2η Εργασία στον Επιστημονικό Υπολογισμό

Αλεβιζοπούλου Αφροδίτη Α.Μ:3879
alevizopou@ceid.upatras.gr

14 Δεκεμβρίου 2014

Εισαγωγικά

Παρακάτω περιγράφονται τα χαρακτηριστικά του υπολογιστικού συστήματος το οποίο χρησιμοποιήσαμε για την υλοποίηση της εργαστηριακής άσκησης. Για τον υπολογισμό των χαρακτηριστικών χρησιμοποιήθηκαν δύο ειδικά προγράμματα, το CPU-Z και το PC Wizard.

(i) **Επεξεργαστής:** Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz

Ο εν λόγω επεξεργαστής διαθέτει τέσσερις πυρήνες και η συχνότητα λειτουργίας του είναι 3.30GHz.

Κρυφή Μνήμη: Η κρυφή του μνήμη, μεγέθους 6MB, αποτελείται από τρία επίπεδα, L1, L2, L3, με το πρώτο επίπεδο να χωρίζεται σε κρυφή μνήμη εντολών και κρυφή μνήμη δεδομένων. Συγκεκριμένα:

L1 Data Cache:	32 KBytes, 8-way set associative, 64-byte line size
L1 Instruction Cache:	32 KBytes, 8-way set associative, 64-byte line size
L2 Cache:	256 KBytes, 8-way set associative, 64-byte line size
L3 Cache:	6 MBytes, 12-way set associative, 64-byte line size

Είδος της πολιτικής εγγραφής στην κρυφή μνήμη: Write-Back.

Λειτουργικό σύστημα του υπολογιστή: Windows 7 Professional Service Pack 1 (64-bit).

(ii) Για την υλοποίηση της άσκησης χρησιμοποιήθηκε η έκδοση 8.01 (R2013a) της MATLAB.

(iii) Η διακριτότητα του χρονομετρητή tic/toc, δηλαδή πόσος χρόνος μεσολαβεί μεταξύ του tic και του toc αν δε διεξαχθεί κανένας υπολογισμός, εκτιμάται στα $8.2935e-08$ sec., δηλ. 8.2935×10^{-8} sec.

(iv) Το αποτέλεσμα της συνάρτησης bench για τη διάσπαση μητρώων LU είναι: 0.0470 sec.

Δε χρησιμοποιούνται εντολές FMA από το εν λόγω υπολογιστικό σύστημα.

1 Πράξεις με Πολυώνυμα

(α΄) Στην παράγραφο που ακολουθεί θα εξηγήσουμε για ποιο λόγο χρησιμοποιούνται οι συναρτήσεις poly και polyval, καθώς και πώς ακριβώς λειτουργούν, έχοντας μελετήσει τους κώδικές τους από το Matlab.

poly: Η poly δέχεται ως είσοδο είτε ένα $n \times n$ μητρώο, είτε ένα διάνυσμα. Όταν η είσοδος είναι ένα μητρώο A, δηλαδή $p = poly(A)$, επιστρέφεται ένα διάνυσμα-στήλη μήκους n+1, τα στοιχεία του οποίου είναι οι συντελεστές του χαρακτηριστικού πολυωνύμου $det(\lambda I - A)$. Οι συντελεστές είναι διατεταγμένοι κατά φθίνουσα δύναμη. Όταν η είσοδος είναι ένα διάνυσμα r, δηλαδή $p = poly(r)$, επιστρέφεται ένα διάνυσμα-στήλη τα στοιχεία του οποίου είναι οι συντελεστές του πολυωνύμου, του οποίου οι ρίζες είναι τα στοιχεία του διανύσματος r. Ουσιαστικά, δίνοντας ως είσοδο τις ρίζες ενός πολυωνύμου, επιστρέφονται οι συντελεστές αυτού. Πιο συγκεκριμένα, η συνάρτηση poly ελέγχει αρχικά τα χαρακτηριστικά του πολυωνύμου. Απομακρύνει τις τιμές που απειρίζονται, και χρησιμοποιεί φόρμουλα αναδρομής. Τα αποτελέσματα που δίνει είναι «πραγματικά», αν οι ρίζες είναι συζυγείς μιγαδικές.

polyval: Η polyval υπολογίζει ένα πολυώνυμο. Η $Y = polyval(P, X)$ επιστρέφει την αξία του πολυωνύμου P στο X. Το P είναι ένα διάνυσμα μήκους n+1, του οποίου τα στοιχεία είναι οι συντελεστές του πολυωνύμου σε φθίνουσα σειρά. Αν το X είναι μητρώο ή διάνυσμα, τότε το πολυώνυμο υπολογίζεται σε όλα τα σημεία στο X. Πιο συγκεκριμένα, αρχικά ελέγχει το ότι η είσοδος είναι διάνυσμα. Επίσης, χρησιμοποιεί τη μέθοδο του Horner για τη γενική περίπτωση που το X είναι ένας πίνακας. Χρησιμοποιεί παραμέτρους που έχουν βγει ως αποτέλεσμα της polyfit. Κατασκευάζει μητρώο Vandermonder για το καινούριο X.

