Επιστημονικός Υπολογισμός Ι
2η Εργαστηριακή Άσκηση

Παπαρροδοπούλου Αναστασία ΑΜ 3873
11/12/13

Εισαγωγή – Χαρακτηριστικά υπολογιστικού συστήματος:

Για τον υπολογισμό των παρακάτω χαρακτηριστικών του υπολογιστικού συστήματος στο οποίο υλοποιήθηκε η εργαστηριακή άσκηση, χρησιμοποιήθηκαν δύο ειδικά προγράμματα(τα οποία κατεβάστηκαν από τη διεύθυνση http://www.cpuid.com/), το CPU-Z και το PC Wizard.

Τύπος και συχνότητα λειτουργίας επεξεργαστή :

Intel Mobile Core 2Duo P8600 @ 2.40GHz

Μέγεθος και αριθμός πιπέδων κρυφής μνήμης :

L1 D-Cache : Size 32 Kbytes x2, Descriptor 8-way set associative, 64-byte line size
L1 I-Cache: Size 32 Kbytes x2, Descriptor 8-way set associative, 64-byte line size
L2 Cache : Size 3072 Kbytes, Descriptor 12-way set associative, 64-byte line size

Το **είδος της πολιτικής εγγραφής** στην κρυφή μνήμη είναι Write-Back.

Το λειτουργικό σύστημα του μηχανήματος είναι Windows 7 Professional Service Pack 1 (64-bit).

Επίσης η **έκδοση Matlab** που χρησιμοποιήθηκε είναι η R2012b(64-bit).

Ερώτημα 1 – Χαρακτηριστικά Αριθμητικής στο MATLAB:

(i) Μετά από την εκτέλεση των εντολών του Matlab, eps, realmax και realmin, πήρα τα αποτελέσματα που καταγράφονται στον ακόλουθο πίνακα, στον οποίο έχω συμπεριλάβει και μία περιγραφή του τι αντιπροσωπεύουν αυτά τα αποτελέσματα.

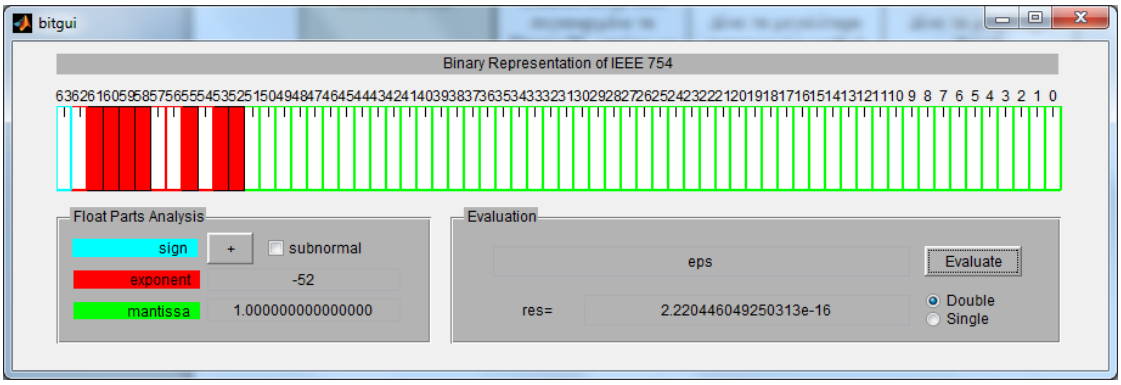
	eps	realmax	realmin
μονή(single)ακρίβεια	1.1921×10^{-7}	3.4028×10^{38}	1.1755×10^{-38}
διπλή(double)ακρίβεια	2.2204×10^{-16}	1.7977×10^{308}	2.2251×10^{-308}
Περιγραφή αποτελεσμάτων	Δίνει την απόσταση μεταξύ αριθμών κινητής υποδιαστολής. Πιο συγκεκριμένα το D=eps(X), ισούται με την απόσταση από το X μέχρι τον επόμενο μεγαλύτερο αναπαραστήσιμο αριθμό κινητής υποδιαστολής(είτε μονής, είτε διπλής ακρίβειας).	Δίνει το μεγαλύτερο πεπερασμένο αριθμό κινητής υποδιαστολής(είτε μονής, είτε διπλής ακρίβειας).	Δίνει το μικρότερο θετικό κανονικοποιημένο αριθμό κινητής υποδιαστολής(είτε μονής, είτε διπλής ακρίβειας).

(ii) Μπορώ να κάνω αρχικά την παρατήρηση ότι στον κώδικα που καλούμαι να τρέξω, συμπεριλαμβάνονται οι εντολές eps, realmin και realmax, και οι τρεις χωρίς όρισμα, κάτι το οποίο σημαίνει σύμφωνα με το Matlab ότι έχουμε διπλή ακρίβεια.

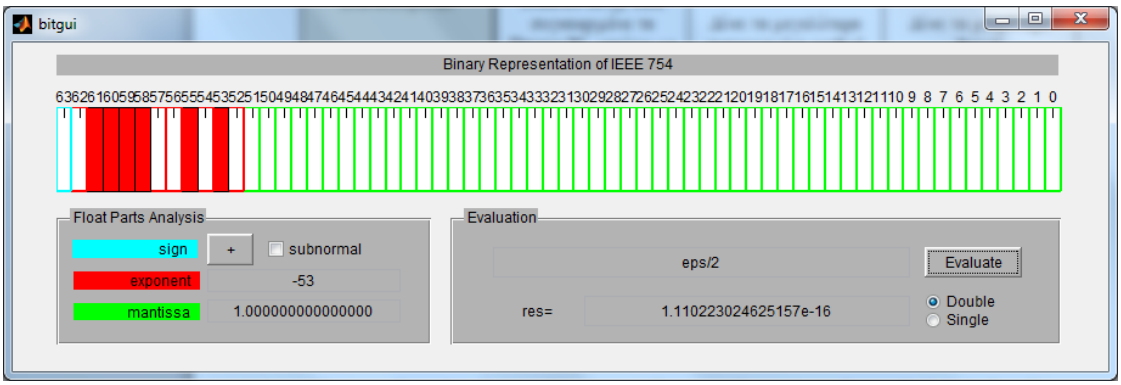
Με την εκτέλεση λοιπόν του κώδικα της εκφώνησης, παίρνω ως αποτέλεσμα 2, 3, 4. Από αυτές τις εκτυπώσεις καταλαβαίνουμε πως έχουν ικανοποιηθεί οι συνθήκες των τριών τελευταίων if-δομών επιλογής(1+eps > 1, 1+realmin == 1, eps + realmax == realmax) ενώ της πρώτης(eps/2 ==0), όχι.

