ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΙΣΤΗΜΟΝΙΚΟΣ ΥΠΟΛΟΓΙΣΜΟΣ Ι

ΔΕΥΤΕΡΗ ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Καφφέζας Γιώργος · ΑΜ 4465 · kaffezas@ceid.upatras.gr

×

Η αναφορά αυτή αφορά την δεύτερη εργαστηριακή άσκηση του μαθήματος «Επιστημονικού Υπολογισμού» για το χειμερινό εξάμηνο του ακαδημαϊκού έτους 2013-2014. Συντάχθηκε με τη βοήθεια του LaTeX και του editor TeXMaker, ενώ το εξώφυλλο βασίστηκε σε κώδικα του Peter Wilson από την ιστοσελίδα www.latextemplates.com. Επιλέχθηκε πρότυπο βιβλίου και η σελιδοποίηση έγινε ώστε η αναφορά να εκτυπώνεται καλά σε μορφή φυλλαδίου μεγέθους A4.

×

Πληροφορίες Συστήματος

• Τα στοιχεία του υπολογιστικού συστήματος στο οποίο πραγματοποιήθηκε η άσκηση είναι τα ακόλουθα, όπως προκύπτουν και μετά από την χρήση των προγραμμάτων CPU-Z και PC Wizard:

Λειτουργικό σύστημα Windows 7 Professional SP1 (×64)

Τύπος επεξεργαστή Intel Core 2 Quad Q6600 @2.40GHz

Επίπεδα κρυφής μνήμης
L1 data cache: 4×32 KB, 8-way set assoc, 64-byte line size
L1 instruction cache: 4×32 KB, 8-way set assoc, 64-byte line size
L2 cache: 2×4096 KB, 16-way set assoc, 64-byte line size

Πολιτική εγγραφής στην cache write-back

• Η έκδοση MATLAB που χρησιμοποιήθηκε για την άσκηση είναι η R2012b (8.0.0.783) για λειτουργικό σύστημα 64-bit.

ΕΡΩΤΗΜΑ 1 - Χαρακτηριστικά Αριθμητικής στο MATLAB

i) Ο ζητούμενος πίνακας με τα αποτελέσματα και τις πληροφορίες είναι ο εξής (τα αποτελέσματα εμφανίζονται με format short ενώ οι πράξεις στη MATLAB γνωρίζουμε ότι γίνονται από default με αριθμητική διπλής ακρίβειας):

	single	**double**	**Περιγραφή**
eps	1.1921×10^{-7}	2.2204×10^{-16}	Επιστρέφει την απόσταση του αριθμού 1.0 από τον αμέσως επόμενο αριθμό μονής και αντίστοιχα διπλής ακρίβειας. Γενικότερα, με όρισμα κάποιον αριθμό επιστρέφει την απόστασή του από τον επόμενο αριθμό που μπορεί να αναπαρασταθεί.
realmax	3.4028×10^{38}	1.7977×10^{308}	Επιστρέφει τον μεγαλύτερο πεπερασμένο αριθμό κινητής υποδιαστολής σύμφωνα με τη μονή και διπλή ακρίβεια της IEEE.
realmin	1.1755×10^{-38}	2.251×10^{-308}	Επιστρέφει τον μικρότερο κανονικοποιημένο θετικό αριθμό κινητής υποδιαστολής σύμφωνα με τη μονή και διπλή ακρίβεια της IEEE.

ii) Ο κώδικας MATLAB για το παρόν ερώτημα είναι ο ακόλουθος:

```
1  if (eps/2 == 0),
2      disp(1)
3  end
4
5  if (1+eps > 1),
6      disp(2)
7  end
8
9  if (1+realmin == 1),
10     disp(3)
11 end
12
13 if (eps+realmax == realmax),
14     disp(4)
15 end
```

files/1/ii/script_2_1_ii.m

Μετά από την εκτέλεσή του στο τερματικό εκτυπώθηκαν οι αριθμοί 2, 3 και 4, δηλαδή ικανοποιήθηκαν οι συνθήκες ελέγχου για όλα τα if-block εκτός του πρώτου. Ας δούμε πιο αναλυτικά το γιατί.

(eps/2 == 0) → false

Γνωρίζουμε ότι το eps ισούται περίπου με 2.2204×10^{-16}, ή ακριβώς σε δεκαεξαδική μορφή 3cb0000000000000. Επομένως, όταν το διαιρέσουμε με το

2, το αποτέλεσμα είναι 3ca0000000000000. Αυτός ο αριθμός είναι σίγουρα μεγαλύτερος του μηδενός και αναπαραστήσιμος, συνεπώς το λογικό αποτέλεσμα της σύγκρισης είναι false.

(1+eps >1) → true

Δεδομένου ότι το eps είναι η απόσταση του 1 από τον αμέσως επόμενο αριθμό κινητής υποδιαστολής, η πρόσθεσή του στο 1 θα έχει ως αποτέλεσμα τον εν λόγω αριθμό. Πιο συγκεκριμένα, η δεκαεξαδική μορφή του 1 είναι το 3ff0000000000000 και του eps είναι το 3cb0000000000000. Άρα, το αποτέλεσμά τους ισούται με 3ff0000000000001, δηλαδή ισούται όντως με τον αμέσως επόμενο αναπαραστήσιμο αριθμό από το 1. Κατ΄ επέκταση είναι μεγαλύτερος αριθμός του 1 κι έτσι το λογικό αποτέλεσμα της πράξης ελέγχου είναι true.

(1+realmin == 1) → true

Γνωρίζουμε ότι το realmin ισούται περίπου με 2.251×10^{-308}, ενώ η δεκαεξαδική αναπαράστασή του είναι ο αριθμός 0010000000000000. Επίσης, γνωρίζουμε ότι το eps ισούται με 2.2204×10^{-16} και το eps/2 με 1.1102×10^{-16}, ή σε δεκαεξαδική μορφή 3ca0000000000000. Βλέπουμε και με τις δύο μορφές ότι το realmin είναι κατά πολύ μικρότερο του eps/2, και κατά συνέπεια η προσθήκη του στο 1 δε θα το επηρεάσει. Αυτό συμβαίνει γιατί, καθώς το realmin είναι μικρότερο από το μισό της απόστασης του 1 από τον αμέσως επόμενο αριθμό, η MATLAB «αποφασίζει» να το στρογγυλοποιήσει προς τα κάτω, δηλαδή προς το 1. Έτσι, το λογικό αποτέλεσμα της πράξης ελέγχου είναι true.

(eps+realmax == realmax) → true

Γνωρίζουμε ότι το eps είναι η απόσταση του αριθμού 1 από τον αμέσως επόμενό του, και ισούται με 3cb0000000000000. Η αντίστοιχη απόσταση του realmax από τον αμέσως επόμενο αναπαραστήσιμο αριθμό του (δηλαδή, το «άπειρο» Inf που ισούται με 7ff0000000000000)είναι ίση με 7ca0000000000000, ενώ το realmax ισούται με 7fefffffffffffff. Έτσι, κατά την προσθήκη του eps στο realmax η MATLAB «αποφασίζει» να στρογγυλοποιήσει το αποτέλεσμα προς τα κάτω, δηλαδή προς το realmax, μιας και το eps είναι κατά πολύ μικρότερο του eps(realmax)/2.

