ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΟ ΚΕΝΤΡΟ

Σφάλματα στρογγυλοποίησης (Round-off errors)

Α. Σπυρόπουλος

Αθήνα, 2016

Ορολογία

bit (binary digit): δυαδικό ψηφίο. Τα δυαδικά ψηφία είναι το 0 και το 1
1 byte = 8 bits
word: η θεμελιώδης μονάδα σύμφωνα με την οποία εκπροσωπούνται οι πληροφορίες στον υπολογιστή. Αποτελείται από μια σειρά δυαδικών ψηφίων. Οι αριθμοί, στους υπολογιστές, αποθηκεύονται σε ένα ή περισσότερα words. Το μέγεθος ενός word στους σημερινούς υπολογιστές είναι 32 bit ή 64 bit.

Σφάλματα στρογγυλοποίησης (Round-off errors)

Τα σφάλματα στρογγυλοποίησης προκύπτουν επειδή οι ψηφιακοί υπολογιστές δεν μπορούν να αναπαραστήσουν ορισμένους αριθμούς ακριβώς. Τα σφάλματα αυτά είναι σημαντικά στην επίλυση προβλημάτων μηχανικής, γιατί μπορεί να οδηγήσουν σε λανθασμένα αποτελέσματα. Σε ορισμένες περιπτώσεις είναι εύκολο να αναγνωρίσουμε τέτοια σφάλματα, υπάρχουν όμως και αρκετές που η ανίχνευση τους είναι πολύ δύσκολη.

Δύο είναι οι κύριες αιτίες που μπορούμε να πάρουμε από τον υπολογιστή λανθασμένα αποτελέσματα εξαιτίας των σφαλμάτων στρογγυλοποίησης:

1) οι ψηφιακοί υπολογιστές έχουν όρια στο μέγεθος και στην ακρίβεια με την οποία αναπαριστούν τους αριθμούς,
2) ορισμένοι αριθμητικοί αλγόριθμοι είναι πολύ ευαίσθητοι στα σφάλματα στρογγυλοποίησης.

Αναπαράσταση αριθμών στο δεκαδικό (decimal) σύστημα

Ένα σύστημα αναπαράστασης αριθμών είναι απλώς μια σύμβαση για την αναπαράσταση ποσοτήτων. Το σύστημα με το οποίο είμαστε πιο εξοικειωμένοι είναι το δεκαδικό ή βάση-10 (base-10). Η βάση είναι ο αριθμός που χρησιμοποιείται ως αναφορά για την κατασκευή του συστήματος. Το σύστημα βάσης-10 χρησιμοποιεί τα 10 ψηφία 0, 1, 2, 3, 4, 5, 6, 7, 8, και 9 για να αναπαραστήσει τους αριθμούς. Από μόνα τους τα ψηφία αυτά είναι ικανά να αναπαραστήσουν τους αριθμούς από το 0 έως 9. Για μεγαλύτερες ποσότητες, γίνονται συνδυασμοί αυτών των ψηφίων και ανάλογα με τη θέση τους προσδιορίζεται το μέγεθος τους. Το δεξιότερο ψηφίο σε ένα ακέραιο αριθμό αντιπροσωπεύει ένα αριθμό από 0 έως 9. Το δεύτερο ψηφίο από τα δεξιά αναπαριστά έναν αριθμό πολλαπλάσιο του 10. Το τρίτο ψηφίο από τα δεξιά αναπαριστά ένα πολλαπλάσιο του 100 και ούτω καθεξής.

Για παράδειγμα, αν έχουμε τον αριθμό 8642.91, τότε:

$(8 \times 10^3) + (6 \times 10^2) + (4 \times 10^1) + (2 \times 10^0) + (9 \times 10^{-1}) + (1 \times 10^{-2}) = 8642.91$

Η παραπάνω μορφή ονομάζεται **συμβολισμός θέσης**.

