

ΕΠΙΣΤΗΜΟΝΙΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ (ΑΡΙΘΜΗΤΙΚΗ ΓΡΑΜΜΙΚΗ ΑΛΓΕΒΡΑ)

Διδάσκουσα :

ΕΔΙΠ Μαρία Λουκά.

Τμήμα : Πληροφορικής και Τηλεπικοινωνιών, ΕΚΠΑ

Επικοινωνία με τη Διδάσκουσα:

ΕΔΙΠ : Μαρία Λουκά

Γραφείο : A42

Τηλέφωνο : 210 7275229

email : malouka@di.uoa.gr
malouka@di.uoa.gr

Ώρες γραφείου : Συνάντηση κατόπιν επικοινωνίας με την Διδάσκουσα

Διδασκαλία:

- Η διδασκαλία γίνεται ως διαλέξεις. Κατά τη διάρκεια των διαλέξεων θα επιλύονται και ασκήσεις για την πλήρη κατανόηση της θεωρίας
- Οι διαλέξεις γίνονται κάθε Τρίτη 13:00-15:00 και κάθε Τετάρτη 15:00-17:00 στην αίθουσα Ζ.
- Εξδοχές:
 1. Θεωρία και Υπολογισμοί Μιγρών, G.H. Golub, C.F. Van Loan
Επιστ. Επιμ. Γ. Γκίρης
 2. Αριθμητική Ανάλυση: Μια αλγοριθμική προέγγιση, Ν. Μικυλίου.

Τρόπος αξιολόγησης:

- Η επίδοση των φοιτητών θα αξιολογηθεί μέσα από ένα σύνολο μακ' οίμων εργασιών και τις γραπτές εξετάσεις
- Η μακ' οίμων εργασία θα περιλαμβάνει 2 θεωρητικές και 2 προγραμματιστικές (εργαστηριακές) ασκήσεις
- Η αναλογία ως προς τον τελικό βαθμό είναι η εξής
Τελική εξέταση 70%
Εργασίες 30% (Θεωρητικές 15%, Εργαστηριακές 15%)

Πιο συγκεκριμένα, η τελική βαθμολογία του μαθήματος προκύπτει με βάση τον ακόλουθο τύπο:

Αν $BΓ \geq 30$, τότε

$$TB = (0.15 * \theta A + 0.15 * \epsilon A + 0.7 * BΓ) / 10$$

όπου TB = Τελικός Βαθμός, θA = μέσος όρος βαθμών Θεωρητικών Ασκήσεων και ϵA = μέσος όρος βαθμών Εργαστηριακών Ασκήσεων.

- Οι βαθμοί των ασκήσεων διασπρούνται μόνο για το επόμενο Ακαδημαϊκό έτος. Σε περίπτωση που είναι επιθυμητή η βελτίωση του βαθμού των Ασκήσεων τότε αυτές θα πρέπει να γίνουν όλες από την αρχή.

Επιστημονικοί Υπολογισμοί (Αριθμητική Γραμμική Άλγεβρα)

0. Εισαγωγή:

Η Αριθμητική Γραμμική Άλγεβρα ασχολείται με την αριθμητική επίλυση προβλημάτων που προέρχονται από τη Γραμμική Άλγεβρα. Τα πιο σημαντικά προβλήματα τα οποία εξετάζονται και εμφανίζονται σχεδόν σε όλες τις εφαρμογές των επιστημών και της τεχνολογίας είναι π.χ. η αριθμητική επίλυση γραμμικών συστημάτων, ο προσδιορισμός λύση ελαχίστων τετραγώνων, ο υπολογισμός των ιδιοτιμών και των ιδιότυπων αξιών ενός πίνακα. Επίσης, οι υπολογισμοί πίνακων αποτελούν συχνά ένα βασικό ψήφια (πυρήνας) των προβλημάτων της Υπολογιστικής Επιστήμης και Τεχνολογίας (όπως είναι η επεξεργασία εικόνας και ήχου, η υπολογιστική οικονομία, η προσομοίωση, η Δομική Βιολογία και Βιοπληροφορική, η Μετεωρολογία, η Ρευστοδυναμική κ.α.).

Δεδομένου ότι για την ανάπτυξη οποιαδήποτε αποτελεσματικής αριθμητικής μεθόδου απαιτείται να γνωρίζουμε τον τρόπο παράστασης των αριθμών στον υπολογιστή και τις ιδιαιτερότητες που χαρακτηρίζουν τη πεπερασμένη αριθμητική του, ιδιαίτερη σημασία έχει η μελέτη της αριθμητικής του Η/Υ καθώς και η ανάλυση των σφαλμάτων που υπεισέρχονται σε κάθε αριθμητικό υπολογισμό. Συχνά τα μαθηματικά μοντέλα απαιτούν για την προσομοίωση τους ένα τεράστιο πλήθος υπολογισμών (βυνόμενοι floating point) και γι' αυτό εκτελούνται σε υπερυπολογιστές ή καταμεριζόμενοι υπολογιστές.