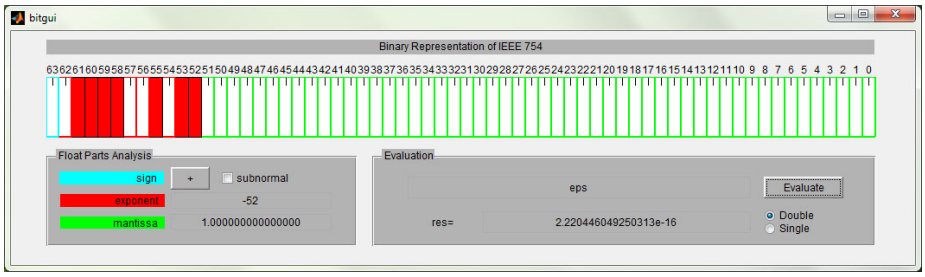
Στο σημείο αυτό αξίζει να εξετάσουμε λίγο πιο συγκεκριμένα το πρότυπο IEEE 754[1] για την αναπαράσταση αριθμών κινητής υποδιαστολής, και συγκεκριμένα για την αναπαράστασή τους με χρήση 64 bit, μιας και σε αυτή γίνονται οι πράξεις στη MATLAB εκ default.

Το πρώτο πιο σημαντικό bit χρησιμοποιείται για την αναπαράσταση του προσήμου του αριθμού και είναι 0, όταν ο αριθμός είναι θετικός, ενώ 1, όταν ο αριθμός είναι αρνητικός. Έπειτα, τα επόμενα 11 bit χρησιμοποιούνται για την αναπαράσταση του εκθέτη του αριθμού, και ειδικά για την αναπαράστασή του με πόλωση. Αυτό σημαίνει ότι αν ε ο εκθέτης, τότε η τιμή των 11 bit προκύπτει από τον τύπο $\varepsilon - 2^{11} + 1 = \varepsilon - 1023$. Τέλος, τα υπόλοιπα 52 bit χρησιμοποιούνται για την αναπαράσταση του λεγόμενου σημαντικού μέρους του αριθμού και ονομάζονται mantissa. Υπάρχει, βέβαια, και ένα κρυμμένο bit που «εννοείται» πριν την υποδιαστολή, με το οποίο ουσιαστικά επιλέγεται αν θα είναι κανονικοποιημένη ή μη η αναπαράσταση του αριθμού, δηλαδή αν θα είναι 1 ή 0.

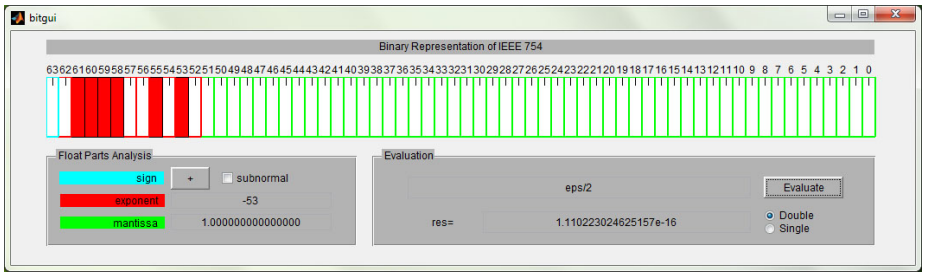
[1] https://en.wikipedia.org/wiki/IEEE_floating_point

Έχοντας αυτά υπ' όψιν μας, μπορούμε να δούμε πιο αναλυτικά κάθε μία από τις παραπάνω περιπτώσεις. Θα χρησιμοποιήσουμε και εικόνες από τη διεπαφή του bitgui[2] για καλύτερη και ακριβέστερη αναπαράσταση των αριθμών.

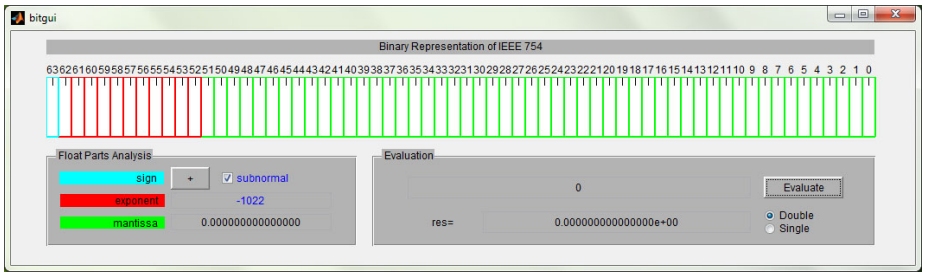
(eps/2 == 0)



Βλέπουμε ότι το eps ισούται με 2^{-52}, καθώς τα bit του εκθέτη είναι ίσα με τον δεκαδικό 971, καθώς ο δυαδικός είναι ο 01111001011. Όπως είπαμε και πριν, ο αριθμός αυτός είναι πολωμένος, κι έτσι ο εκθέτης του 2 προκύπτει από τον τύπο 971 - 1023 = -52.



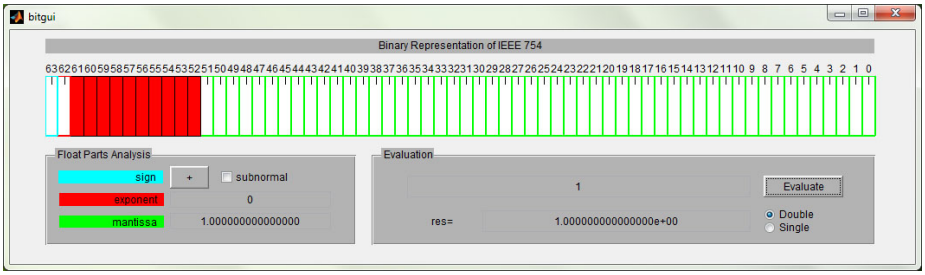
Εδώ το αποτέλεσμα της διαίρεσης του eps με το δύο αλλάζει μόνο ένα bit στον εκθέτη, όπως ήταν αναμενόμενο.



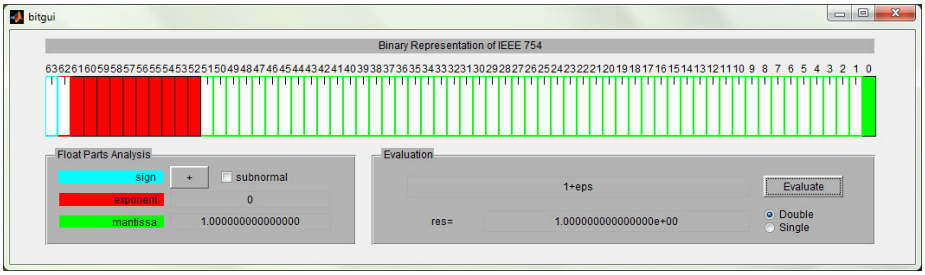
Τέλος, η αναπαράσταση του μηδενός είναι με υποκανονικοποιημένο τρόπο και με όλα τα bit ίσα με μηδέν. Είναι προφανές ότι η τιμή του μηδενός διαφέρει από αυτή του eps/2, γι' αυτό προκύπτει και το ανάλογο λογικό αποτέλεσμα false κατά τη σύγκρισή τους.

[2] http://www.mathworks.com/matlabcentral/fileexchange/33874-bitgui-a-graphical-explorer-of-the-ieee-754-floating-point-formats

(1+eps >1)

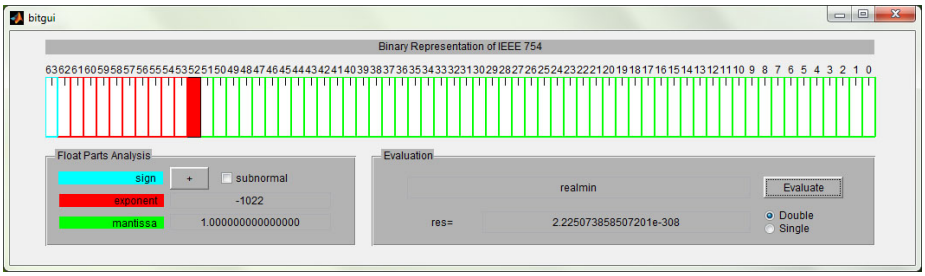


Βλέπουμε ότι το 1 αναπαρίσταται με όλα τα bit της mantissa να είναι ίσα με το μηδέν, και τα bit του εκθέτη να σχηματίζουν τον αριθμό 1023 (καθώς 01111111111) κι έτσι $2^{1023\text{-}1023} = 2^0 = 1$.