(β') Για το δοθέν πολυώνυμο υλοποιήσαμε script MATLAB (αρχείο ex1.m) με το οποίο υπολογίζουμε τους συντελεστές του σε μορφή δυναμοσειράς μέσω της poly, τις τιμές του κάθε πολυωνύμου στα σημεία $x = 1, x = n$, καθώς και τις ρίζες που επιστρέφει η συνάρτηση MATLAB roots. Στη συνέχεια συγκρίνουμε τις ρίζες που επιστρέφει η roots με τις πραγματικές ρίζες του πολυωνύμου. Τέλος, παρουσιάζουμε τα αποτελέσματα και τις πραγματικές ρίζες σε κοινή γραφική παράσταση και σχολιάζουμε τι παρατηρούμε.

(i)

```
format long

k = [15:2:25];
values = zeros(6,2);

for i = 1:length(k)

    % Ypologismos suntelestwn apo tis gnwstes rizes
    coefficients = poly(1:k(i));
    solutions = [1:k(i)];
    disp(strcat('Coefficients for k= ', num2str(k(i))));
    coefficients'
    disp(strcat('Roots for k= ', num2str(k(i))));
    disp('...........');
    r = roots(coefficients)

    % Ypologismos timwn tou ka8e polywnumoy sta x = 1, x = n
    values(i, [1 2]) = polyval(coefficients,[1 k(i)]);

    figure
    subplot(2,1,1), plot([1:k(i)], 'k*', 'Markersize', 5)
    title('theoritical solutions')
    xlabel('solution number')
    ylabel('values')
    subplot(2,1,2), plot(real(r), imag(r), 'r*')
    title('solutions computed by roots')
    xlabel('real part')
    ylabel('imaginary part')

end
```

(ii) Παραθέτουμε τις τιμές του κάθε πολυωνύμου στα σημεία $x = 1, x = n$, όπως υπολογίστηκαν από την εκτέλεση του παραπάνω script:

Πολυώνυμο/Σημείο	$x = 1$	$x = n$
p_{15}	0	0
p_{17}	0	0
p_{19}	0	33362176
p_{21}	-65536	2.294070598041600e+13
p_{23}	8388608	3.644865039019541e+17
p_{25}	1.503238553600000e+10	-1.712305311450888e+22

Πίνακας 1: Τιμές πολυωνύμου

Παρατηρούμε πως ενώ δίνουμε στο εκάστοτε πολυώνυμο ως είσοδο τις ρίζες του $x = 1, x = n$, δεν παίρνουμε μηδενικό αποτέλεσμα για όλες τις περιπτώσεις.

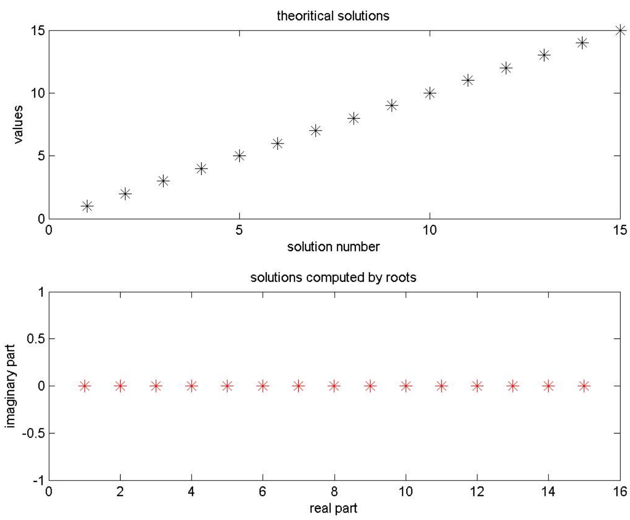
(iii) Παίρνουμε τα παρακάτω αποτελέσματα:

• $n = 15$

Η roots επιστρέφει το διάνυσμα:

$$\begin{bmatrix} 15.000000268968613 \\ 13.999997689630304 \\ 13.000008647420207 \\ 11.999981276991626 \\ 11.000026161192281 \\ 9.999975134726144 \\ 9.000016483519099 \\ 7.999992319684219 \\ 7.000002501047653 \\ 5.999999441413502 \\ 5.000000082683621 \\ 3.999999992322418 \\ 3.000000000411776 \\ 1.999999999988686 \\ 1.000000000000111 \end{bmatrix}$$