Στη συνέχεια θα προσπαθήσω να δώσω μία όσο το δυνατόν πιο πλήρη αιτιολόγηση γι αυτά τα αποτελέσματα, εξετάζοντας την κάθε περίπτωση ξεχωριστά, μέσω του εργαλείου bitgui, το οποίο χρησιμοποιεί το πρότυπο IEEE 754 για την αναπαράσταση αριθμών κινητής υποδιαστολής με τη χρήση 64-bit. Φαίνεται λοιπόν παρακάτω ξεκάθαρα, στο interface του εργαλείου, το πρόσημο, ο εκθέτης και η ουρά του εκάστοτε αριθμού, καθώς και τα 64 bit.

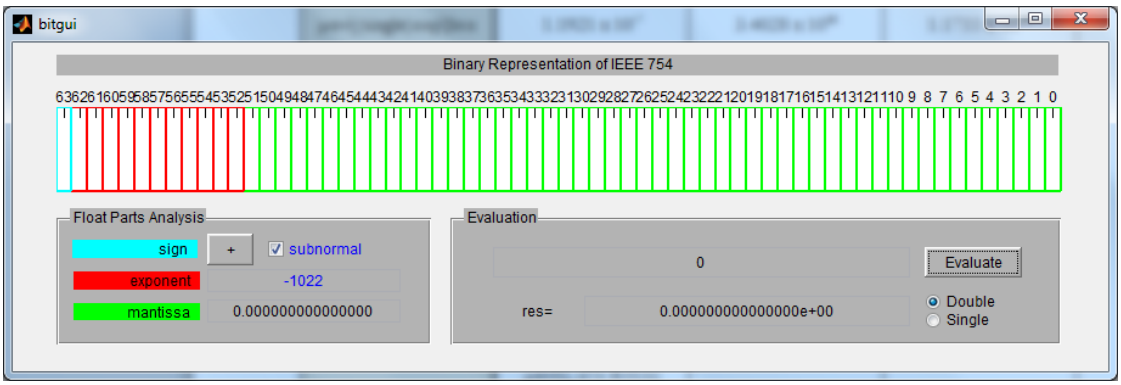
Για τον έλεγχο της πρώτης συνθήκης λοιπόν(**eps/2 ==0**), εξετάζω αρχικά την αναπαράσταση του eps, του οποίου η τιμή, όπως έχουμε καταγράψει και στον παραπάνω πίνακα, είναι 2.2204 x 10^{-16}, και φαίνεται και κάτω δεξιά στο bitgui. Επίσης στο εργαλείο βλέπω πως ο αριθμός έχει θετικό πρόσημο, εκθέτη ίσο με -52 και στη mantissa του μηδενικά.



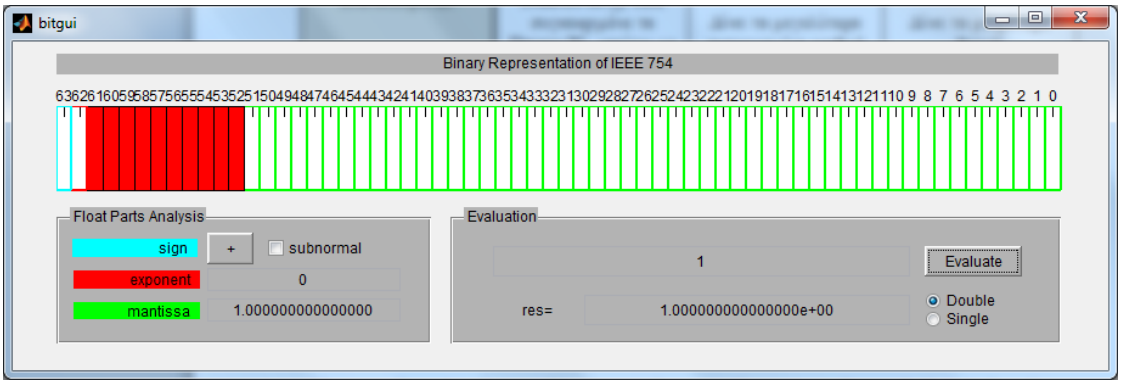
Στην αναπαράσταση τώρα του eps/2 παρατηρώ μία αλλαγή στο λιγότερο σημαντικό ψηφίο του εκθέτη, κάτι το οποίο είναι και λογικό.



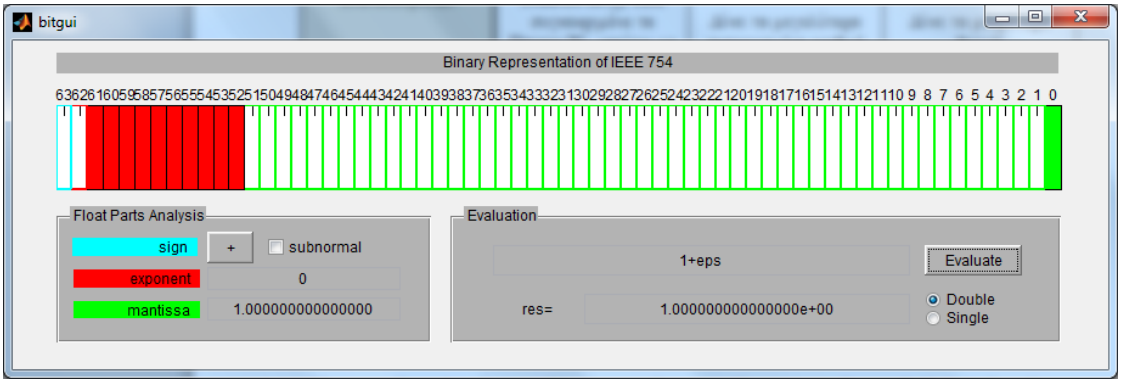
Η αναπαράσταση τώρα του 0, είναι υποκανονικοποιημένη και όλα τα bit ισούνται με 0 επίσης. Οπότε και σώστα δεν έγινε η εκτύπωση του 1 από τον κώδικα, που θα ικανοποιούσε τη συνθήκη της παραπάνω ισότητας.