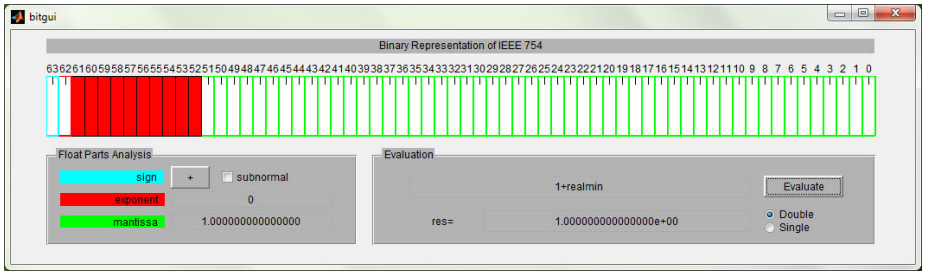


Εδώ παρατηρούμε ότι με την πρόσθεση του eps στο 1 ο αριθμός όντως αλλάζει, καθώς το λιγότερο σημαντικό bit γίνεται 1, ενώ ο εκθέτης παραμένει ίδιος. Ο αριθμός που προκύπτει είναι όντως ο αμέσως επόμενος αριθμός από το 1 που μπορεί να αναπαρασταθεί με διπλή ακρίβεια από το πρότυπο της IEEE.

(1+realmin == 1)

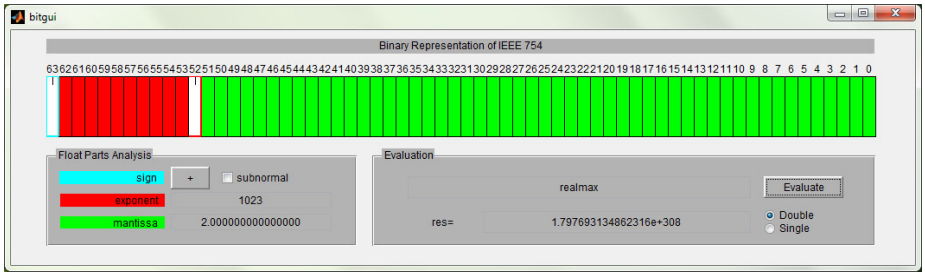


Παρατηρούμε ότι ο μικρότερος κανονικοποιημένος αριθμός που μπορεί να αναπαρασταθεί με διπλή ακρίβεια σύμφωνα με το πρότυπο έχει μηδενικά bit εκτός από τον εκθέτη, ο οποίος είναι ίσος με 1, δηλαδή αντιστοιχεί στο $2^{1\text{-}1023} = 2^{\text{-}1022}$.

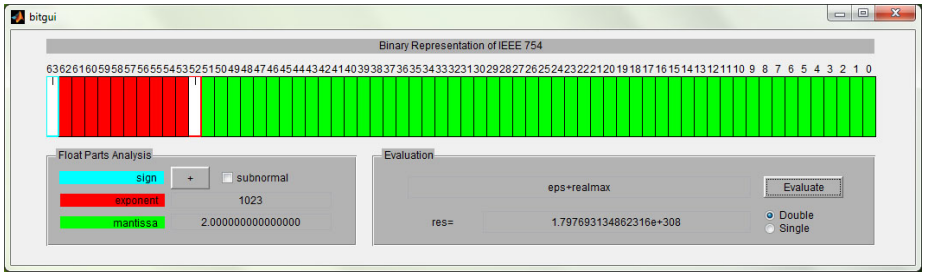


Όταν προσθέσουμε αυτόν τον αριθμό στο 1, παρατηρούμε ότι το 1 δεν επηρεάζεται, ότι όλα τα bit του μένουν ίδια. Αυτό όπως είπαμε συμβαίνει εξαιτίας της πολιτικής της MATLAB για στρογγυλοποίηση στον πλησιέστερο αριθμό με τελευταίο bit ίσο με το μηδέν, όταν ο προστιθέμενος αριθμός είναι μικρότερος ή ίσος του eps/2. Στην περίπτωσή μας, όντως το realmin είναι κατά πολύ μικρότερο του eps/2, καθώς ισχύει ότι $10^{-308} \ll 10^{-16}$.

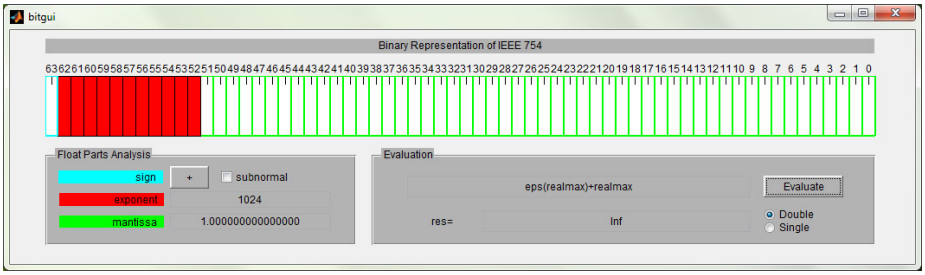
(eps+realmax == realmax)



Παρατηρούμε ότι το realmax αναπαρίσταται έχοντας σχεδόν όλα του τα bit ίσα με 1, εκτός ένα από του εκθέτη και του προσήμου. Τα bit της mantissa είναι όλα ίσα με 1.



Αν προσθέσουμε στο realmax το eps παρατηρούμε ότι δεν επέρχεται κάποια αλλαγή. Όπως είπαμε και πιο πριν, αυτό οφείλεται στο γεγονός ότι το eps είναι κατά πολύ μικρότερο του eps(realmax)/2 καθώς ισχύει ότι $2^{-52} \ll 1.7977 \cdot 2^{1023}$, *με συνέπεια η MATLAB να στρογγυλοποιεί το αποτέλεσμα προς τα κάτω.*



Όντως, αν προσθέσουμε αντί του eps το eps(realmax), τότε βλέπουμε ότι ο αριθμός που προκύπτει είναι το «άπειρο» (Inf). Άρα, γι΄ αυτό δεν επηρεάστηκε με την προσθήκη του απλού eps.