Η Αριθμητική Γραμμική Άλγεβρα κυριο λόγο έχει να εφοδιάσει την επιστημονική κοινότητα με αριθμητικά αποτελεσματικούς αλγόριθμους, που αποτελούν τη βάση δημιουργίας αξιόπιστου μαθηματικού λογισμικού, ενώ είναι άρρηκτα συνδεδεμένη με τους Επιστημονικούς Υπολογισμούς (ή Υπολογιστική Επιστήμη). Πράγματι, για την ανάπτυξη καλού λογισμικού απαιτείται μαθηματική κατανόηση του προς λύση προβλήματος, δυνατότητα αλγοριθμικής έκφρασης του και γνώση της αριθμητικής πεπερασμένης ακρίβειας που χρησιμοποιείται από τους υπολογιστές για την επίλυσή του. Η Αριθμητική Γραμμική Άλγεβρα αντιμετωπίζει διεξοδικά όλα αυτά τα θέματα και σε συνάρτηση με τους Επιστημονικούς Υπολογισμούς, δημιουργεί και υλοποιεί υψηλής ποιότητας μαθηματικό λογισμικό για ευρεία χρήση από επιστήμονες και μηχανικούς.

Το μάθημα αυτό παρουσιάζει κάποια βασικά θέματα της Αριθμητικής Γραμμικής Άλγεβρας

Περιεχόμενα:

1. Σφάλματα στους Αριθμητικούς Υπολογισμούς
με αριθμούς σε παράσταση κινητής υποδιαστολής (π.χ. μετάδοση σφάλματος στο εσωτερικό χυόμενο διανυσμάτων)
2. Άμεσοι (ή απ'ευθείας) μέθοδοι για την επίλυση γραμμικών συστημάτων,
με γνωστό πλήθος βυμάτων και σφάλμα σφαιρόμενο μόνο στη εσφαλμένη λύση λόγω του υπολογιστικού μέσου (π.χ. μέθοδος απαλοιφής Gauss)

3. Επαναληπτικές Μέθοδοι για την επίλυση Γραμμικών Συστημάτων
με κριτήριο διαμοιής, αγνώστο πλήθος βυγμάτων και εφάλλα
οφειλόμενο όχι μόνο λόγω της ετροχρήλεσης αλλά και λόγω της
απουσίας του υπολογιστικού τύπου (δηλ του προεχρητισμού τύπου)
4. Αριθμητικός Υπολογισμός Ιδιοτιμών και Ιδιοδιανυσμάτων ενός πίνακα
5. Βασικά στοιχεία αριθμητικής επίλυσης Μητρώων Διαφορικών Εξισώσεων

0.1. Πρόβλημα της Αριθμητικής Γραμμικής Άλγεβρας:

1. Πρόβλημα επίλυσης Γραμμικού Συστήματος

Δίνονται $A^{n \times n}$ μη ιδιάζων (ή αντιστρέψιμος $\Leftrightarrow \det(A) \neq 0$) και
 $b^{n \times 1}$ ένα n -διάστατο διάνυσμα. Να βρεθεί το n -διάστατο
διάνυσμα x , τέτοιο ώστε $Ax=b$

Γενικά: $A^{n \times n} \cdot X^{n \times m} = B^{n \times m}$, $X = (x_1, x_2, \dots, x_n)^T$, $B = (b_1, b_2, \dots, b_m)^T$
όπου $x^T \rightarrow$ διάνυσμα γραμμής.

• Σχετικά προβλήματα

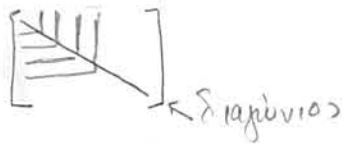
→ Αντιστροφή Πίνακα

→ Ταξη (Rank) πίνακα

δηλ. πλήθος γραμμικά ανεξαρτήτων γραμμών ή στήλών
πχ. ένας πίνακας $A^{3 \times 5}$ έχει Rank το πολύ 3 (ήναυα το ελάχιστο)

→ Έρεση ορίζουσας

→ Κύριες, υπερέχουσες, ελάσσονες ορίζουσες



← Διαγώνιος

→ Ορθογώνια βάση του Range του A :

$$R(A) = \{b \in \mathbb{R}^m \mid b = Ax, x \in \mathbb{R}^n\}, \quad A^{m \times n}, \quad x^{n \times 1}, \quad b^{m \times 1}$$

→ Μηδενικόχωρος (Null space) $N(A) = \{x \in \mathbb{R}^n \mid Ax = 0\}$

→ Διαφοροί πίνακες προβολής του A .

