,1)=X_r(Seira(i),1);
end
end
```

Παρακάτω παρουσιάζονται οι γραφικές παραστάσεις για το κάθε πολυώνυμο που δείχνουν τι θέση των ριζών τους με και χωρίς τις μεταβολές. Συγκεκριμένα παρουσιάζονται με τέτοιο τρόπο ώστε να μπορούμε να παρατηρήσουμε τις διαφορές τους.




Λέγοντας μεταβολή =

(α′) 1, εννοούμε οτι δεν έχει συμβεί κάποια μεταβολή

(β′) 2, εννοούμε την μεταβολή *(α)*

(γ′) 3, εννοούμε την μεταβολή *(β)*

(δ′) 4, εννοούμε την μεταβολή *(γ)*

Οι ρίζες της Wilkinson για τις μεταβολές *(α),(β),(γ)* ταξινομημένες βάση την ευαισθησία τους φαίνονται παρακάτω:

(α′) 9.0000
8.0000
10.0000
7.0000
6.0000
11.0000
5.0000
4.0000
12.0000
3.0000
2.0000
1.0000

(β′) 8.0000
9.0000
7.0000
11.0000
6.0000
12.0000
5.0000
10.0000
4.0000
3.0000
2.0000
1.0000

(γ′) 11.0000
12.0000
10.0000
9.0000
8.0000
7.0000
6.0000
5.0000
4.0000
3.0000
2.0000
1.0000

Οι ρίζες της Grcar για τις μεταβολές *(α),(β),(γ)* ταξινομημένες βάση την ευαισθησία τους φαίνονται παρακάτω:

(α′)
$1.5878 + 0.9619i$
$1.5878 - 0.9619i$
$1.4081 + 1.1320i$
$1.4081 - 1.1320i$
$1.5396 + 0.3476i$
$1.5396 - 0.3476i$
$0.8650 + 1.4976i$
$0.8650 - 1.4976i$
$0.4356 + 1.8913i$
$0.4356 - 1.8913i$
$0.1639 + 2.1661i$
$0.1639 - 2.1661i$

(β′)
$1.5878 + 0.9619i$
$1.5878 - 0.9619i$
$1.4081 + 1.1320i$
$1.4081 - 1.1320i$
$1.5396 + 0.3476i$
$1.5396 - 0.3476i$
$0.8650 + 1.4976i$
$0.8650 - 1.4976i$
$0.4356 + 1.8913i$
$0.4356 - 1.8913i$
$0.1639 + 2.1661i$
$0.1639 - 2.1661i$

(γ′)
$1.4081 - 1.1320i$
$1.5878 + 0.9619i$
$1.5878 - 0.9619i$
$1.4081 + 1.1320i$
$1.5396 + 0.3476i$
$0.8650 + 1.4976i$
$0.8650 - 1.4976i$
$0.4356 + 1.8913i$
$0.4356 - 1.8913i$
$1.5396 - 0.3476i$
$0.1639 + 2.1661i$
$0.1639 - 2.1661i$

Να σημειωθεί εδώ ότι κάθε φορα που θα εκτελούμε τον παραπάνω κώδικα θα παίρνουμε και διαφορετικές τιμές μιας και υπάρχει στο κώδικα μας η συνάρτηση

*randn* που επηρεάζει σε τελείως διαφορετική μορφή τα αποτελέσματα μας κάθε φορά. Βέβαια για λόγους που εύκολα γίνονται αντιληπτοί οι μετρήσεις που καταγράφηκαν σε αυτό το ερώτημα έγιναν από ένα τρέξιμο του κώδικα.

6. Στο σημείο αυτό κληθήκαμε να υπολογίσουμε με διάφορους τρόπους το άθροισμα των συντελεστών του πολυωνύμου Wilkinson και να δούμε ποιος από αυτούς επιστρέφει το σωστό (ίσο με 0).

   Ο κώδικας που χρησιμοποιήσαμε ήταν ο ακόλουθως(ask2_6.m):

```
p=poly(single(1:12));
n=length(p);
s=sum(p)

%a
s_a=0;
for i=1:1:n
    s_a=s_a+p(1,i);
end
s_a

%b
s_b=0;
for i=1:1:n
    s_b=s_b+p(1,n-i+1);
end
s_b

%c
temp=abs(p);
[temp,Seira]=sort(temp,'ascend');
s_c=0;
for i=1:1:n
    s_c=s_c+p(1,Seira(i));
end
s_c

%d
temp=abs(p);
[temp,Seira]=sort(temp,'descend');
s_d=0;
for i=1:1:n
    s_d=s_d+p(1,Seira(i));
end
s_d

%e
j=1;
z=1;
for i=1:1:n
    if p(1,i)<0
        arnitika(1,j)=p(1,i);
        j=j+1;
    else
        thetika(1,z)=p(1,i);
        z=z+1;
    end
end
n_temp=length(arnitika);
[arnitika,Seira]=sort(arnitika,'descend');
s_e=0;
for i=1:1:n_temp
```

```
    s_e=s_e+arnitika(1,i);
end

n_temp=length(thetika);
[thetika,Seira]=sort(thetika,'ascend');
s_temp=0;
for i=1:1:n_temp
    s_temp=s_temp+thetika(1,i);
end
s_e=s_e+s_temp

%f
[temp,Seira]=sort(p,'ascend');
for i=1:1:n-1
    temp=[temp(1,1)+temp(1,2),temp(1,3:n-i+1)];
    [temp,Seira]=sort(temp,'ascend');
end
s_f = temp
```

Τα αποτελέσματα που λάβαμε φαίνονται παρακάτω:

(α′) 192

(β′) 187

(γ′) 256

(δ′) 187

(ε′) 0

(ϛ′) 512

(ζ′) και απο την κλασσική $sum(p) = 192$ (σε μονή ακρίβεια, αλλιώς έβγαινε κανόνικα 0).

Από τα παραπάνω εύκολα μπορούμε να δούμε ότι το σωστό αποτέλεσμα το έδωσε η μέθοδος κατα την οποία προσθέσαμε πρώτα τα θετικά και τα αρνητικά ξεχωριστά (κατα αύξουσα απόλυτη τιμή) και μετά το τελικό άθροισμα. Αυτό συμβαίνει γιατί από την πρόσθεση των θετικών προκύπτει ένα θετικό σφάλμα και από τα αρνητικά ένα αντίστοιχο και όταν τα προσθέτουμε αυτά τα δύο λόγω των αντίθετων προσήμων των σφαλμάτων τους ἀπαλοίφονται᾽ και αυτά (η έννοια της απαλοιφής εδώ χρησημοποιήται με μεταφορική σημασιά καθώς τα σφάλματα είναι σφάκματα και δεν τα γνωριζουμε). Σε όλες της άλλες περιπτώσεις δεν συμβαίνει αυτό και για αυτό έχουμε και αποτελέσματα που απκλύνουν τόσο, λόγω της τόσο μεγάλης συσόρεφσης σφάλματος (λόγω αριθμητικής μονής ακρίβειας).