ΕΡΩΤΗΜΑ 2 - Διερεύνηση Σύγκλισης Σειρών

i) Η συνάρτηση MATLAB για το παρόν ερώτημα είναι η ακόλουθη:

```
1  function [] = script_2_2_i
2  % Variable initialization.
3  summary = 0; temp_sum = 0;
4  n = 1; run = true;
5  % Infinite loop til criterion's fulfilled.
6  while run
7      % Temporary summary.
8      temp_sum = summary + 1/n;
9      % Criterion.
10     if abs(temp_sum - summary) == 0
11         run = false;
12     end
13     % Displaying current difference.
14     disp(abs(temp_sum - summary))
15     summary = temp_sum; % Current summary.
16     n = n + 1; % Increment for next division.
17 end
```

files/2/i/script_2_2_i.m

ii) Εκτελώντας το παραπάνω πρόγραμμα γνωρίζουμε ότι θα τερματίσει, καθώς κάποια στιγμή το υπάρχον μερικό άθροισμα θα είναι πολύ πιο μεγάλο από τον προστιθέμενο όρο, με αποτέλεσμα να μην επηρεαστεί από την προσθήκη του κι έτσι να ικανοποιηθεί το κριτήριο τερματισμού του προγράμματος. Σύμφωνα με τη θεϊκή αριθμητική, την οποία θα είχε στο νου του κάποιος μαθηματικός, το πρόγραμμα δε θα τερμάτιζε ποτέ, καθώς η αρμονική σειρά που εξετάζουμε αποκλίνει (και μάλιστα με πολύ αργό ρυθμό).[3]

Εκτός αυτού, όμως, το πρόγραμμά μας υπάρχει περίπτωση να τερματίσει εξαιτίας της αδυναμίας διαχείρισης του μεγέθους των αριθμών που σχηματίζονται. Δεδομένου ότι η MATLAB έχει συγκεκριμένο χώρο διαθέσιμο για την αποθήκευση μητρώων, υπάρχει περίπτωση η συγκεκριμένη πράξη να βγει «εκτός μνήμης». Για να εξετάσουμε αυτή την περίπτωση θα χρησιμοποιήσουμε την εξής γραμμή κώδικα, η οποία εκφράζει το άθροισμα το οποίο μελετάμε και εμείς με τη βοήθεια της ενδογενούς συνάρτησης sum:

$$\text{s} = \text{sum}(1./(1{:}\text{boundary}))$$

όπου boundary είναι αριθμός του παρονομαστή μέχρι τον οποίο μπορεί να φτάσουμε πριν μας βγάλει το μήνυμα σφάλματος για τη μνήμη.

Στο υπολογιστικό μας σύστημα η εντολή memory μας επιστρέφει τα εξής:

```
Maximum possible array:      7626 MB (7.997e+09 bytes) *
Memory available for all arrays:      7626 MB (7.997e+09 bytes) *
Memory used by MATLAB:       606 MB (6.358e+08 bytes)
Physical Memory (RAM):      4094 MB (4.293e+09 bytes)

*  Limited by System Memory (physical + swap file) available.
```

[3] https://en.wikipedia.org/wiki/Harmonic_series_(mathematics)#Rate_of_Divergence

Δηλαδή, η MATLAB μπορεί να αποθηκεύσει μέχρι και περίπου 953 double αριθμούς, μιας και ένας τέτοιος αριθμός απαιτεί 8MB.

Έτσι, μετά από αρκετό πειραματισμό με το μέγεθος του boundary, στο υπάρχον υπολογιστικό σύστημα προέκυψε το εν λόγω σφάλμα περίπου ανάμεσα από το 449500000 (4.495×10^8) και το 450000000 (4.5×10^8), άρα κάπου εκεί κυμαίνονται και οι επαναλήψεις που πιθανόν να χρειαστούν για την εκτέλεση του προγράμματός μας. Θα εξετάσουμε, όμως, λίγο τα δεδομένα για το άθροισμα μέχρι και την εκτέλεση για boundary ίσο με 449500000.

Το άθροισμα στο σημείο εκείνο ισούται με 20.5009, ενώ ο τελευταίος όρος που προστέθηκε ήταν ίσος με 2.2247×10^{-9}. Το eps του μέχρι τότε αθροίσματος ισούται με 3.5527×10^{-15}. Γίνεται προφανές, λοιπόν, ότι η εκτέλεση δε θα σταμάταγε εκεί, αν δεν υπήρχε το πρόβλημα με τη μνήμη, καθώς ο επόμενος όρος που θα επρόκειτο να προστεθεί θα ήταν αρκετά μεγαλύτερος από το eps του αθροίσματος, και δεδομένων των αργών ρυθμών που αναφέραμε προηγουμένως, ο χρόνος εκτέλεσης που απέμενε ήταν πολύς.

iii) Παρά τα όσα είπαμε στο προηγούμενο υποερώτημα, φαίνεται ότι το πρόγραμμα δεν τερματίζει, τουλάχιστον όχι σε σχετικά σύντομο χρονικό διάστημα. Δηλαδή, παρατηρούμε ότι ο χρόνος εκτέλεσής του είναι αρκετά μεγάλος, και είναι πολύ πιθανό να ανέρχεται ακόμα και σε μέρες, από τη στιγμή που και η ενδογενής συνάρτηση για το ίδιο πράγμα απαιτεί αρκετό χρόνο.

Αυτό οφείλεται και στο γεγονός ότι η MATLAB χρησιμοποιεί το πρότυπο διπλής ακρίβειας της IEEE και έτσι μπορεί να αναπαραστήσει πολύ περισσότερους αριθμούς με αυτή την ακρίβεια, απ΄ ότι με μονή ακρίβεια. Για να είμαστε πιο συγκεκριμένοι, αν temp_sum το μερικό άθροισμα των k-1 όρων της σειράς, το πρόγραμμά μας θα τερματίσει όταν $(eps(temp\_sum)/2) \geq (1/k)$. Κι αυτό διότι τότε το κριτήριο θα μηδενιστεί, καθώς η MATLAB θα στρογγυλοποιήσει το αποτέλεσμα της πράξης προς το ήδη υπολογισμένο μερικό άθροισμα. Όλα αυτά, φυσικά, αν αποφευχθεί το σφάλμα της μνήμης.

iv) Ο κώδικας για το παρόν ερώτημα είναι ο ίδιος με παραπάνω, ελαφρά τροποποιημένος ώστε να γίνονται οι πράξεις σε μονή ακρίβεια, όπως μας ζητείται:

```
1  function [] = script_2_2_iv
2  % Variable initialization.
3  summary = 0; temp_sum = 0;
4  n = single(1); run = true;
5  % Infinite loop til criterion's fulfilled.
6  while run
7      % Temporary summary.
8      temp_sum = single(summary) + single(1/n);
9      % Criterion.
10     if abs(temp_sum - summary) == 0
11         run = false;
12     end
13     % Displaying current difference.
14     disp(abs(temp_sum - summary))
15     summary = temp_sum; % Current summary.
16     n = n + 1; % Increment for next division.
17 end
```

files/2/iv/script_2_2_iv.m

Λόγω ομοιότητας του κώδικα, γνωρίζουμε και πάλι ότι θα τερματίσει, όπως και ο προηγούμενος με τη διπλή ακρίβεια. Αυτή τη φορά, όμως, κατά την εκτέλεση του κώδικα, παρατηρούμε ότι τερματίζει σχετικά σύντομα, μέσα σε λίγα λεπτά. Συγκεκριμένα, τερματίζει έπειτα από 2097152 επαναλήψεις, έχοντας φτάσει στο μερικό άθροισμα *summary*, το οποίο είναι περίπου ίσο με 15.4036827 και για το οποίο ισχύει ότι *eps(summary)* $= 9.5367432 \cdot 10^{-7}$.

Αν εξετάσουμε το λόγο 1/2097152 θα δούμε ότι ισούται με το eps(summary)/2, και συνεπώς, επειδή το summary έχει μηδενικό τελευταίο ψηφίο αναπαράστασης (σε δεκαεξαδική μορφή ισούται με 4176757c, όπου το c σε δυαδική μορφή ισούται με 1100), η MATLAB στρογγυλοποιεί το αποτελέσμα προς αυτό ως τον πλησιέστερο αριθμό με μηδενικό τελευταίο bit. Τώρα πλέον το κριτήριο ισούται με μηδέν και ικανοποιείται η συνθήκη μας, τερματίζοντας την εκτέλεση του προγράμματός μας.