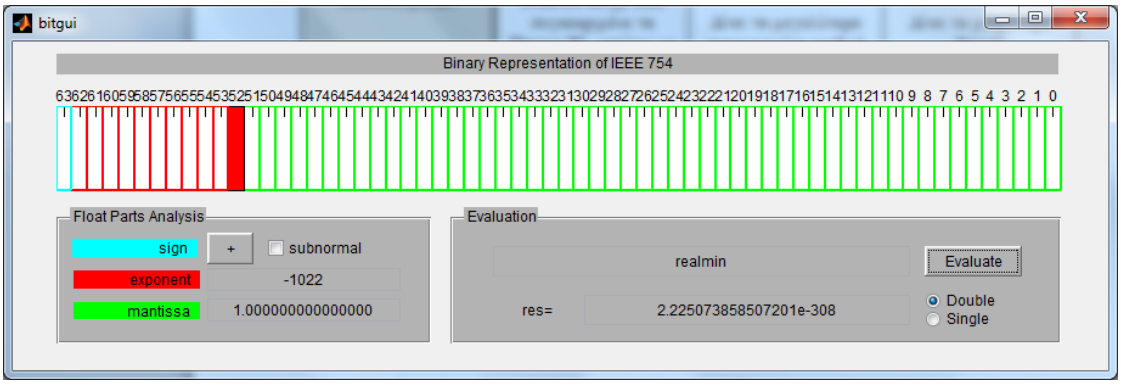
Στη συνέχεια, για τον έλεγχο της δεύτερης συνθήκης (**1+eps >1**), εξετάζω αρχικά την αναπαράσταση του 1. Όλα τα bit της mantissa είναι ίσα με μηδέν, ενώ τα bit του εκθέτη ίσα με 1, πέρα από το πιο σημαντικό που είναι ίσο με μηδέν.



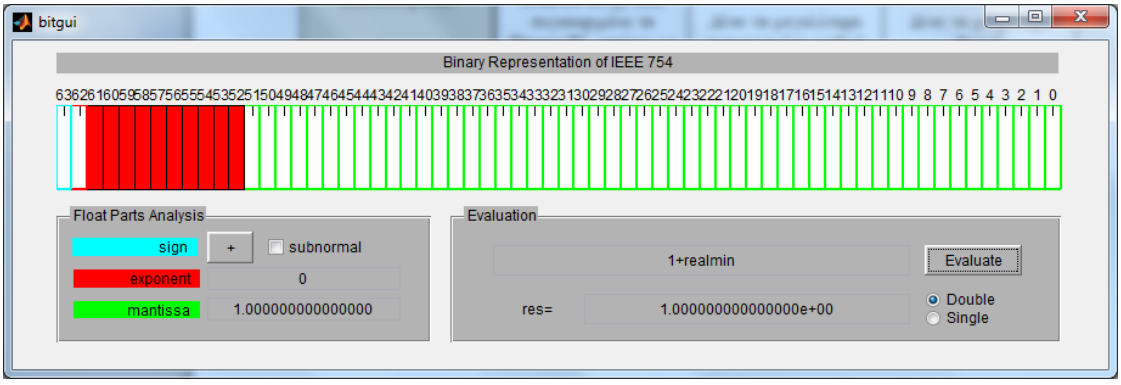
Αν λοιπόν προσθέσουμε το eps(το οποίο έχω παραθέσει παραπάνω), παρατηρώ πως ο αριθμός αλλάζει, μιας και το λιγότερο σημαντικό ψηφίο της mantissa από μηδέν γίνεται ένα. Αυτός ο αριθμός τώρα είναι αμέσως μεγαλύτερος από το 1, οπότε όντως η ανισότητα της συνθήκης ισχύει, γι αυτό έχουμε και την εκτύπωση του 2.



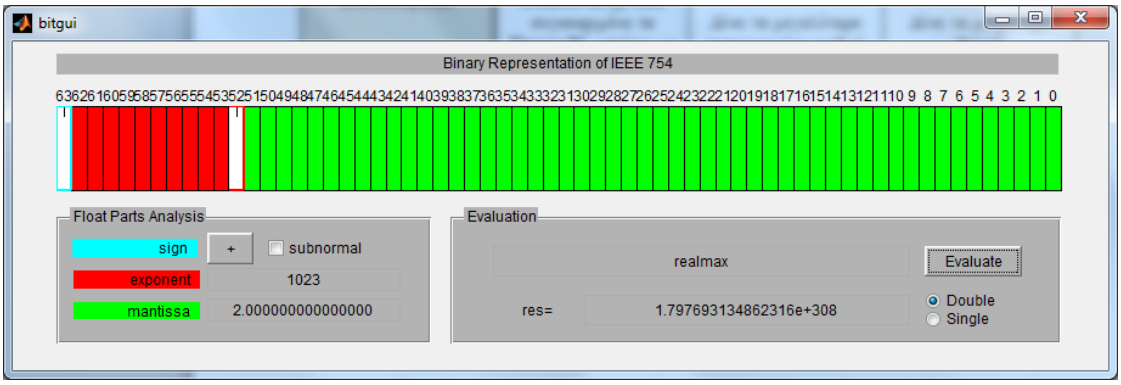
Στη συνέχεια εξετάζω την αναπαράσταση των στοιχείων της ανισότητας (**1+realmin == 1**). Αφού έχω παραθέσει ήδη αναπαράσταση για το 1. Για τον realmin λοιπόν, που είναι ο μικρότερος αναπαραστήσιμος αριθμός σε διπλή ακρίβεια, παρατηρούμε πως έχει μηδενικά στη mantissa του, όπως και στον εκθέτη του, εκτός από το λιγότερο σημαντικό ψηφίου του, που είναι 1.