Αναπαράσταση αριθμών στο δυαδικό (binary) σύστημα

Στους υπολογιστές η αναπαράσταση των αριθμών γίνεται με τη χρήση του δυαδικού (binary) συστήματος αρίθμησης ή βάση-2 (base-2). Σύμφωνα με το σύστημα αυτό οι αριθμοί αναπαρίστανται με τα δυαδικά ψηφία (bits) 0 και 1. Αυτό σχετίζεται με το γεγονός ότι οι πρωταρχικές λογικές μονάδες των ψηφιακών υπολογιστών ήταν τα on/off ηλεκτρονικά εξαρτήματα.
Όπως με στο δεκαδικό σύστημα, έτσι και στο δυαδικό, ένας αριθμός μπορεί να αναπαρασταθεί με το **συμβολισμό θέσης**.

Για παράδειγμα, ο δυαδικός αριθμός 101.1 ή $(101.1)_2$ ισοδυναμεί με:
$(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) = 4 + 0 + 1 + 0.5 = 5.5$ στο δεκαδικό σύστημα ή $(5.5)_{10}$.

Δυαδική αναπαράσταση ακέραιων στους υπολογιστές

Οι ακέραιοι αριθμοί αποθηκεύονται σύμφωνα με το πρότυπο *2s complement*. Παρακάτω θα παρουσιαστεί μια απλοποιημένη μορφή του προτύπου αυτού που ονομάζεται *signed magnitude*.

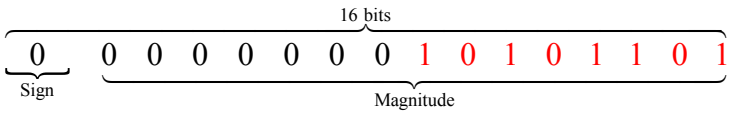
Σύμφωνα με το μοντέλο αυτό το πρώτο bit αναπαριστά το πρόσημο του αριθμού, 0 για θετικό και ένα 1 για αρνητικό, τα υπόλοιπα bits χρησιμοποιούνται για να αποθηκεύσουν τον αριθμό.

<u>Παράδειγμα</u>

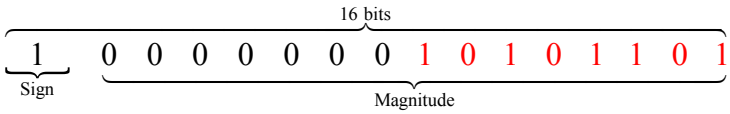
Έστω ο ακέραιος αριθμός 173 που αντιπροσωπεύεται στο δυαδικό ως 10101101.

$(10101101)_2 = 2^7 + 2^5 + 2^3 + 2^2 + 2^0 = 128 + 32 + 8 + 4 + 1 = (173)_{10}$

Σε ένα word με μήκος 16 bits θα αποθηκευτεί ως:



Ο ακέραιος αριθμός -173 θα αποθηκευτεί ως:



Συνεπώς ένα bit χρησιμοποιείται για το πρόσημο και τα υπόλοιπα 15 bits μπορούν να αντιπροσωπεύουν τους δυαδικούς ακέραιους από 000000000000000 έως 111111111111111.

Ο μεγαλύτερος ακέραιος που μπορεί να απεικονιστεί είναι ο

$(111111111111111)_2 = (1 \times 2^{14}) + (1 \times 2^{13}) + \ldots + (1 \times 2^{1}) + (1 \times 2^{0}) = 32\,767$

Σημειώστε ότι $2^{15} - 1 = 32\,767$.

Συνεπώς, ένα word με μήκος 16 bits, μπορεί να αναπαραστήσει τους ακέραιους που κυμαίνονται από -32 767 έως 32 767.

Επιπλέον, επειδή το 0 έχει ήδη οριστεί ως 0000000000000000, είναι περιττό να χρησιμοποιήσουμε τον αριθμό 1000000000000000 για το -0. Έτσι συμβατικά χρησιμοποιείται για να αναπαραστήσει τον αρνητικό αριθμό: -32 768, επεκτείνοντας την περιοχή από -32 768 έως 32 767.

Γενικά ένα word με μήκος n-bit, μπορεί να αναπαραστήσει τους ακέραιους από -2^{n-1} έως $2^{n-1} - 1$.