ΕΡΩΤΗΜΑ 3 - Πράξεις με Πολυώνυμα

Αρχικά, παρατίθενται οι πληροφορίες που ζητούνται στην εκφώνηση για τις συναρτήσεις της MATLAB που αφορούν στη διαχείριση πολυωνύμων και τις οποίες χρησιμοποιούμε στη συνέχεια για τα υπόλοιπα υποερωτήματα.

poly

Η συνάρτηση αυτή χρησιμοποιείται για τη μετατροπή ριζών σε πολυώνυμο. Δηλαδή, δεδομένου ενός διανύσματος που περιέχει τις ρίζες του πολυωνύμου, η συνάρτηση επιστρέφει ένα αντίστοιχο διάνυσμα με τους συντελεστές του πολυωνύμου που έχει αυτές τις ρίζες, κατά φθίνουσα σειρά ως προς τη δύναμη της μεταβλητής του πολυωνύμου.

Κοιτώντας λίγο πιο «βαθιά», στον κώδικα της συνάρτησης, βλέπουμε ότι χρησιμοποιεί αναδρομή για τον υπολογισμό των ζητούμενων συντελεστών, αφού πρώτα διακρίνει αν το εισαχθέν όρισμα είναι μητρώο ή διάνυσμα, και επιστρέφει ένα πραγματικό αποτέλεσμα.

roots

Η συνάρτηση αυτή χρησιμοποιείται για την εύρεση των ριζών ενός πολυωνύμου. Δηλαδή, δέχεται ως είσοδο ένα διάνυσμα με τους συντελεστές του πολυωνύμου σε φθίνουσα σειρά ως προς τη δύναμη, όπως και πριν, και επιστρέφει τις ρίζες του πολυωνύμου αυτού.

Εξετάζοντας τον κώδικα της συνάρτησης αυτής, βλέπουμε ότι αρχικά εξαλείφει τα μηδενικά που «οδηγούν» (leading zeros) στο διάνυσμα εισόδου καθώς και τα μηδενικά που «ακολουθούν» (trailing zeros), έπειτα ότι απομακρύνει μικρούς συντελεστές που πιθανόν να οδηγήσουν σε άπειρο (Inf) για αποτέλεσμα, και, τέλος, ότι υπολογίζει τις ρίζες μέσω του συνοδού μητρώου[4] και των ιδιοτιμών του. Αυτή η μέθοδος υπολογισμού των ριζών είναι η καλύτερη δυνατή, σύμφωνα και με τα λεγόμενα του ίδιου του Moler.[5]

polyval

Η συνάρτηση αυτή χρησιμοποιείται για τον υπολογισμό της τιμής ενός πολυωνύμου. Πιο αναλυτικά, η τυπική της μορφή δέχεται ως ορίσματα ένα διάνυσμα p, που περιλαμβάνει τους συντελεστές του πολυωνύμου κατά φθίνουσα σειρά, και ένα διάνυσμα ή μητρώο x, που περιέχει τις τιμές της μεταβλητής για τις οποίες ψάχνουμε την τιμή του πολυωνύμου.

Στον κώδικα της συγκεκριμένης συνάρτησης, εκτός από τους αρχικούς ελέγχους, βλέπουμε ότι στη γενική περίπτωση γίνεται χρήση της μεθόδου Horner για τον υπολογισμό της ζητούμενης τιμής του πολυωνύμου. Εκτός αυτού, στη συνέχεια καθορίζεται η μέθοδος υπολογισμού για τις υπόλοιπες περιπτώσεις, δηλαδή όταν έχουμε επιστροφή περισσότερων ορισμάτων ή επιπλέον ορίσματα εισόδου (όπως π.χ. το αποτέλεσμα της συνάρτησης polyfit).

[4] G. Strang, «Εισαγωγή στη Γραμμική Άλγεβρα», σ. 714

[5] http://www.mathworks.com/company/newsletters/articles/roots-of-polynomials-that-is.html

Μέρος Α΄

(Α) Ο κώδικας που υλοποιεί όλα τα ζητούμενα του υποερωτήματος είναι ο εξής:

```
% Creating a text file for the results.
file_id = fopen('2_3_a_a.txt','w');
% Finding polynomial coefficients from given roots.
z = [1:20]; p = poly(z);
% Printing initial polynomial coefficients.
fprintf(file_id, '-------------------------------------------------\n');
fprintf(file_id, '||||||||| INITIAL COEFFICIENTS ||||||||||\n');
fprintf(file_id, '-------------------------------------------------\n');
fprintf(file_id, '-----[ p(1,:) ]----#\n');
fprintf(file_id, '%3.12e\n', p);
% Specifying required disturbances.
e = [10^(-8) (-10^3) 10^10]; er = p*(2^(-52));
% Preallocating space for later calculations.
g = zeros(5,21); f = zeros(5,21); r = zeros(5,20);
% Calculating coefficients of disturbance polynomials.
g(1,(21-19)) = 1;
g(2,(21-11)) = 1;
g(3,(21-1)) = 1;
g(4,2:21) = 1;
g(5,2:21) = 1;
% Calculating final coefficients.
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(1);
f(5,:) = p + g(5,:).*er;
% Printing the perturbed coefficients.
fprintf(file_id, '\n');
fprintf(file_id, '-------------------------------------------------\n');
fprintf(file_id, '||||||||| PERTURBED COEFFICIENTS |||||||||\n');
fprintf(file_id, '-------------------------------------------------\n');
for i=1:5
    fprintf(file_id, '-----[ f(%d,:) ]----#\n', i);
    for j=1:21
        fprintf(file_id, '%3.12e\n', f(i,j));
    end
end
% Finding the roots of the new polynomials.
r(1,:) = roots(f(1,:));
r(2,:) = roots(f(2,:));
r(3,:) = roots(f(3,:));
r(4,:) = roots(f(4,:));
r(5,:) = roots(f(5,:));
% Printing the roots of the perturbed polynomials.
fprintf(file_id, '\n');
fprintf(file_id, '-------------------------------------------------\n');
fprintf(file_id, '|||||||||||||| NEW ROOTS ||||||||||||||||\n');
fprintf(file_id, '-------------------------------------------------\n');
for i=1:5
    fprintf(file_id, '-----[ r(%d,:) ]----#\n', i);
    for j=1:20
        fprintf(file_id, '%3.12e + ', r(i,j));
        fprintf(file_id, '(%3.12e)i\n', imag(r(i,j)));
    end
end
% Finding required errors.
```

```
r_e = zeros(5,20); k_r = zeros(5,20);
has_imag_roots = 0; % To check for imaginary roots.
temp_r = zeros(5,20); % Temporarily save roots.
for i=1:5
    % Checking if there are any imaginary roots.
    for j=1:20
        if (imag(r(i,j)) > 0)
            has_imag_roots = 1;
            break;
        end
    end
    % Two kinds of calculations.
    if (has_imag_roots == 1) % 1) For both roots.
        temp_r(i,:) = sort(r(i,:),'descend');
        for j=1:20, r(i,j) = temp_r(i,j); end
        k_r(i,:) = [20:-1:1];
        for j=1:20
            r_e(i,j) = abs(k_r(i,j)-r(i,j))/abs(k_r(i,j));
        end
    else % 2) For real roots.
        for j=1:20
            [dif,pos] = min(abs(z-r(i,j)));
            r_e(i,j) = abs(z(pos)-r(i,j))/abs(z(pos));
            k_r(i,j) = z(pos);
        end
    end
    % Setting flag to zero.
    has_imag_roots = 0;
end
% Sorting roots based on previous errors.
s_r = zeros(5,20); s_r_e = zeros(5,20);
s_k_r = zeros(5,20); ix = zeros(5,20);
for i=1:5
    [s_r_e(i,:),ix(i,:)] = sort(r_e(i,:),'descend');
    s_r(i,:) = r(i,ix(i,:));
    s_k_r(i,:) = k_r(i,ix(i,:));
end
% Printing sorted roots, known roots and their relative error.
fprintf(file_id, '\n');
fprintf(file_id, '-------------------------------------------------------\n');
fprintf(file_id, '|| SORTED ROOTS || KNOWN ROOTS || ERROR ||\n');
fprintf(file_id, '-------------------------------------------------------\n');
for i=1:5
    fprintf(file_id, '---[ s_r, s_k_r, s_r_e (%d) ]-----#\n', i);
    for j=1:20
        fprintf(file_id, '%3.3e + ', s_r(i,j));
        fprintf(file_id, '(%3.3e)i  ||  ', imag(s_r(i,j)));
        fprintf(file_id, '%d  ||  ', s_k_r(i,j));
        fprintf(file_id, '%3.3e\n', s_r_e(i,j));
    end
end
% Closing text file with results.
fclose(file_id);
% Plotting known and found roots.
figure;
plot(real(s_r(1,:)), imag(s_r(1,:)),'bo','MarkerSize',6); hold on;
plot(real(s_k_r(1,:)), imag(s_k_r(1,:)),'r*','MarkerSize',5);
hold off; legend('found','known','Location','Best'); grid on;
title('Roots (#1)'); xlabel('real'); ylabel('imag');
figure;
plot(real(s_r(2,:)), imag(s_r(2,:)),'bo','MarkerSize',6); hold on;
plot(real(s_k_r(2,:)), imag(s_k_r(2,:)),'r*','MarkerSize',5);
```