Οι τιμές αυτές είναι πολύ κοντά στις πραγματικές ρίζες του πολυωνύμου:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.$

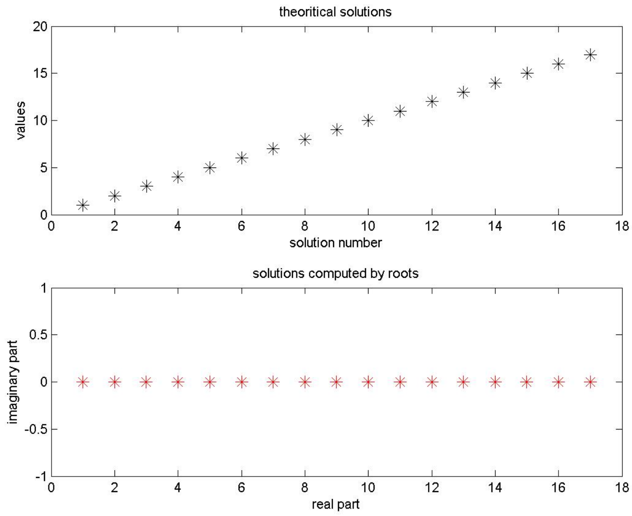


• $n = 17$

Η roots επιστρέφει το διάνυσμα:

$$\begin{bmatrix} 16.999999407654144 \\ 16.000011076558543 \\ 14.999934400092544 \\ 14.000204047694421 \\ 12.999603130249795 \\ 12.000523707211315 \\ 10.999509103972596 \\ 10.000335288275501 \\ 8.999831184160357 \\ 8.000062848716047 \\ 6.999982890121404 \\ 6.000003314774610 \\ 4.999999567516789 \\ 4.000000034362123 \\ 2.999999998620772 \\ 2.000000000019045 \\ 0.999999999999960 \end{bmatrix}$$

Οι πραγματικές ρίζες του πολυωνύμου είναι:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17.$

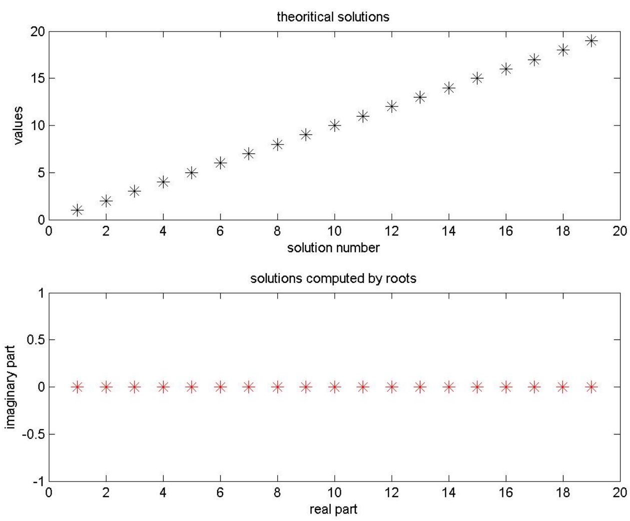


• $n = 19$

Η roots επιστρέφει το διάνυσμα:

$$\begin{bmatrix} 19.000027021273119 \\ 17.999834075241566 \\ 17.000328251661717 \\ 16.000180233276151 \\ 14.997759270007226 \\ 14.005378955541451 \\ 12.992379126571681 \\ 12.007294475160844 \\ 10.995047480109756 \\ 10.002458915286169 \\ 8.999124012863504 \\ 8.000223182949775 \\ 6.999960725408893 \\ 6.000004606184322 \\ 4.999999651955219 \\ 4.000000017019094 \\ 2.999999999486902 \\ 2.000000000002510 \\ 1.000000000000145 \end{bmatrix}$$

Οι πραγματικές ρίζες του πολυωνύμου είναι:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19.$