Κύρια Πηγή: Αριθμητική επίλυση Διαφορικών Εξισώσεων

2. Πρόβλημα των ελαχίστων τετραγώνων

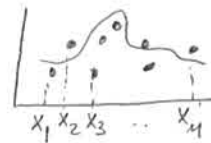
• Χρήση σε ML, AI.

• Δίνονται $A^{m \times n}$ πίνακας, $b^{m \times 1}$ διάνυσμα. Να βρεθεί διάνυσμα $x^{n \times 1}$ τέτοιο ώστε $\|Ax - b\|_2 = r$, όπου $r = \text{υπόλοιπο}$, να είναι ελάχιστο

Σχετικά προβλήματα:

Εφαρμογή στη στατιστική και τη γεωμετρία

$$\varepsilon_1^2 + \varepsilon_2^2 + \dots + \varepsilon_n^2 \rightarrow \text{μήγ} \Leftrightarrow \| \varepsilon \|_2^2 \rightarrow \text{μήγ}$$



3. Πρόβλημα ιδιοτιμών

Δίνεται πίνακας $A^{n \times n}$. Να βρεθούν τα βαθμωτά* λ και μη μηδενικά n διανύσματα x : $Ax = \lambda x$ [πίνακας A επί διάνυσμα x ($O(n^2)$), αριθμός λ επί διάνυσμα x ($O(n)$) $\rightarrow$ μείωση πολυπλοκότητας κατά μία τάξη]

* βαθμωτό μέγεθος = μέγεθος χωρίς διάσταση, δηλ. μονόμετρο
Διανυσματικό μέγεθος: Θέλει μέτρο, διεύθυνση, φορά. Πχ. η ταχύτητα.

Σχετικά προβλήματα

Επίλυση και ανάλυση της ευστάθειας ενός ομογενούς συστήματος 1^{ης} τάξης Διαφορικών εξισώσεων.

Εφαρμοχές:

- Ανάλυση αποθεμάτων αγοράς
- Μελέτη συμπεριφοράς δυναμικών συστημάτων

4. Γενικευμένο Πρόβλημα Ιδιοτιμών

Δίνονται πίνακες $A^{n \times n}$, $B^{n \times n}$ και ζητείται να βρεθούν βαθμωτά (πραγματικά ή μιγαδικά) λ και μη μηδενικά n -διαστάσια διανύσματα:

$$Ax = \lambda Bx$$

Πηγή: Ανάλυση ταλαντώσεως (π.χ. σήματα) ή παλμού χωρίς απόσβεση, η οποία μοντελοποιείται με ένα σύστημα διαφορικών εξισώσεων 2^{ης} τάξης, της μορφής $B \frac{d^2 z}{dt^2} + Az = 0$ (όπου $\frac{dz}{dt^2} \rightarrow$ απομακρυσή)

5. Πρόβλημα διάσπασης (ή διαχωρισμού) σε Ιδιάζουσες Τιμές (Singular Value Decomposition ή διάσπαση SVD)

• Χρήση σε ML, AI

• Δίνεται πίνακας $A^{m \times n}$ και ζητούνται να βρεθούν ορθογώνιοι πίνακες $U^{m \times m}$ και $V^{n \times n}$ (X ορθογώνιος $\Leftrightarrow X^T X = I \Leftrightarrow X^T = X^{-1}$) και διαγώνιος πίνακας $\Sigma^{n \times n}$: $A = U \Sigma V^T$, όπου

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 \\ & \sigma_2 & \\ 0 & & \ddots \\ & & & \sigma_n \end{pmatrix} \text{ και } \sigma_1, \sigma_2, \dots, \sigma_n \text{ οι ιδιάζουσες τιμές του } \Sigma$$

Εφαρμογές

- Θεωρία ελέγχου συστημάτων
- Βιοϊατρική
- Επεξεργασία σήματος και εικόνας
- Στατιστικές εφαρμογές

Μετασχηματισμοί Πινάκων

Γενικά θέλουμε τον μετασχηματισμό του πίνακα A σε από πίνακα (δηλ. διαγώνιο, τριδιαγώνιο, άνω τριγωνικό, κ.α).