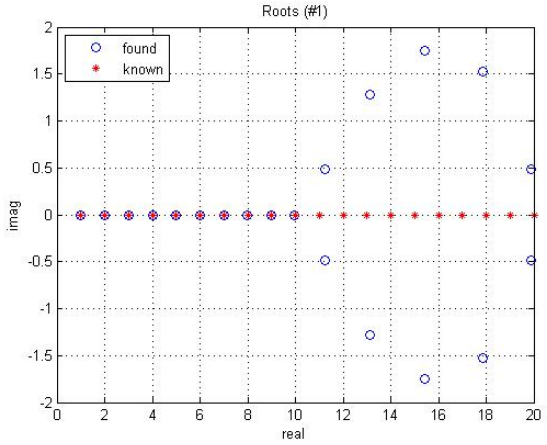
```
119 hold off; legend('found','known','Location','Best'); grid on;
120 title('Roots (#2)'); xlabel('real'); ylabel('imag');
121 figure;
122 plot(real(s_r(3,:)), imag(s_r(3,:)),'bo','MarkerSize',6); hold on;
123 plot(real(s_k_r(3,:)), imag(s_k_r(3,:)),'r*','MarkerSize',5);
124 hold off; legend('found','known','Location','Best'); grid on;
125 title('Roots (#3)'); xlabel('real'); ylabel('imag');
126 figure;
127 plot(real(s_r(4,:)), imag(s_r(4,:)),'bo','MarkerSize',6); hold on;
128 plot(real(s_k_r(4,:)), imag(s_k_r(4,:)),'r*','MarkerSize',5);
129 hold off; legend('found','known','Location','Best'); grid on;
130 title('Roots (#4)'); xlabel('real'); ylabel('imag');
131 figure;
132 plot(real(s_r(5,:)), imag(s_r(5,:)),'bo','MarkerSize',6); hold on;
133 plot(real(s_k_r(5,:)), imag(s_k_r(5,:)),'r*','MarkerSize',5);
134 hold off; legend('found','known','Location','Best'); grid on;
135 title('Roots (#5)'); xlabel('real'); ylabel('imag');
```

files/3/a/a/script_2_3_a_a.m

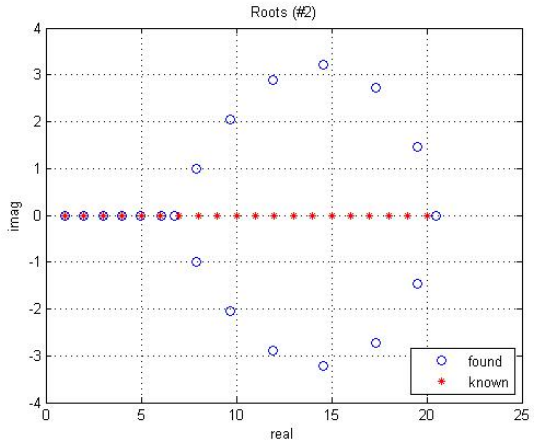
Τα αποτελέσματα των υποερωτημάτων, όπως οι συντελεστές των πολυωνύμων που προκύπτουν καθώς και του αρχικού, οι ρίζες τους, κτλ., βρίσκονται όλα στο αρχείο κειμένου *files/3/a/a/2_3_a_a.txt*. Επιλέχθηκε η εκτύπωση των δεδομένων αυτών σε αρχείο μέσω της εκτέλεσης του προγράμματος ως ένας πιο ακριβής, σύντομος και λιγότερο χρονοβόρος τρόπος καταγραφής των αποτελεσμάτων που ζητούνται.

Πιο συγκεκριμένα, η δομή του αρχείου κειμένου είναι η ακόλουθη. Αρχικά, παρατίθενται οι συντελεστές του πολυωνύμου p_{20}, σε φθίνουσα σειρά. Έπειτα οι διαταραγμένοι συντελεστές του πολυωνύμου αυτού ως συντελεστές των πέντε πολυωνύμων f_{20}, κι αυτοί σε φθίνουσα σειρά, και στη συνέχεια οι ρίζες τους σε μιγαδική αναπαράσταση. Τέλος, παρατίθενται τα αποτελέσματα μετά από την ταξινόμηση των ριζών αυτών βάσει του σχετικού σφάλματος με την πλησιέστερη σε κάθε μία ρίζα, δηλαδή παρατίθεται ένας «πίνακας» με στήλες την ρίζα, την πλησιέστερη σε αυτή γνωστή ρίζα και το μεταξύ τους σχετικό σφάλμα.

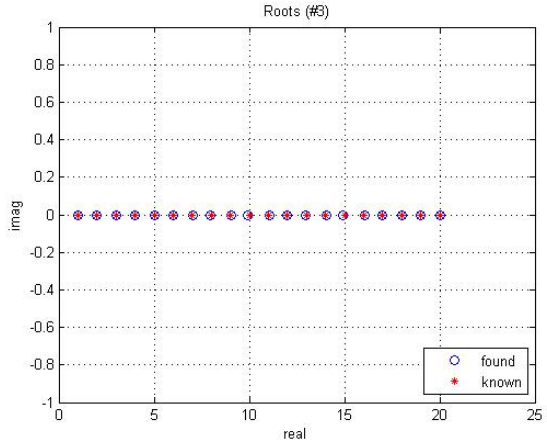
(B) Οι γραφικές παραστάσεις των ριζών που προέκυψαν έπειτα από την εκτέλεση του παραπάνω script είναι οι ακόλουθες:



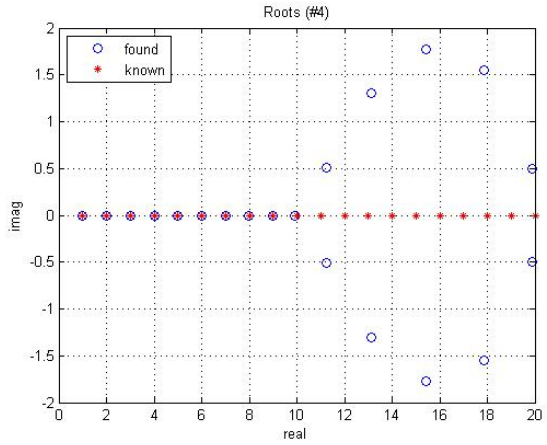
Roots (#1): *οι ρίζες για το πολυώνυμο με διαταραχή 10^{-8} στο συντελεστή της $19^{ης}$ δύναμης.*