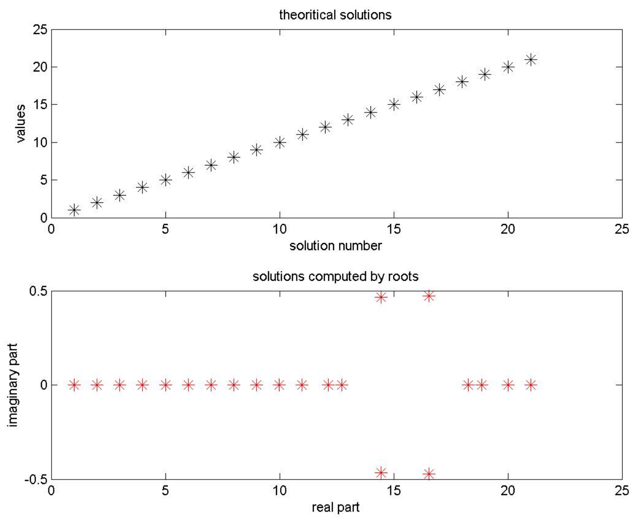


• $n = 21$

Η roots επιστρέφει το διάνυσμα:

$$\begin{bmatrix} 20.997800308398620 \\ 20.021729406742342 \\ 18.870779624438509 \\ 18.271882618961232 \\ 16.554835386399514 + 0.471136226424383i \\ 16.554835386399514 - 0.471136226424383i \\ 14.444280044942694 + 0.467320726095183i \\ 14.444280044942694 - 0.467320726095183i \\ 12.732342978660608 \\ 12.124699576746448 \\ 10.983391254127591 \\ 9.997185695247495 \\ 9.002595902502229 \\ 7.999256283842197 \\ 7.000113898831021 \\ 5.999991505594797 \\ 5.000000050308115 \\ 4.000000034765502 \\ 2.999999998126284 \\ 2.000000000022433 \\ 1.000000000000056 \end{bmatrix}$$

Οι πραγματικές ρίζες του πολυωνύμου είναι:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21.$

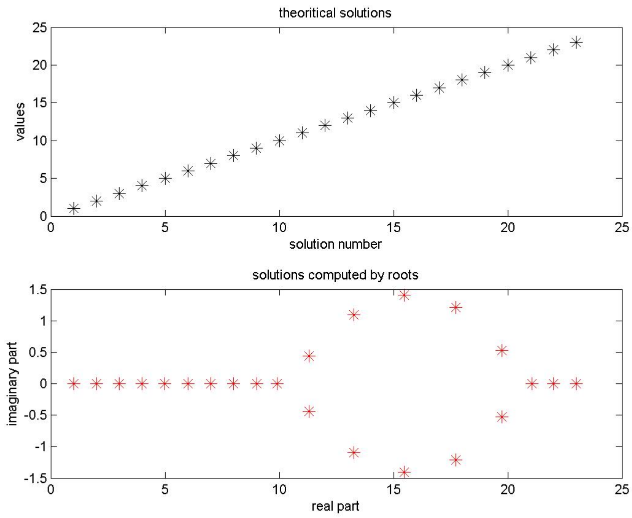


• $n = 23$

Η roots επιστρέφει το διάνυσμα:

$$\begin{bmatrix} 22.999457536723781 \\ 21.998042447037392 \\ 21.064524037339204 \\ 19.750580271489248 + 0.523720291638804i \\ 19.750580271489248 - 0.523720291638804i \\ 17.729097022357909 + 1.215904497216104i \\ 17.729097022357909 - 1.215904497216104i \\ 15.461147897523340 + 1.404320150598003i \\ 15.461147897523340 - 1.404320150598003i \\ 13.258884719596038 + 1.098567605301376i \\ 13.258884719596038 - 1.098567605301376i \\ 11.303458344708041 + 0.444511947002807i \\ 11.303458344708041 - 0.444511947002807i \\ 9.918975030614840 \\ 9.013691232835587 \\ 7.998975659473170 \\ 6.999986806042221 \\ 6.000011826881678 \\ 4.999998871227238 \\ 4.000000040524066 \\ 2.999999999969369 \\ 1.999999999982311 \\ 1.000000000000058 \end{bmatrix}$$

Οι πραγματικές ρίζες του πολυωνύμου είναι:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23.$

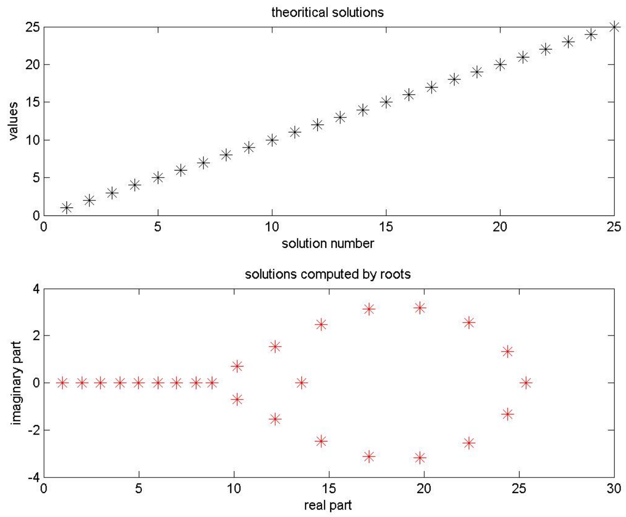


• $n = 25$

Η roots επιστρέφει το διάνυσμα:

$$
\begin{bmatrix}
25.369470471347050 \\
24.413163143270872 + 1.323138924749880i \\
24.413163143270872 - 1.323138924749880i \\
22.360817449056771 + 2.560238212460677i \\
22.360817449056771 - 2.560238212460677i \\
19.783028762770847 + 3.183285324361284i \\
19.783028762770847 - 3.183285324361284i \\
17.106777800468446 + 3.115396005074317i \\
17.106777800468446 - 3.115396005074317i \\
14.584168402863160 + 2.466558423920374i \\
14.584168402863160 - 2.466558423920374i \\
13.558726511676019 \\
12.174264314070756 + 1.532004057855084i \\
12.174264314070756 - 1.532004057855084i \\
10.179546621867281 + 0.709672134182649i \\
10.179546621867281 - 0.709672134182649i \\
8.839979049685768 \\
8.030978691814989 \\
6.997122496612382 \\
6.000198198832249 \\
4.999991383779600 \\
4.000000209967163 \\
2.999999997524334 \\
2.000000000024500 \\
0.999999999999802
\end{bmatrix}
$$

Οι πραγματικές ρίζες του πολυωνύμου είναι:
$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25.$



Παρατηρούμε ότι όσο αυξάνει η δύναμη στο πολυώνυμο, οι ρίζες μας χαλάνε, δηλαδή εμφανίζονται και μιγαδικές ρίζες. Για να εξηγήσουμε αυτό το γεγονός, αρκεί να παρατηρήσουμε τους συντελεστές που επιστρέφει η συνάρτηση poly. Παρατηρούμε ότι υπάρχουν συντελεστές πάρα πολύ μεγάλοι της τάξης του 10^{20} και συντελεστές πάρα πολύ μικροί. Σύμφωνα με τα παραπάνω, η συμπεριφορά του πολυωνύμου είναι όντως δόλια, όπως την είχε χαρακτηρίσει ο James Wilkinson, γιατί η διαχείρισή τους κρύβει παγίδες.

2 Αθροίσματα - Μονάδα Στρογγύλευσης

(α΄) Παρακάτω υλοποιήσαμε τέσσερις συναρτήσεις MATLAB οι οποίες υπολογίζουν το άθροισμα για είσοδο x ενός συνόλου αριθμών δοθέντων μονής ακρίβειας με τις παρακάτω μεθόδους:

(i) Με σειρά από αριστερά προς τα δεξιά όπως δίνεται η είσοδος (αρχείο sum1.m):

```
function [xsum] = sum1(x)

xsum = 0;

for i=1:length(x);
    xsum = xsum + x(i);
end

end
```

(ii) Η είσοδος ταξινομείται πρώτα σε αύξουσα σειρά (αρχείο sum2.m):

```
function [xsum] = sum2(a)

x = sort(a);
xsum = 0;

for i=1:length(a);
    xsum = xsum + x(i);
end

end
```

(iii) Η είσοδος ταξινομείται πρώτα σε αύξουσα σειρά. Στη συνέχεια υπολογίζεται το $x_1 + x_2$ και το αποτέλεσμα ενθέτεται στη λίστα έτσι ώστε να διατηρείται η διάταξη $x_3, ..., x_n$. Η διαδικασία επαναλαμβάνεται μέχρι να απομείνει μόνο ένα νούμερο που θα είναι και το αποτέλεσμά μας (αρχείο sum3.m):

```
function [x] = sum3(x)

x = sort(x);
i = 1;

for k = 1:(length(x)-1);
    x(i+1) = x(i+1) + x(i);
    x(i) = [];
    x = sort(x);
end

end
```

(iv) Χρησιμοποιώντας τον αλγόριθμο άθροισης Pichat & Neumaier's (αρχείο pichat.m):

```
function [s] = pichat(x)

s = x(1);
e = 0;
n = length(x);

for i = 2:n
    if abs(s) >= abs(x(i))
        [s,ei] = fast2sum(s,x(i));
    else
        [s,ei] = fast2sum(x(i),s);
    end
        e = e + ei;
end

s = s+e;

end
```