Για n=32 bit μπορούν να αναπαρασταθούν οι ακέραιοι από
-2 147 483 648 έως 2 147 483 647.

Για n=64 bit μπορούν να αναπαρασταθούν οι ακέραιοι που από -2^{63} έως $2^{63} - 1$.

Αριθμοί πέρα από αυτά τα όρια δεν μπορούν αναπαρασταθούν. Καταλαβαίνουμε λοιπόν πως όλοι οι ψηφιακοί υπολογιστές έχουν περιορισμένη ικανότητα στην αναπαράσταση ακέραιων αριθμών. Μεγαλύτεροι είναι οι περιορισμοί στην αποθήκευση και το χειρισμό των πραγματικών αριθμών.

Για παράδειγμα για να απεικονιστεί ο πραγματικός αριθμός 0.1 στο δυαδικό σύστημα απαιτείται μια άπειρη σειρά:

$$\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \ldots$$

μετά τον πρώτο όρο η ακολουθία 1,0,0,1 επαναλαμβάνεται συνεχώς. Συνεπώς η δυαδική αναπαράσταση του πραγματικού αριθμού 0.1 είναι:

$(0.1)_{10} = (0.0001100110011001100110011001\ldots)_2$

Όπως βλέπουμε απαιτούνται άπειρα bits για την αναπαράσταση του 0.1 σε δυαδική μορφή. Όμως για να αποθηκευτεί ο αριθμός αυτός στον υπολογιστή θα πρέπει να κρατηθούν μόνο μερικά bits ανάλογα με το μήκος του word. Έτσι ο πραγματικός αριθμός 0.1 αποθηκεύεται στον υπολογιστή ως 0.1±ε. Το ε εξαρτάται από τον το μήκος του word. Όσο πιο μεγάλο είναι το μήκος του τόσο μικρότερο είναι το ε και συνεπώς τόσο μεγαλύτερη είναι η ακρίβεια που *προσεγγίζεται* ο αριθμός 0.1.

Δυαδική αναπαράσταση πραγματικών στους υπολογιστές

Για την αναπαράσταση πραγματικών αριθμών στους υπολογιστές χρησιμοποιείται το πρότυπο της κινητής υποδιαστολής (floating point). Σύμφωνα με το πρότυπο αυτό ένας πραγματικός x δίνεται από τη σχέση:

$$x = \pm(1+f)\cdot 2^{e} \qquad (1)$$

όπου

$f = \frac{i}{2^t}$

i όλοι οι ακέραιοι για τους οποίους ισχύει $0 \le f < 1$

t το πλήθος των bits που δεσμεύονται για την αποθήκευση του f

e όλοι οι ακέραιοι στο διάστημα $[e_{\min}\ \ e_{\max}]$

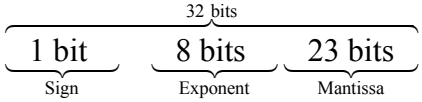
Το e ονομάζεται *exponent* και το 1+f *mantissa*.

Οι παράμετροι t, e_{min} και e_{max} δίνονται στον παρακάτω πίνακα σύμφωνα με το πρότυπο κινητής υποδιαστολής IEEE Standard 754:

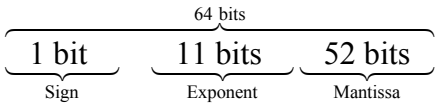
Πίνακας 1: Πρότυπο κινητής υποδιαστολής IEEE Standard 754

	word (bits)	t (bits)	e_{min}	e_{max}
single precision	32	23	-126	127
double precision	64	52	-1022	1023

Σε ένα word με μήκος 32 bits ένας πραγματικός αριθμός θα αποθηκευτεί ως:



και σε ένα word με μήκος 64 bits ως:



Όπως με τους ακέραιους αριθμούς έτσι και με τους πραγματικούς υπάρχουν όρια πέρα από τα οποία δεν μπορούν να αναπαρασταθούν. Τα όρια αυτά καθορίζονται από τα e_{min} και e_{max}. Επιπλέον όμως υπάρχουν και πραγματικοί που αν και βρίσκονται μέσα στα όρια αυτά και πάλι δεν μπορούν να αναπαρασταθούν, ακριβώς, αλλά προσεγγίζονται. Το πόσο καλή είναι η προσέγγιση αυτή εξαρτάται από το πόσο μεγάλο είναι το t.