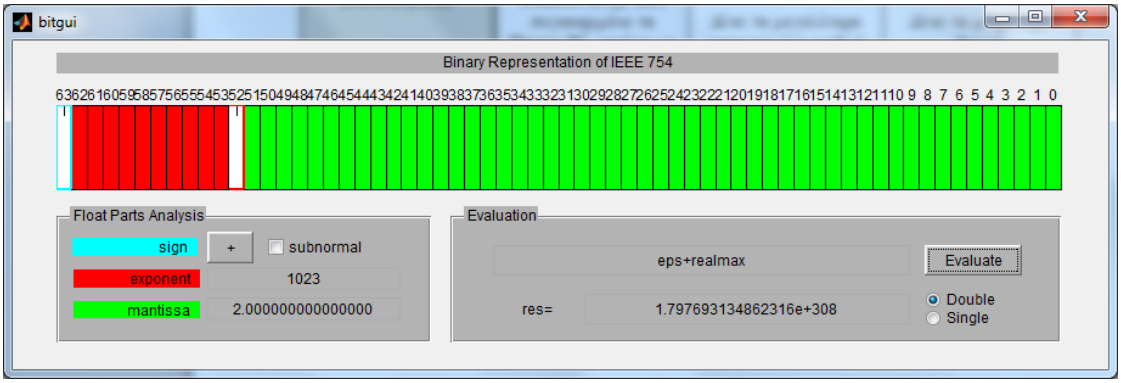
Όταν λοιπόν τώρα κάνουμε αυτή την πρόσθεση, παρατηρώ πως τα bit του αθροίσματος αυτού, είναι ίδια με αυτά του 1. Γνωρίζουμε γενικά για τη συμπεριφορά του Matlab, ότι όταν προστίθεται αριθμός που είναι μικρότερος του eps/2(το realmin είναι σημαντικά μικρότερό του όπως φαίνεται και στον παραπάνω πίνακα), τότε πραγματοποιείται στρογγύλευση στον κοντινότερο αναπαραστήσιμο αριθμό με τελευταίο bit ίσο με μηδέν. Οπότε σωστά εκτυπώνεται το 3, μιας και ισχύει η ισότητα της συνθήκης.



Τέλος εξετάζω την αναπαράσταση των στοιχείων της ισότητας (**eps+realmax == realmax**), και αρχικά παραθέτω το realmax, στο οποίο βλέπουμε τα bit της mantissa να ισούνται όλα με 1, καθώς και τα bit του εκθέτη, πέρα από το λιγότερο σημαντικό.



Όταν προσθέτω λοιπόν τώρα το eps στον realmax(το μεγαλύτερο αναπαραστήσιμο αριθμό σε διπλή ακρίβεια), δε φαίνεται κάποια αλλαγή, και αυτό γιατί όπως είπαμε και πριν, όταν προστίθεται κάτι πολύ μικρό σε έναν αριθμό(πόσο μάλλον μεγάλο αριθμό όπως στην περίπτωση του realmax), τότε η τιμή του αθροίσματος στρογγυλοποιείται προς τα κάτω.



Ερώτημα 2 – Διερεύνηση Σύγκλισης Σειρών:

(i) Για τον υπολογισμό των μερικών αθροισμάτων της σειράς της εκφώνησης, υλοποίησα την παρακάτω συνάρτηση Matlab, ώστε να τερματίζει όταν ικανοποιείται ένα κριτήριο σύγκλισης. Για το κριτήριο αυτό ξέρουμε ότι η διαφορά της νέας υπολογισμένης τιμής από την αμέσως προηγούμενή της, πρέπει να είναι ίση με 0.
Οπότε αν σε μία κατάσταση έχω μερικό άθροισμα έστω S = 1/n +1/n+…+1/n, και S' αύτο της αμέσως επόμενής της, τότε θα πρέπει |S'-S|=0 (S'=S+1/n).

Έτσι λοιπόν, ο κώδικάς μου είναι ο ακόλουθος.

```
function [] = harmonic_ser_2_i

crit = true;
new_value = 0;
prev_value = 0;
n = 1;

while crit
    new_value = prev_value + 1/n;
    if abs(new_value - prev_value) == 0
        crit = false;
    end
    prev_value = new_value;
    n = n + 1;
end
```

Σε αυτό το σημείο να κάνω την παρατήρηση πως κατασκευάζω ένα infinite loop, καθώς αρχικά, σύμφωνα και με την άποψη του μαθηματικού όπως βλέπουμε στο επόμενο ερώτημα, δεν ξέρω πότε θα ικανοποιηθεί το κριτήριο σύγκλισης. Έτσι, απλά η συνάρτηση θα τερματίσει όταν το κριτήριο αυτό ικανοποιηθεί.

Θα μπορούσα επίσης να συμπεριλάβω στον κώδικά μου μία εντολή εκτύπωσης disp, με τη διαφορά των δύο εκάστοτε τιμών κάθε φορά(new_value-prev_value), μετά την if δομή επιλογής, ώστε να παρακολουθώ τα αποτελέσματα της εκτέλεσης.

(ii) Σύμφωνα με την εκφώνηση, ο μαθηματικός προβλέπει ότι το πρόγραμμα δε θα τερματίσει. Αυτό θεωρώ πως συμβαίνει επειδή δε θα λάβει υπ'όψιν του το κριτήριο σύγκλισης.

Εμείς αντίθετα σα μηχανικοί, θεωρούμε πως το κριτήριο θα ικανοποιηθεί, δηλαδή ότι κάποια στιγμή, η διαφορά που μας ενδιαφέρει θα μηδενιστεί(στην πράξη βέβαια, εκτελωντας τον κώδικα στο Matlab, δε φαίνεται να τερματίζει, τουλάχιστον σε λογικό χρονικό διάστημα που να μπορεί να παρακολουθηθεί/καταγραφεί), καθώς μετά από όλο και μεγαλύτερη συσσώρευση αθροισμάτων, θα προστίθεται κάτι μικρού μεγέθους, σε κάτι μεγάλου μεγέθους, οπότε και θα επιφέρει κάποια(σημαντική)αλλαγή.