(β') Υλοποιούμε script MATLAB (αρχείο ex2.m) με το οποίο δοκιμάζουμε τις διαφορετικές μεθόδους άθροισης, τις οποίες υλοποιήσαμε ως συναρτήσεις στο προηγούμενο ερώτημα, για τις ζητούμενες εισόδους:

- x_i είναι ο i-οστός όρος της σειράς Taylor για το e^{-2pi} για $n = 64$ όρους
- $x_1 = x_2 = ... = 1.0, x_{2048} = x_{2049} = 1.0e - 18, x_{2050} = x_{2051} = ... = x_{4096} = -1.0$
- x_i ισαπέχουν στο διάστημα [1,2] για $n = 4096$
- $x_i = 1/i^2$, για $n = 4096$

```
%Erwthma 2bi
disp('----------2bi----------');

for n = 1:64
    x(n) = -2*pi^(n-1)/factorial(n-1);
end

% double
d1 = sum1(x);
d2 = sum2(x);
d3 = sum3(x);
d4 = pichat(x);

% single
x_single = single(x);
s1 = sum1(x_single);
s2 = sum2(x_single);
s3 = sum3(x_single);
s4 = pichat(x_single);

diaforal = norm(s1-d1,inf)
```

```
diafora2 = norm(s2-d2,inf)
diafora3 = norm(s3-d3,inf)
diafora4 = norm(s4-d4,inf)

%Erwthma 2bii
disp('----------2bii----------');

x = zeros(4096,1);
x(1:2047) = 1.0;
x(2048:2049) = 1.0e-18;
x(2050:4096) = -1.0;

% double
d1 = sum1(x);
d2 = sum2(x);
d3 = sum3(x);
d4 = pichat(x);

% single
x_single = single(x);
s1 = sum1(x_single);
s2 = sum2(x_single);
s3 = sum3(x_single);
s4 = pichat(x_single);

diafora1 = norm(s1-d1,inf)
diafora2 = norm(s2-d2,inf)
diafora3 = norm(s3-d3,inf)
diafora4 = norm(s4-d4,inf)

%Erwthma 2biii
disp('----------2biii----------');

x = linspace(1,2,4096);

% double
d1 = sum1(x);
d2 = sum2(x);
d3 = sum3(x);
d4 = pichat(x);

% single
x_single = single(x);
s1 = sum1(x_single);
s2 = sum2(x_single);
s3 = sum3(x_single);
s4 = pichat(x_single);

diafora1 = norm(s1-d1,inf)
diafora2 = norm(s2-d2,inf)
diafora3 = norm(s3-d3,inf)
```

```
diafora4 = norm(s4-d4,inf)

%Erwthma 2biv
disp('----------2biv----------');

for i = 1:4096
    x(i) = 1/(i^2);
end

% double
d1 = sum1(x);
d2 = sum2(x);
d3 = sum3(x);
d4 = pichat(x);

% single
x_single = single(x);
s1 = sum1(x_single);
s2 = sum2(x_single);
s3 = sum3(x_single);
s4 = pichat(x_single);

diafora1 = norm(s1-d1,inf)
diafora2 = norm(s2-d2,inf)
diafora3 = norm(s3-d3,inf)
diafora4 = norm(s4-d4,inf)
```

(γ') Θέλοντας να σχολιάσουμε τα αποτελέσματά μας για τους διαφορετικούς τύπους εισόδων και μεθόδων, παρατηρούμε το αποτέλεσμα της σύγκρισης για όλους τους τύπους εισόδων όταν η εισόδος αναπαρίσταται σε μονή και διπλή ακρίβεια για κάθε μέθοδο. Συμπεραίνουμε πως ο πρώτος τρόπος άθροισης (συνάρτηση sum1 -με σειρά από αριστερά προς τα δεξιά όπως δίνεται η είσοδος-) συμπεριφέρεται καλύτερα για τον δεύτερο τύπο εισόδου.

Ο δεύτερος τρόπος άθροισης (συνάρτηση sum2 -η είσοδος ταξινομείται πρώτα σε αύξουσα σειρά-) συμπεριφέρεται καλύτερα για τον δεύτερο τύπο εισόδου, ομοίως με τη συνάρτηση sum1.

Ο τρίτος τρόπος άθροισης (συνάρτηση sum3 -η είσοδος ταξινομείται πρώτα σε αύξουσα σειρά και το κάθε αποτέλεσμα ενθέτεται στη λίστα-) συμπεριφέρεται καλύτερα για τον δεύτερο και τον τρίτο τύπο εισόδου.

Ο τέταρτος τρόποις άθροισης (συνάρτηση pichat) συμπεριφέρεται καλύτερα για τον τρίτο τύπο εισόδου, όπου τα x_i ισαπέχουν στο διάστημα [1,2] για $n = 4096$.