Παράδειγμα

Υποθέστε ότι για την αναπαράσταση των πραγματικών αριθμών επιλέγονται:

t = 2
e_{min} = - 1 και
e_{max} = 2

σύμφωνα με την σχέση (1) παράγεται ένα σύνολο *διακριτών τιμών,* με συγκεκριμένα όρια, των θετικών πραγματικών αριθμών που μπορούν να αναπαρασταθούν ακριβώς:

0.5000
0.6250
0.7500
0.8750
1.0000
1.2500
1.5000
1.7500
2.0000
2.5000
3.0000
3.5000
4.0000
5.0000
6.0000
7.0000

Θετικοί πραγματικοί αριθμοί μεγαλύτεροι του 7 και μικρότεροι το 0.5 δεν μπορούν να αναπαρασταθούν. Συγκεκριμένα αριθμοί μεγαλύτεροι το 7 προκαλούν *overflow* ενώ μικρότεροι του 0.5 *underflow*.

Τα όρια μπορούν να υπολογιστούν και από τις παρακάτω σχέσεις:

$$\begin{aligned} \text{realmin} &= 2^{e_{\min}} \\ \text{realmax} &= (2-2^{-t})\cdot 2^{e_{\max}} \end{aligned} \qquad (2)$$

Συνεπώς realmin=0.5 και realmax=7.

Επίσης αριθμοί που είναι μεταξύ 2 διακριτών τιμών του παραπάνω συνόλου δεν μπορούν να αναπαρασταθούν ακριβώς και συνεπώς *στρογγυλοποιούνται* στην πλησιέστερη διακριτή τιμή.

Για παράδειγμα ο αριθμός 3.1 θα αποθηκευτεί ως 3. Η διαφορά 3.1 - 3 = 0.1 ονομάζεται σφάλμα στρογγυλοποίησης (*round-off error*)

Η απόσταση μεταξύ του αριθμού 1 και του αμέσως μεγαλύτερου αριθμού (στο παράδειγμα μας είναι ο 1.25) ονομάζεται ακρίβεια του υπολογιστή (*machine epsilon* - συμβολίζεται με τη μεταβλητή *eps*) και μπορεί να υπολογιστεί από τη σχέση:

$$eps = 2^{-t} \qquad (3)$$

Συνεπώς eps=0.25.

Το μέγιστο σφάλμα στρογγυλοποίησης για τους αριθμούς που βρίσκονται μεταξύ του 1 και 1.25 είναι $\frac{eps}{2}$.

Συμπερασματικά, η σχέση (1) παράγει διακριτές τιμές για να προσεγγιστεί το σύνολο των πραγματικών αριθμών. Όσοι πραγματικοί αριθμοί βρίσκονται εκτός των ορίων των τιμών αυτών δεν μπορούν να αναπαρασταθούν. Ενώ όσοι πραγματικοί αριθμοί βρίσκονται ανάμεσα σε 2 διακριτές τιμές θα πρέπει να στρογγυλοποιηθούν στην πλησιέστερη διακριτή τιμή.

Από τη σχέση (2) βλέπουμε ότι οι παράμετροι e_{min} και e_{max} ορίζουν τον μικρότερο και τον μεγαλύτερο πραγματικό αριθμό που μπορεί να αναπαρασταθεί ενώ από τη σχέση (3) βλέπουμε ότι η παράμετρος t ορίζει την ακρίβεια της αναπαράστασης του συνόλου των πραγματικών αριθμών.