Αυτό που μπορώ να κάνω από μέρους μου πειραματικά, και να εκμεταλευτώ την ανυπαρξία άπειρης μνήμης του συστήματός μου, είναι να τρέξω μεγάλα νούμερα x σε ένα άθροισμα sum(1./(1:10^x))(όπου x δοκιμαστικοί αριθμοί), ώστε να βγει αποτέλεσμα στο Matlab "out of memory". Έτσι θα μπορέσω να κάνω μία θεωρητική προσέγγιση του αριθμού των επαναλήψεων.

Για δυνάμεις του 10^6, 10^7, 10^8 το άθροισμα υπολογίζεται κανονικά με αυτό τον τρόπο. Αν όμως δοκιμάσω με 10^9 εμφανίζεται το μήνυμα "out of memory" που ανέφερα παραπάνω. Δοκιμάζοντας στη συνέχεια τιμές ανάμεσα στο 10^8 και στο 10^9, καταλήγει να εμφανίζεται το μήνυμα περίπου στις $4x10^8$επαναλήψεις. Άρα θα χρειαστούν τουλάχιστον τόσες επαναλήψεις.

(iii) Όπως ανέφερα και στο προηγούμενο υποερώτημα, αν τρέξω τον κώδικά μου, δεν τερματίζει. Μπορώ να θεωρήσω πως αυτή η φαινομενικά προβληματική κατάσταση, οφείλεται στο ότι το Matlab μπορεί να αναπαραστήσει πάρα πολλούς αριθμούς, σύμφωνα και με το ότι στηρίζεται σε πρότυπο διπλής ακρίβειας.

Γνωρίζουμε λοιπόν ότι 1+eps/2 = 1 καθώς επίσης και ότι ο eps(k), είναι ο επόμενος αναπαραστήσιμος αριθμός του k. Από αυτά τα δεδομένα τώρα, αναρωτιέμαι για ποιές τιμές του k θα έχω την ισότητα $S_k = S_{k-1} + 1/k = S_{k-1}$ (ώστε να μηδενίζεται η διαφορά μου και να ικανοποιείται το κριτήριο).

Όπως ανέφερα και στο πρώτο υποερώτημα, |S'-S|=0, άρα θέλω μεταξύ των δύο αναπαραστήσιμων αριθμών να μην υπάρχει απόσταση μεγαλύτερη από eps/2 ώστε να γίνει στρογγυλοποίηση προς το μερικό άθροισμα της προγενέστερης κατάστασης. Οπότε συγκεντρωτικά από τις παραπάνω ισότητες έχω:

S - (S-1/k) < eps/2 => 1/k < eps/2 , όπου eps ουσιαστικά είναι το eps(next_value) σύμφωνα με τον κώδικα

(iv) Για την υλοποίηση τώρα του ίδιου κώδικα σε μονή ακρίβεια, οι μικροαλλαγές που γίνονται με την single, και είναι στις πράξεις του κώδικα και πουθενά αλλού, π.χ. εκχωρήσεις.

Οπότε και έχω τον ακόλουθο κώδικα:

```
function [] = harmonic_ser_2_iv

crit = true;
new_value = 0;
prev_value = 0;
n = single(1);

while crit
    new_value = single(prev_value) + single(1/n);
    if abs(new_value - prev_value) == 0
        crit = false;
    end
    prev_value = new_value;
    n = n + 1;
end
```

Σε αυτή την εκτέλεση τώρα, με τη χρήση αποκλειστικά μονής ακρίβειας, παρατηρώ πως και τερματίζει ο κώδικας, αλλά και σε μικρό χρονικό διάστημα. Οι επαναλήψεις που χρειάστηκαν για να τερματίσει ήταν 2097152(η τιμή του n στην οποία κρατάμε τον αριθμό των επαναλήψεων είναι ουσιαστικά τελικά 2097153, αλλά σύμφωνα με τον κώδικα, στον οποίο αυξάνουμε στο τέλος του while το n, αφότου ικανοποιηθεί η συνθήκη του if, στη συνέχεια αυξάνται μία φορά ακόμα). Παρατηρώ ότι είναι πολύ μικρότερης τάξης από τις ελάχιστες επαναλήψεις που θα μπορούσε να κάνει ο κώδικας για διπλή ακρίβεια.

Επίσης, το ότι τερματίζει οφείλεται πάλι στη μονή ακρίβεια, μιας και το εκάστοτε eps θα είναι μεγαλύτερο σε σχέση με το eps που θα χρειαζότανε για να ικανοποιηθεί το αρχικό μας κριτήριο. Δεδομένου ότι η σειρά αποκλίνει με πολύ αργό ρυθμό σύμφωνα με τη θεωρία, θα φτάνει συγκριτικά πιο γρήγορα στο eps του μερικού αθροίσματος για μονή ακρίβεια, ενώ για διπλή θα θέλει ακόμαι και εκθετικά πολλαπλάσιο για να φτάσει.

Ερώτημα 3 – Πράξεις με Πολυώνυμα:

Στην παράγραφο που ακολουθεί, θα εξηγήσω για ποιο λόγο χρησιμοποιούνται οι συναρτήσεις poly, roots και polyval, έχοντας μελετήσει και τους κώδικές τους από το Matlab.

- **poly**: Η poly δέχεται ως είσοδο τις ρίζες ενός πολυωνύμου και επιστρέφει τους συντελεστές αυτού του πολυωνύμου κατά φθίνουσα δύναμη. Όταν το V είναι ένα διάνυσμα στην poly(V), είναι ένα διάνυσμα του οποίου τα στοιχεία, είναι οι συντελεστές του πολυωνύμου, του οποίου οι ρίζες είναι στοιχεία του διανύσματος V.
  Πιο συγκεκριμένα η συνάρτηση poly ελέγχει αρχικά τα χαρακτηριστικά του πολυωνύμου. Απομακρύνει τις τιμές που απειρίζονται, και χρησιμποιεί φόρμουλα αναφρομής. Τα αποτελέσματα που δίνει είναι «πραγματικά», αν οι ρίζες είναι συζυγείς μιγαδικές.