3 Γραμμικά Συστήματα

Μέρος Α

(α') Παραθέτουμε το ζητούμενο script MATLAB (αρχείο ex3a.m), το οποίο υπολογίζει το δείκτη κατάστασης του εκάστοτε μητρώου ως προς τη νόρμα μεγίστου, το εμπρός σχετικό σφάλμα και το πίσω σφάλμα:

```
format long

n = 512;
P = zeros(6,3);
%P(:,1) ---> Deiktis katastasis
%P(:,2) ---> Empros sxetiko sfalma
%P(:,3) ---> Pisw sfalma

% case 1
A = randn(n);
P(1,1) = cond(A,inf);

% case 2
B = tril(A);
P(2,1) = cond(B,inf);

% case 3
[L,U] = lu(A);
P(3,1) = cond(U,inf);

% case 4
UC = U;
UC(n,:) = [];
UC(:,n) = [];
C = gfpp(UC);
P(4,1) = cond(C,inf);

% case 5i
d = linspace(-1,1,n);
D = vander(d);
P(5,1) = cond(D,inf);

% case 5ii
e = cos([1:n]*pi/(n+1));
E = vander(e);
P(6,1) = cond(E,inf);

%pragmatiki lusi
x = ones(n,1);
b = A*x;

%lusi pou vrisko

% case 1
```

```
x1 = mldivide(A,b);
P(1,2) = norm(x1-x,inf)/norm(x,inf);
P(1,3) = norm((A*x1-b),inf)/(norm(A,inf)*norm(x1,inf)+norm(b,inf));

% case 2
b = B*x;
x2 = mldivide(B,b);
P(2,2) = norm(x2-x,inf)/norm(x,inf);
P(2,3) = norm((B*x2-b),inf)/(norm(B,inf)*norm(x2,inf)+norm(b,inf));

% case 3
b = U*x;
x3 = mldivide(U,b);
P(3,2) = norm(x3-x,inf)/norm(x,inf);
P(3,3) = norm((U*x3-b),inf)/(norm(U,inf)*norm(x3,inf)+norm(b,inf));

b = C*x;
x4 = mldivide(C,b);
P(4,2) = norm(x4-x,inf)/norm(x,inf);
P(4,3) = norm((C*x4-b),inf)/(norm(C,inf)*norm(x4,inf)+norm(b,inf));

b = D*x;
x5 = mldivide(D,b);
P(5,2) = norm(x5-x,inf)/norm(x,inf);
P(5,3) = norm((D*x5-b),inf)/(norm(D,inf)*norm(x5,inf)+norm(b,inf));

b = E*x;
x6 = mldivide(E,b);
P(6,2) = norm(x6-x,inf)/norm(x,inf);
P(6,3) = norm((E*x6-b),inf)/(norm(E,inf)*norm(x6,inf)+norm(b,inf));

% Sthlh me deikti katastasis
p1 = P(:,1)
% Sthlh me empros sxetiko sfalma
p2 = P(:,2)
% Sthlh me pisw sfalma
p3 = P(:,3)

P(1,2)<P(1,1)*P(1,3)
P(2,2)<P(2,1)*P(2,3)
P(3,2)<P(3,1)*P(3,3)
P(4,2)<P(4,1)*P(4,3)
P(5,2)<P(5,1)*P(5,3)
P(6,2)<P(6,1)*P(6,3)
```

(b) Παραθέτουμε τον πίνακα με τις τιμές που υπολογίσαμε για το δείκτη κατάστασης, το εμπρός σχετικό σφάλμα και το πίσω σφάλμα:

Μητρώο	**Δείκτης Κατάστασης**	**Εμπρός σχετικό σφάλμα**	**Πίσω σφάλμα**
randn	6.6211e+05	2.9188e-13	1.9343e-15
tril	7.8624e+19	3.7564e+132	4.4869e-26
lu	3.5820e+04	1.2068e-13	1.4063e-16
gfpp	2.5681e+05	4.4142e-13	8.8746e-17
Vandermonde (i)	4.8990e+222	3.5278e+203	9.6677e-20
Vandermonde (ii)	1.9699e+156	2.2121e+137	1.4076e-18

Τα αποτελέσματα για τα φράγματα συμφωνούν με τις θεωρητικές προβλέψεις, εκτός από την περίπτωση των κάτω τριγωνικών μητρώων, όπου οι τιμές που έχουμε βρει για το δείκτη κατάστασης, το εμπρός σχετικό σφάλμα και το πίσω σφάλμα δεν επαληθεύουν την ανισότητα, όπως μας υποδεικνύει η θεωρία. Αυτό οφείλεται στην πολύ μεγάλη τιμή του εμπρός σχετικού σφάλματος.