Αναπαράσταση των αριθμών στο MATLAB

Ως προεπιλογή (default) του MATLAB είναι να αποθηκεύει όλους τους αριθμούς (συμπεριλαμβανομένων και των ακέραιων) ως αριθμούς κινητής υποδιαστολής διπλής ακρίβειας. Σύμφωνα με το πρότυπο IEEE Standard 754 double precision έχουμε:

t (bits)	e_{min}	e_{max}
52	-1022	1023

Από τις σχέσεις (2) και (3) μπορούμε να υπολογίσουμε τον μικρότερο και τον μεγαλύτερο θετικό πραγματικό αριθμό που μπορεί να απεικονιστεί στο MATLAB καθώς και το machine epsilon.

$$\text{realmin} = 2^{e_{\min}} = 2^{-1022} = 2.2251\text{e}-308$$
$$\text{realmax} = (2-2^{-t})\cdot 2^{e_{\max}} = (2-2^{-52})\cdot 2^{1023} = 1.7977\text{e}+308$$
$$eps = 2^{-52} = 2.2204\text{e}-16$$

Επιπλέον το MATLAB διαθέτει τις συναρτήσεις `realmax, realmin, eps` που τυπώνουν τα όρια και την ακρίβεια των υπολογισμών.

```
>> realmax

ans =

  1.7977e+308
```

Οι αριθμοί που εμφανίζονται στους υπολογισμούς και κατ΄ απόλυτη τιμή υπερβαίνουν την τιμή αυτή δημιουργούν *overflow*. Το MATLAB θέτει τους αριθμούς αυτούς τους ίσους με `inf` ή `-inf` ανάλογα με το αν είναι θετικοί οι αρνητικοί.

```
>> realmin

ans =

  2.2251e-308
```

Οι αριθμοί που κατ' απόλυτη τιμή είναι μικρότεροι από την τιμή αυτή δημιουργούν *underflow*. Το MATLAB θέτει τους αριθμούς αυτούς τους ίσους με το μηδέν.

```
>> eps

ans =

   2.2204e-16
```

Η συνάρτηση *eps* του MATLAB επιστρέφει την απόσταση από το 1.0 έως τον επόμενο μεγαλύτερο αριθμό διπλής ακρίβειας. Επιπλέον η *eps(x)* επιστρέφει την απόσταση από τον αριθμό abs(x) έως το μεγαλύτερο κατ' απόλυτη τιμή αριθμό.

Η τιμή της *eps* σχετίζεται και με το πλήθος των σημαντικών ψηφίων (significant digits) που τυπώνει το MATLAB τις μεταβλητές. Τα 52 bits που χρησιμοποιούνται για το την δυαδική αναπαράσταση του mantissa (Εξ. 2) αντιστοιχούν σε 15 με 16 σημαντικά ψηφία στη δεκαδική αναπαράστασή του.

Μεταβλητές απλής και διπλής ακρίβειας στη FORTRAN 90

Στη FORTRAN 90 οι μεταβλητές απλής ακρίβειας (single precision) δηλώνονται ως:
real *variable_name* ή ισοδύναμα
real*4 *variable_name*
real(KIND=4) :: *variable_name*

οι μεταβλητές διπλής ακρίβειας (double precision) δηλώνονται ως:

real(8) *variable_name* ή ισοδύναμα
real*8 *variable_name*
real(KIND=8) :: *variable_name*

Από τις σχέσεις (2) και (3) και τον Πίνακα 1 μπορούμε να υπολογίσουμε τον μικρότερο και τον μεγαλύτερο θετικό πραγματικό αριθμό που μπορεί να απεικονιστεί στη FORTRAN καθώς και το machine epsilon.