- **roots**: Η roots βρίσκει πολυωνυμικές ρίζες. Υπολογίζει τις ρίζες του πολυωνύμου του οποίου οι συντελεστές είναι τα στοιχεία του διανύσματος V(roots(V)). Αν το V έχει N+1 στοιχεία, το πολυώνυμο είναι το εξής: V(1)*X^N+…+V(N)*X+V(N+1). Για διανύσματα, η roots και η poly είναι αντίστροφες συναρτήσεις μεταξύ τους.
  Πιο συγκεκριμένα και πάλι η roots βρίσκει τις ιδιοτιμές του συνοδού πίνακα, απομακρύνει τα μηδενικά που οδηγούν και αυτά που ακολουθούν, ενώ τα θυμάται σα ρίζες στο μηδέν. Προστατεύει σχετικά μικρούς συντελεστές από το να απειριστούν, αφαιρώντας τους. Βρίσκει τις πολυωνυμικές ρίζες διαμέσου του συνοδού πίνακα.

- **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.

  $$Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)$$

  Αν το X είναι μητρώο η διάνυσμα, τότε το πολυώνυμο υπολογίζεται σε όλα τα σημεία στο X.
  Πιο συγκεκριμένα και γι αύτη τη συνάρτηση λοιπόν, 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 αποτελέσματα της polyfit. Κατασκευάζει μητρώο Vandermonde για το καινούριο X.

Μέρος Α

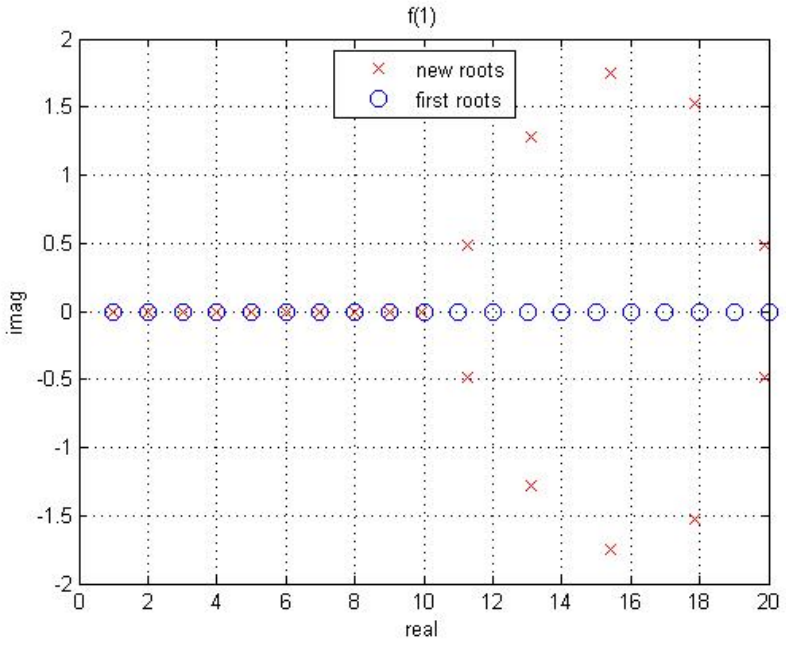
(Α) Το script Matlab που υλοποίησα για τα ρωτήματα του (Α) υποερωτήματος, είναι το ακόλουθο:

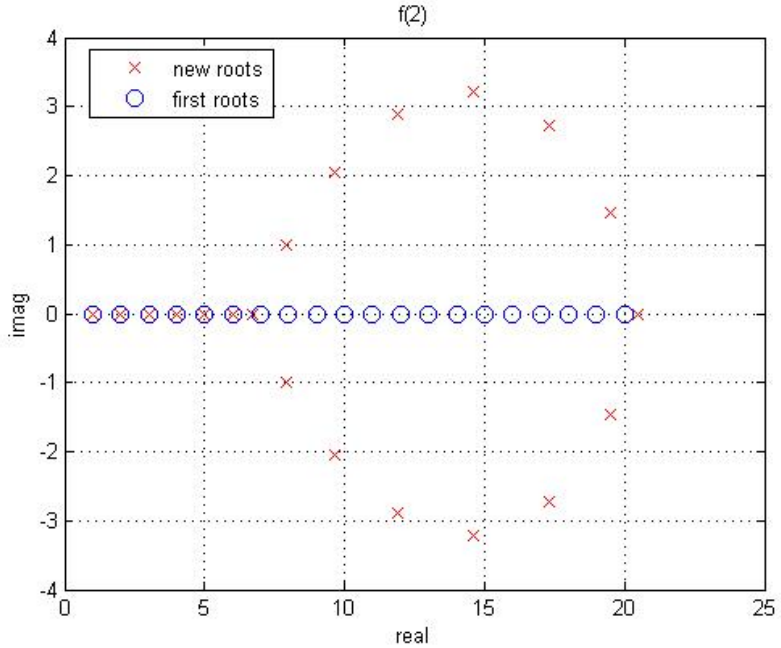
```
%Υπολογισμός συντελεστών από τις γνωστές ρίζες
z = [1:20];
p = poly(z);
%Δήλωση συνόλων διαταραχών
e = [10^(-8) (-10^3) 10^10] ;
e_all = [10^(-8)];
e_all_rel = p*(2^(-52));
%Αρχικοποίηση του πολυωνύμου g και προσδιορισμός των συντελεστών
που θα διαταραχθούν
g = zeros(5,21);
g(1,2) = 1;
g(2,10) = 1;
g(3,20) = 1;
g(4,2:21) = 1;
g(5,2:21) = 1;
%Αρχικοποίηση του πολυωνύμου f και υπολογισμός των νέων συντελεστών
f = zeros(5,21);
f(1,:) = p + g(1,:).*e(1);
f(2,:) = p + g(2,:).*e(2);
f(3,:) = p + g(3,:).*e(3);
f(4,:) = p + g(4,:).*e_all;
f(5,:) = p + g(5,:).*e_all_rel;
%Οι ρίζες των πολυωνύμων μετά τις διαταραχές
new_roots = zeros(5,20);
new_roots(1,:) = roots(f(1,:));
new_roots(2,:) = roots(f(2,:));
new_roots(3,:) = roots(f(3,:));
new_roots(4,:) = roots(f(4,:));
new_roots(5,:) = roots(f(5,:));
%Διαχωρισμός ριζών πολυωνύμου σε πραγματικές και φανταστικές
relative_error = zeros(5,20);
first_roots = zeros(5,20);
for i=1:5
   if (isreal(new_roots(i,:)) == 1)
       for j=1:20
           [diff, posi] = min(abs(new_roots(i,j)-z));
           relative_error(i,j) = abs(new_roots(i,j)-
posi)/abs(posi);
           first_roots(i,j) = posi;
       end
   else
       new_roots(i,:) = sort(new_roots(i,:), 'ascend');
       for j=1:20
           relative_error(i,j) = abs(new_roots(i,j)-
z(j))/abs(z(j));
           first_roots(i,j) = z(j);
       end
   end
end
```