προς τα εμπρός σφάλμα $<$ δείκτης κατάστασης πρβλ. $\times$ πίσω σφάλμα

Επίσης, παρατηρούμε πως ο δείκτης κατάστασης και το εμπρός σχετικό σφάλμα για την περίπτωση των μητρώων Vandermonde που παράγονται τόσο από ισαπέχουσες τιμές μεταξύ [-1,1], αλλά και από τιμές Chebyshev (περίπτωση $5i, 5ii$) λαμβάνουν πολύ μεγάλη τιμή. Ο δείκτης κατάστασης ενός μητρώου δείχνει πόσο επηρεάζεται η λύση ενός γραμμικού συστήματος από λάθη στα δεδομένα εισόδου. Επίσης, αποτελεί ένδειξη για την ακρίβεια (accuracy) των αποτελεσμάτων που προκύπτουν από την αντιστροφή ενός μητρώου. Η εντολή cond επιστρέφει τη νόρμα 2 του δείκτη κατάστασης ενός μητρώου (το λόγο του μεγαλύτερου στοιχείου του μητρώου προς το μικρότερο). Μεγάλος δείκτης κατάστασης δείχνει ένα σχεδόν ιδιάζον μητρώο.

Μέρος Β

(α') Στο μέρος αυτό συγκρίνουμε σφάλματα αριθμητικής κινητής υποδιαστολής που προκύπτουν για την πράξη του κλασσικού πολλαπλασιασμού μητρώων και τον block πολλαπλασιασμό μητρώων χρησιμοποιώντας τον αλγόριθμο Strassen. Για τους ζητούμενους τύπους μητρώων υλοποιήσαμε το παρακάτω script MATLAB (αρχείο ex3b.m)το οποίο συγκρίνει το εμπρός σχετικό σφάλμα για τις δύο διαφορετικές υλοποιήσεις του πολλαπλασιασμού μητρώων. Για να υπολογίσουμε το εμπρός σχετικό σφάλμα εκτελούμε τις πράξεις σε MATLAB με μονή ακρίβεια, ενώ θεωρούμε ως ¨άπειρη¨ ακρίβεια την πράξη που υλοποιείται με τη συνάρτηση MATLAB mtimes σε διπλή ακρίβεια.

```
n = 512;

disp('----------(a.i)----------');

A = randn(1024);
B = randn(1024);

% double mtimes
C = mtimes(A,B);

% single mtimes
CS = mtimes(single(A),single(B));
S1 = norm(CS-C,inf)/norm(C,inf)

% single Strassen
```

```
DS = strassen(single(A),single(B));
S2 = norm(DS-C,inf)/norm(C,inf)

disp('----------(a.ii)----------');

a = randn(1024,1);
b = randn(1024,1);
A = vander(a);
B = vander(b);

% double mtimes
C = mtimes(A,B);

% single mtimes
CS = mtimes(single(A),single(B));
S3 = norm(CS-C,inf)/norm(C,inf)

% single Strassen
DS = strassen(single(A),single(B));
S4 = norm(DS-C,inf)/norm(C,inf)

disp('----------(a.iii)----------');

I = eye(n);
M = 10^7;
II = [I zeros(n); zeros(n) I];
A1 = randn(n);
A2 = randn(n);
A3 = randn(n);
A4 = randn(n);
AA = [M*A1 A2; A3 A4];

% double mtimes
C = mtimes(II,AA);

% single mtimes
CS = mtimes(single(II),single(AA));
S5 = norm(CS-C,inf)/norm(C,inf)

% single Strassen
DS = strassen(single(II),single(AA));
S6 = norm(DS-C,inf)/norm(C,inf)
```

(b) Θέλοντας να σχολιάσουμε τα αποτελέσματα που επιστρέφονται από την εκτέλεση του παραπάνω script, παρατηρούμε πως εντοπίζονται διαφορές στο εμπρός σχετικό σφάλμα μεταξύ των δύο υλοποιήσεων. Συγκεκριμένα, για τις περιπτώσεις των μητρώων (i) και (iii), η υλοποίηση με χρήση του αλγορίθμου Strassen εμφανίζει μεγαλύτερο εμπρός σχετικό σφάλμα, ενώ για την περίπτωση των μητρώων Vandermonde (περίπτωση (ii)), επιστρέφεται NaN ως αποτέλεσμα της σύγκρισης των εμπρός σχετικών σφαλμάτων μεταξύ των δύο υλοποιήσεων.