Single precision

$$\text{realmin} = 2^{e_{\min}} = 2^{-126} = 1.1755\text{e-}38$$
$$\text{realmax} = (2 - 2^{-t}) \cdot 2^{e_{\max}} = (2 - 2^{-23}) \cdot 2^{127} = 3.4028\text{e+}38$$
$$eps = 2^{-23} = 1.1921\text{e-}07$$

Double precision

$$\text{realmin} = 2^{e_{\min}} = 2^{-1022} = 2.2251\text{e}-308$$
$$\text{realmax} = (2-2^{-t})\cdot 2^{e_{\max}} = (2-2^{-52})\cdot 2^{1023} = 1.7977\text{e}+308$$
$$eps = 2^{-52} = 2.2204\text{e}-16$$

Οι αριθμοί που εμφανίζονται στους υπολογισμούς και κατ΄ απόλυτη τιμή είναι μεγαλύτεροι από την τιμή **realmax** δημιουργούν ***overflow***. Οι αριθμοί αυτοί τίθενται ίσοι με **`Infinity`** ή **`-Infinity`** ανάλογα με το αν είναι θετικοί οι αρνητικοί αντίστοιχα.

Οι αριθμοί που εμφανίζονται στους υπολογισμούς και κατ΄ απόλυτη τιμή είναι μικρότεροι από την τιμή **realmin** δημιουργούν **underflow**. Οι αριθμοί αυτοί τίθενται ίσοι με το **μηδέν**.

Ποσότητες που δεν μπορούν να αναπαρασταθούν (π.χ. σε ορισμένα αποτελέσματα μαθηματικών συναρτήσεων) τίθενται ίσοι με **NaN** (Not-a-Number).

Η τιμή ***eps*** σχετίζεται και με το πλήθος των σημαντικών ψηφίων (significant digits). Τα 52 bits που χρησιμοποιούνται για το την δυαδική αναπαράσταση των μεταβλητών διπλής ακρίβειας αντιστοιχούν σε 15 σημαντικά ψηφία στη δεκαδική αναπαράστασή του.

Η FORTRAN 90 διαθέτει τις εσωτερικές συναρτήσεις `PRECISION, EPSILON, TINY, HUGE` που τυπώνουν τα όρια και την ακρίβεια των υπολογισμών. Ο παρακάτω κώδικας δείχνει τον τρόπο χρήσης των συναρτήσεων αυτών.

```
PROGRAM get_sizes
IMPLICIT NONE
INTEGER si
INTEGER(8) di
REAL s
REAL(8) d
PRINT *, 'DECIMAL PRECISION'
PRINT *, '-----------------'
PRINT *, 'Single precision:',PRECISION(s)
PRINT *, 'Double precision:',PRECISION(d)
PRINT *, ''
PRINT *, ''
PRINT *, 'Number that is almost negligible compared to one'
PRINT *, '-----------------------------------------------'
PRINT *, 'Single precision:',EPSILON(s)
PRINT *, 'Double precision:',EPSILON(d)
PRINT *, ''
PRINT *, ''
PRINT *, 'SMALLEST POSITIVE NUMBER'
PRINT *, '-----------------------------------------------'
PRINT *, 'Single precision:',TINY(s)
PRINT *, 'Double precision:',TINY(d)
PRINT *, ''
PRINT *, ''
PRINT *, 'LARGEST NUMBER'
PRINT *, '-----------------------------------------------'
PRINT *, 'Single precision:',HUGE(s)
PRINT *, 'Double precision:',HUGE(d)
PRINT *, ''
PRINT *, ''
PRINT *, 'LARGEST INTEGER NUMBER'
PRINT *, '----------------------'
PRINT *, 'Single precision:',HUGE(si)
PRINT *, 'Double precision:',HUGE(di)
END
```

Ο παραπάνω κώδικας όταν εκτελεστεί δίνει:

```
 DECIMAL PRECISION
 -----------------
 Single precision:           6
 Double precision:          15

 Number that is almost negligible compared to one
 ------------------------------------------------
 Single precision:  1.1920929E-07
 Double precision:  2.220446049250313E-016

 SMALLEST POSITIVE NUMBER
 ------------------------------------------------
 Single precision:  1.1754944E-38
 Double precision:  2.225073858507201E-308

 LARGEST NUMBER
 ------------------------------------------------
 Single precision:  3.4028235E+38
 Double precision:  1.797693134862316E+308

 LARGEST INTEGER NUMBER
 ----------------------
 Single precision:  2147483647
 Double precision:  9223372036854775807
```