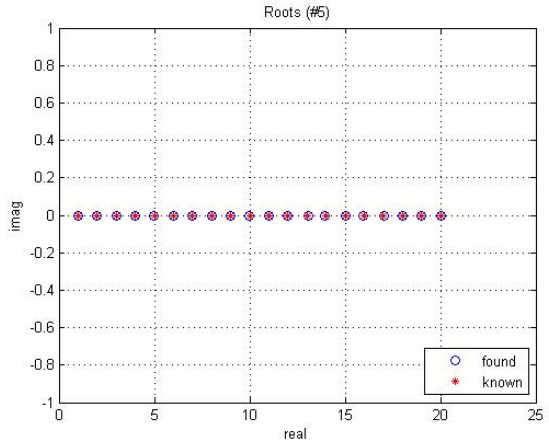
Roots (#2): *οι ρίζες για το πολυώνυμο με διαταραχή -10^{-3} στο συντελεστή της $11^{ης}$ δύναμης.*



Roots (#3): *οι ρίζες για το πολυώνυμο με διαταραχή 10^{10} στο συντελεστή της $1^{ης}$ δύναμης.*



Roots (#4): *οι ρίζες για το πολυώνυμο με διαταραχή 10^{-8} όλων των συντελεστών.*



Roots (#5): *οι ρίζες για το πολυώνυμο με διαταραχή $2^{-52}\cdot$(συντ.) όλων των συντελεστών.*

Σαν ένα πρώτο σχόλιο μπορούμε να πούμε ότι οι μεγαλύτερες διαταραχές των ριζών παρατηρούνται έπειτα από την πρώτη διαταραχή των συντελεστών, την δεύτερη και την τέταρτη, ενώ ελάχιστες έως καθόλου παρατηρούνται μετά από την τρίτη και πέμπτη διαταραχή. Επίσης, στις πρωτοαναφερθείσες περιπτώσεις το μέγεθος της διαταραχής είναι τέτοιο που έχουμε ως αποτέλεσμα ακόμα και μιγαδικές ρίζες.

Αν εξετάσουμε λίγο βαθύτερα τις αιτίες των διαφορών μεταξύ των διαταραχών, θα δούμε ότι οι μεγαλύτερες διαταραχές προκύπτουν όταν αλλάζει ο συντελεστής μεγάλης σχετικά δύναμης κατά ένα σχετικά αντιληπτό ποσοστό. Παραδείγματος χάριν, στην πρώτη διαταράσσεται ο συντελεστής του x^{19}, που ισούται με -210, κατά 10^{-8}, και γίνεται περίπου -209.999. Αντίστοιχα, στην δεύτερη ο συντελεστής του x^{11} κατά -10^3, και από $-1.355851828995 \times 10^{14}$ γίνεται $-1.355851829005 \times 10^{14}$. Στην τέταρτη περίπτωση διαταραχών, επηρεάζονται όλοι οι συντελεστές εκτός του πρώτου, αλλά λόγω του μεγέθους τους, άλλοι επηρεάζονται λιγότερο κι άλλοι περισσότερο. Έτσι, μιας και η διαταραχή είναι ίδια με αυτή της πρώτης περίπτωσης, σίγουρα επηρεάζονται οι συντελεστές των μεγαλύτερων δυνάμεων της μεταβλητής, ενώ οι υπόλοιποι, όσο φθίνει η τάξη της δύναμης, δεν επηρεάζονται τόσο. Γι΄ αυτό, άλλωστε, μοιάζουν λίγο οι γραφικές παραστάσεις της πρώτης και τέταρτης περίπτωσης.

Και για τον ίδιο λόγο η τρίτη διαταραχή, η οποία γίνεται στο συντελεστή του x^1 και είναι της τάξης του 10^{10} δεν έχει ιδιαίτερο αποτέλεσμα στις νέες ρίζες. Βέβαια, στην τελευταία περίπτωση, αυτό που συμβαίνει είναι κάπως διαφορετικό, μιας και διαταράσσουμε όλους (πλην του πρώτου πάλι) τους συντελεστές κατά ένα μικρό ποσοστό του «εαυτού» τους, και συγκεκριμένα κατά 2^{-52}, επηρεάζοντας έτσι μονάχα το τελευταίο bit τους κυρίως. Έτσι, η συνολική διαταραχή είναι πολύ μικρότερη έως μηδαμινη, και για το λόγο που αναφέραμε προηγουμένως.

Αν μελετήσουμε λίγο και τα αποτελέσματα των σχετικών σφαλμάτων και των αντίστοιχων ριζών μετά την ταξινόμησή τους, θα παρατηρήσουμε ότι τα μεγαλύτερα σφάλματα εμφανίζονται στις ρίζες μεγαλύτερου μεγέθους, σε γενικές γραμμές. Π.χ. στις τέσσερις από τις πέντε περιπτώσεις διαταραχής, τα μικρότερα σφάλματα εμφανίζονται για τις γνωστές ρίζες από 1 μέχρι 6, ενώ μετά ποικίλει η σειρά τους ανάλογα το σφάλμα. Πέρα από τα παραπάνω, αυτό φαίνεται και στις γραφικές

παραστάσεις που προέκυψαν.

Τώρα, εκτός από τα αποτελέσματα του παρόντος ερωτήματος, μπορούμε να δούμε και τους πειραματισμούς του J. Wilkinson για να κατανοήσουμε το πόσο μπορεί να επηρεάσει τις ρίζες του συγκεκριμένου πολυωνύμου μια αλλαγή σε συντελεστή μεγάλης δύναμης. Ο Wilkinson, λοιπόν, όπως κι εμείς στην αρχική μας περίπτωση, είχε αλλάξει τον συντελεστή του x^{19}, -210, κατά -2^{-23}, και είχε διαταραχές αντίστοιχες με αυτές που μας προέκυψαν. Η διαταραχή[6] που επέλεξε ο ίδιος είχε να κάνει με το σύστημα στο οποίο έκανε τους υπολογισμούς και τον τρόπο που αναπαριστούσε αυτό τους αριθμούς. Μια καλή αναπαράσταση και μελέτη[7] της συμπεριφοράς για διάφορες διαταραχές έχει κάνει και ο Moler, όπου φαίνεται και το εύρος των μιγαδικών που προκύπτουν, αλλά και ότι από τις διαταραχές επηρεάζονται περίπου οι ρίζες που είναι μεγαλύτερες του 8 ή του 9, κάτι που παρατηρήσαμε και εμείς στους δικούς μας πειραματισμούς προηγουμένως.

Μετά από όλα αυτά, μπορούμε όντως να καταλάβουμε την «ταλαιπωρία» του Wilkinson και να πούμε ότι το υπό εξέταση πολυώνυμο είναι ιδιαίτερα «δόλιο». Και, εξαιτίας αυτής της «δολιότητας» ή αλλιώς «ευαισθησίας», ίσως να αποτέλεσε και ένα από τα πρώτα παραδείγματα που έδωσαν το έναυσμα για τη διατύπωση της θεωρίας του χάους.[8]

Μέρος Β΄

Στο ερώτημα αυτό μας ζητείται να μελετήσουμε την πίσω ευστάθεια του αλγορίθμου Horner και να την χρησιμοποιήσουμε για να φράξουμε το εμπρός σφάλμα του πολυωνύμου $p(x) = \prod_{j=1}^{10}(x-j)$ για τις δοθείσες τιμές. Μπορούμε να βρούμε την απόδειξη της πίσω ευστάθειας του αλγορίθμου στο βιβλίο[9] του μαθήματος, και συγκεκριμένα στις σελίδες 80-81. Έτσι, στην περίπτωσή μας, το n ισούται με 10, η $f(x)$ συνάρτηση είναι η $p(x)$ ως γινόμενο, ενώ η $f_{prog}(x)$ είναι η $p(x)$ ως δυναμομορφή.