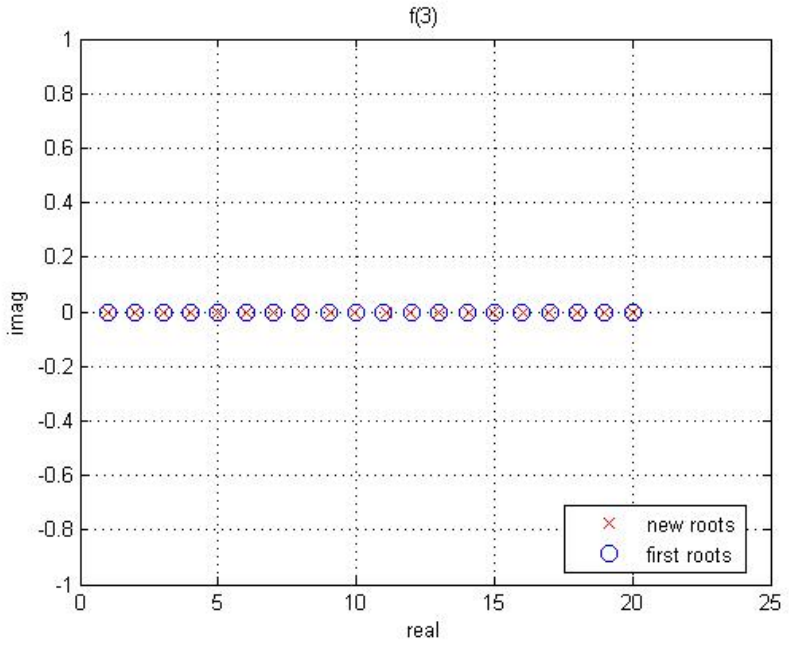
```
%Φθήνουσα ιεραρχία και υπολογιζόμενες και γνωστές ρίζες στο
μιγαδικό επίπεδο
for i=1:5
    [S,I] = sort(relative_error(i,:), 'descend');
    relative_error(i,:) = S;
    new_roots(i,:) = new_roots(i,I);
    first_roots(i,:) = first_roots(i,I);
end
%Γράφημα για την πρώτη διαταραχή
figure
plot(real(new_roots(1,:)),imag(new_roots(1,:)),'rx','MarkerSize',8)
;
hold on;
plot(real(first_roots(1,:)),zeros(1,20),'bo','MarkerSize',8);
hold off; grid on;
title('f(1)'); xlabel('real'); ylabel('imag');
legend('new roots','first roots','Location','Best');
% Γράφημα για τη δεύτερη διαταραχή
figure
plot(real(new_roots(2,:)),imag(new_roots(2,:)),'rx','MarkerSize',8)
;
hold on;
plot(real(first_roots(2,:)),zeros(1,20),'bo','MarkerSize',8);
hold off; grid on;
title('f(2)'); xlabel('real'); ylabel('imag');
legend('new roots','first roots','Location','Best');
% Γράφημα για την τρίτη διαταραχή
figure
plot(real(new_roots(3,:)),imag(new_roots(3,:)),'rx','MarkerSize',8)
;
hold on;
plot(real(first_roots(3,:)),zeros(1,20),'bo','MarkerSize',8);
hold off; grid on;
title('f(3)'); xlabel('real'); ylabel('imag');
legend('new roots','first roots','Location','Best');
% Γράφημα για την τέταρτη διαταραχή
figure
plot(real(new_roots(4,:)),imag(new_roots(4,:)),'rx','MarkerSize',8)
;
hold on;
plot(real(first_roots(4,:)),zeros(1,20),'bo','MarkerSize',8);
hold off; grid on;
title('f(4)'); xlabel('real'); ylabel('imag');
legend('new roots','first roots','Location','Best');
% Γράφημα για την πέμπτη διαταραχή
figure
plot(real(new_roots(5,:)),imag(new_roots(5,:)),'rx','MarkerSize',8)
;
hold on;
plot(real(first_roots(5,:)),zeros(1,20),'bo','MarkerSize',8);
hold off; grid on;
title('f(5)'); xlabel('real'); ylabel('imag');
legend('new roots','first roots','Location','Best');
```

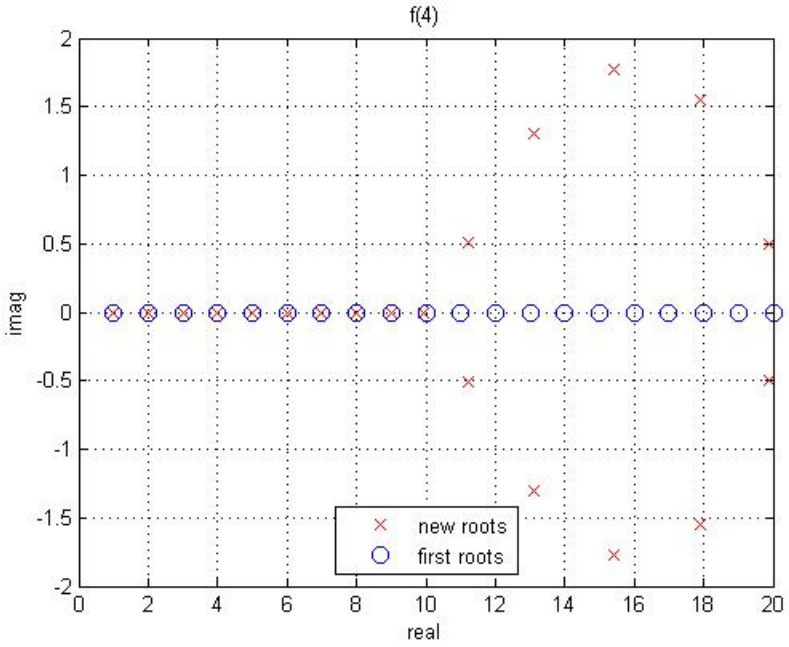
Σε αυτό το σημείο να σημειώσω πως στο γενικότερο παραδοτέο πακέτο της άσκησης, έχω συμπεριλάβει μέσα στον αντίστοιχο φάκελο τα αποτελέσματα των f(διαταραγμένοι συντελεστές), first_roots(γνωστές ρίζες), new_roots(ρίζες των διαταραγμένων πολυωνύμων), relative_error(σχετικά σφάλματα), όλα ταξινομημένα(εκτός από τους συντελεστές).