Αντίστοιχα, και λαμβάνοντας υπ΄ όψιν τα εργαστήρια και τα φροντιστήρια του μαθήματος, το ζητούμενο σχετικό σφάλμα είναι το εμπρός σφάλμα και ισούται με $\frac{\|f(x)-f_{prog}(x)\|}{\|f(x)\|}$. Επίσης, ο δείκτης κατάστασης της τιμής του πολυωνύμου σε δυναμομορφή στο x ισούται με $cond = \frac{\sum_{k=0}^{n}|a_k|\cdot|x|^k}{|f(x)|}$, όπως διευκρινίστηκε και στο φροντιστήριο.

Τέλος, αν και δεν υπολογίζουμε το πίσω σφάλμα, μπορούμε να υπολογίσουμε ένα φράγμα γι΄ αυτό, το οποίο στην περίπτωσή μας, σύμφωνα πάντα με τη θεωρία που έχουμε διδαχθεί στο μάθημα και που υπάρχει και στο βιβλίο, ισούται με $\gamma_{2n} = \frac{2nu}{1-2nu}$, όπου $u \approx eps = 2.2204 \times 10^{-16}$.

Οπότε, το φράγμα που μας ζητείται για το εμπρός σφάλμα προκύπτει από το γινόμενο του δείκτη κατάστασης και του φράγματος του πίσω σφάλματος.

Έπειτα από την παραπάνω σύντομη θεωρητική ανασκόπηση, παραθέτουμε τον κώδικα MATLAB με τον οποίο έγιναν οι απαιτούμενοι υπολογισμοί:

[6] https://en.wikipedia.org/wiki/Wilkinson%27s_polynomial#Conditioning_of_Wilkinson.27s_polynomial
[7] http://blogs.mathworks.com/cleve/2013/03/04/wilkinsons-polynomials/
[8] https://en.wikipedia.org/wiki/Chaos_theory
[9] Ευ. Γαλλόπουλος, «Επιστημονικός Υπολογισμός Ι», έκδοση του 2008.

```
% Creating a text file for the results.
file_id = fopen('2_3_b.txt','w');
fprintf(file_id, '-------------------------------------------------\n');
fprintf(file_id, '|||||||||||||||| RESULTS ||||||||||||||||||\n');
fprintf(file_id, '-------------------------------------------------\n');
% Finding polynomial coefficients from given roots.
z = [1:10]; p = poly(z);
% Specifying x values as input.
x = [1+eps 1+10^(-8) 1.5 5.5 9+10^(-8) 9+eps(9)];
rd = zeros(6,1); % For values using polynomial form.
rp = zeros(6,1); % For values using product form.
re = zeros(6,1); % For the relative error.
cond_num = zeros(6,1); % For the condition number.
n = length(z); u = eps; % Specifying n and u.
g2n = (2*n*u) / (1-2*n*u); % Barrier of back-error.
barrier = zeros(6,1); % (g2n * cond_num)
for i=1:6 % Calculations.
    rd(i) = polyval(p,x(i)); % Calculating p(x). (1)
    rp(i) = prod(x(i)-[1:10]); % Calculating p(x). (2)
    re(i) = abs(rp(i)-rd(i)) / abs(rp(i)); % Relative error.
    temp_sum = 0;
    for k=1:n+1 % Temporary summary for condition number.
        temp_sum = temp_sum + abs(p(k))*abs(x(i)^(n+1-k));
    end
    cond_num(i) = temp_sum / abs(rp(i)); % Condition number.
    barrier(i) = g2n * cond_num(i); % Barrier of front-error.
    % Printing results to file.
    fprintf(file_id, 'x(%d): %3.15e\n',i,x(i));
    fprintf(file_id, 'rd(%d): %3.15e\n',i,rd(i));
    fprintf(file_id, 'rp(%d): %3.15e\n',i,rp(i));
    fprintf(file_id, 're(%d): %3.15e\n',i,re(i));
    fprintf(file_id, 'cond_num(%d): %3.15e\n',i,cond_num(i));
    fprintf(file_id, 'barrier(%d): %3.15e\n',i,barrier(i));
    fprintf(file_id, '-------------------------------*\n');
end
% Closing text file with results.
fclose(file_id);
```

files/3/b/script_2_3_b.m

Ο κώδικας αυτός, εκτός των υπολογισμών των τιμών που απαιτούνται για τη συμπλήρωση του πίνακα που προσδιορίζεται στην εκφώνηση, αποθηκεύει τα αποτελέσματά του στο αρχείο κειμένου *files/3/b/2_3_b.txt*, όπου παρουσιάζονται με τη σειρά, δηλαδή για κάθε τιμή του διανύσματος x. Παρόλ' αυτά, παραθέτουμε τα αποτελέσματα και στην αναφορά, μιας και ζητείται συγκεκριμένη μορφή πίνακα, με στρογγυλοποίηση και επιστημονική σήμανση όμως, για την καλύτερη εμφάνισή τους. Τα πιο ακριβή αποτελέσματα βρίσκονται στο αρχείο κειμένου, ενώ ο εν λόγω πίνακας είναι ο ακόλουθος:

x(i)	**rd(i)**	**rp(i)**	**re(i)**	**cond_num(i)**	**barrier(i)**
1+eps	0.00e+00	-8.06e-11	1.00e+00	4.95e+17	2.20e+03
$1+10^{-8}$	-3.63e-03	-3.63e-03	1.79e-07	1.10e+10	4.88e-05
1.5	-3.37e+04	-3.37e+04	0.00e+00	3.06e+03	1.36e-11
5.5	-8.72e+02	-8.72e+02	0.00e+00	2.07e+07	9.18e-08
$9+10^{-8}$	-4.01e-04	-4.03e-04	5.26e-03	8.31e+14	3.69e+00
9+eps(9)	3.94e-06	-7.16e-11	5.50e+04	4.68e+21	2.08e+07

Να σημειωθεί ότι υπάρχει αντιστοιχία μεταξύ των θέσεων των στηλών και των ονομάτων των μεταβλητών που έχουν χρησιμοποιηθεί μέσα στο πρόγραμμα, για λόγους κατανόησης. Έτσι, το rd(i) αναφέρεται στα αποτελέσματα της δυναμομορφής, το rp(i) στα αποτελέσματα του γινομένου, το re(i) στο σχετικό σφάλμα, το cond_num(i) στον δείκτη κατάστασης, και το barrier(i) στο γινόμενο που ισούται με το φράγμα για το εμπρός σφάλμα.

Αν θέλαμε να κάνουμε και ένα σύντομο σχολιασμό στα παραπάνω αποτελέσματα, θα λέγαμε ότι για κάθε τιμή από αυτές που έχουμε το σχετικό ή εμπρός σφάλμα είναι όντως μικρότερο από το γινόμενο του δείκτη κατάστασης επί το πίσω σφάλμα, το οποίο αποτελεί το φράγμα του σφάλματος αυτού. Σε κάποιες τιμές πλησιάζει περισσότερο στο φράγμα, σε κάποιες άλλες λιγότερο, όμως πάντα φράσσεται από αυτό, άρα συμφωνεί και με τη θεωρία μας.