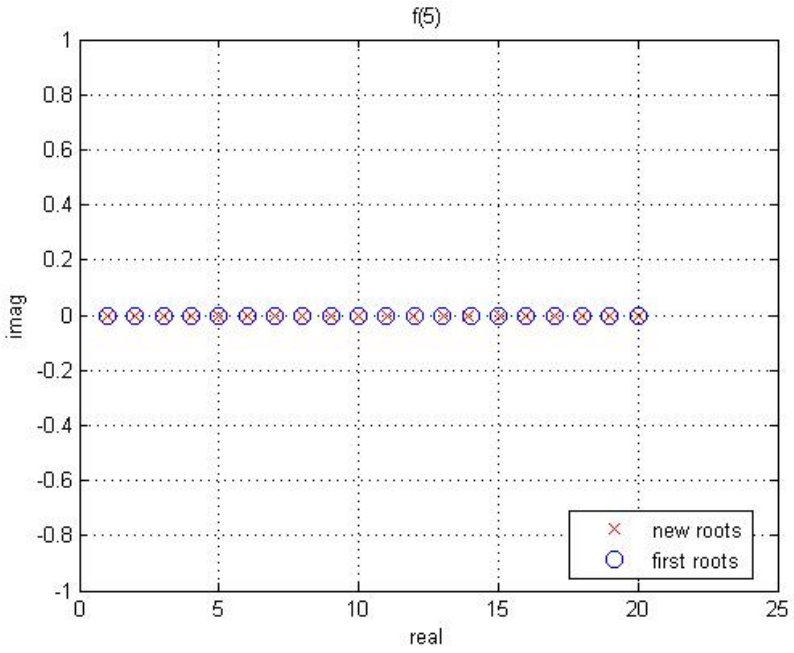
Στη συνέχεια παραθέτω τις γραφικές παραστάσεις των υπολογιζόμενων και γνωστών ριζών του πολυωνύμου στο μιγαδικό επίπεδο.






f(3)
imag
real
new roots
first roots


f(4)
imag
real
new roots
first roots



(B) Αρχικά παρατηρώ ότι μεγαλύτερες διαταραχές στις ρίζες προκύπτουν στις περιπτώσεις 1, 2 και 4, ενώ στις περιπτώσεις 3 και 5 οι διαταραχές αυτές είναι μικρές. Επίσης το γεγονός ότι έχουμε μιγαδικές ρίζες στις πρώτες περιπτώσεις ενω γνωρίζουμε ότι το αρχικό πολυώνυμο είχε μόνο πραγματικές, προκαλεί έκπληξη μιας και σύμφωνα με τις διαταραχές των συντελεστών δεν περιμέναμε τέτοια αποτελέσματα.

Πιο συγκεκριμένα οι μεγάλες διαταραχές 1, 2, 4 οφείλονται στο γεγονός ότι κάθε φορά η διαταραχή είναι «αισθητή» από τους συντελεστές. Στην περίπτωση 1 ο συντελεστής α_{19} είναι -210 και τον διαταράσσουμε κατά 10^{-8}, στην περίπτωση 2 ο α_{11} είναι $-1.3558\text{x}10^{14}$ και τον διαταράσσουμε κατά -10^{3}, ενώ στην περίπτωση 4 έχουμε πάλι τη διαταραχή 10^{-8}, που σίγουρα πειράζει πάλι πολύ τον α_{19}.(και γι αυτό το λόγο μοιάζουν οι 2 γραφικές παραστάσεις, 1 και 4. Βέβαια και η δύναμη της μεταβλητής της οποίας επηρεάζεται και ο συντελεστής κάθε φορά, παίζι ρόλο, πχ ο α_{19} είναι συντελεστής της x_{19}, άρα υπάρχει μεγάλη επηρροή μέσω αυτού στις ρίζες.

Αντίθετα, στις περιπτώσεις 3 και 5 έχουμε διαταραχή σε συντελεστής μικρών δυνάμεων ή μικρό μέγεθος διαταραχής. Ειδικότερα στην περίπτωση 3 επηρρεάζεται ο α_1 συντελεστής, ο οποιός από ότι φαίνεται δεν έχει μεγάλη επίδραση στις τιμές του πολυωνύμου. Στην περίπτωση 5, διαταράσσονται όλοι οι συντελεστές, πλην του α_{20}, κατά πολύ μικρό ποσοστό, στην ουσία κατά 1 bit, μιας και το 2_{-52} με το οποίο πολλαπλασιάζονται, είναι πολύ μικρό και αλλάζει το τελευταίο bit στη δυαδική τους αναπαράσταση σύμφωνα με το πρότυπο της IEEE.

Τέλος αξίζει να σχολιάσω πως εμφανίζονται μεγαλύτερα σφάλματα στις μεγάλες ρίζες σε σχέση με τις μικρότερες, όπως για παράδειγμα στις $1^{η}$ και $4^{η}$ περίπτωση στις οποιές οι νέες ρίζες που αντιστοιχούν στις γνωστές ρίζες από 1 έως 11 είναι πλησιέστερες σε αυτές από ότι οι υπόλοιπες.

Σύμφωνα με τα παραπάνω, όντως η συμπεριφορά του πολυωνύμου είναι δόλια, όπως το είχε χαρακτηρίσει και ο J.Wilkinson, μιας και σε πολλές περιπτώσεις δεν περιμένουμε τέτοια ακραία αποτελέσματα.

Μέρος Β