Έχουμε δύο κατηγορίες

(I) $Ax = b$ (Ισοδυναμίας)

(II) $Ax = \lambda x$ (Ομοιότητας)

(I) Οι μετασχηματισμοί Ισοδυναμίας δεν επηρεάζουν τη λύση του συστήματος

$Ax = b \Leftrightarrow \dots \Leftrightarrow Ux = c$, δηλ. καταλύω με μετασχηματισμούς

$A_1 = A \rightarrow A_2 \rightarrow A_3 \rightarrow \dots \rightarrow A_n = U$ (ακολουθία πινάκων Gauss) σε

έναν άνω τριγωνικό πίνακα.

(II) Οι μετασχηματισμοί Ομοιότητας χωρίζονται σε δύο υποκατηγορίες

(II₁) Περίστροφες (π. Givens rotations)

(II₂) Ορθογώνιοι (π. Householder)

Οι μετασχηματισμοί αυτοί δεν επηρεάζουν τις ιδιοτιμές λ και τα ιδιοδιανύσματα v ενός πίνακα

0.2. Μαθηματικό λογισμικό για προβλήματα της Αριθμητικής Γραμμικής Άλγεβρας

Το διαθέσιμο λογισμικό για προβλήματα Αριθμητικής Γραμμικής Άλγεβρας είναι πολύ πλούσιο. Αυτή τη στιγμή υπάρχουν διαθέσιμα αρκετά πακέτα υψηλής ποιότητας μαθηματικού λογισμικού (mathematical software) για την επίλυση προβλημάτων που γενικά απαιτούν υπολογισμούς με πίνακες (περίπου το 65% των προβλημάτων ανάγονται σε επίλυση συστημάτων και εύρεση ιδιοτιμών-ιδιοδιανυσμάτων). Αρχικά παραθέτουμε τα πρώτα που αναπτύχθηκαν και είναι τα ακόλουθα:

1. LINPACK (Linear)

Το LINPACK είναι μία συλλογή (ή πακέτο) από υποπρογράμματα στο γλώσσα προγραμματισμού FORTRAN, τα οποία αναλύουν και επιλύουν διάφορα συστήματα γραμμικών αλγεβρικών εξισώσεων. Τα υποπρογράμματα αυτά είναι σχεδιασμένα ώστε να είναι ανεξάρτητα της μηχανής, πλήρως ευρότητα (portable) και μπορούν να τρέχουν αποτελεσματικά, έχοντας περίπου βέλτη αποδοτικότητα στα περισσότερα λειτουργικά περιβάλλοντα.

2. EISPACK (Eigenvalues)

Το EISPACK είναι πακέτο υποπρογραμμάτων για τον υπολογισμό του ιδιοσυστήματος ενός πίνακα.

- Υπολογισμός των ιδιοτιμών και των ιδιοδιανυσμάτων ενός πίνακα.
- Επίλυση του γενικευμένου προβλήματος ιδιοτιμών ($Ax = \lambda Bx$, A, B πραγματικοί)
- Διάσπαση πίνακα σε ιδιάζουσα τιμή (SVD)

3. LAPACK (Linear Algebra)

Το LAPACK είναι μια βιβλιοθήκη από υποπρογράμματα σε FORTRAN 77, είναι ανοιχτό επίσης λογισμικό, για την επίλυση των πιο κοινών προβλημάτων που φφαιούνται στην Αριθμητική Γραμμική Άλγεβρα.

Σχεδιάστηκε για την αντιμετάθεση του λογισμικού των LINPACK και EISPACK με στόχο την πολύ μεγάλη αποδοτικότητα σε διανυσματικούς επεξεργαστές (vector processors), σε υψηλές επιδόσεις σταθμούς εργασίας (workstations) ή σε πολυεπεξεργαστές με κοινά διαμοιραζόμενη μνήμη (shared memory). Το LAPACK δημιουργεί blocks στον αλγόριθμους της Αριθμητικής Γραμμικής Άλγεβρας και χρησιμοποιεί υποπρογράμματα που εστιάζονται στην ευέλεση πράξεων μεταξύ διανυσμάτων και πινάκων.

Έχει 3 επίπεδα BLAS (Basic Linear Algebra Subroutines).

- Επίπεδο 1 (BLAS-1): Για πράξεις διάνυσμα με διάνυσμα (ως μορφής $y = ax + b$, όπου x, y διανύσματα και a, b βαθμωτά μεγέθη)
- Επίπεδο 2 (BLAS-2): Για πράξεις πίνακα με διάνυσμα (ως μορφής $y = aAx + y$, όπου x, y διανύσματα, A πίνακας, a βαθμωτό μέγεθος)
- Επίπεδο 3 (BLAS-3): Για πράξεις πίνακα με πίνακα (ως μορφής $C = aAB + bC$, όπου A, B, C πίνακες και a, b βαθμωτά μεγέθη)

4. NETLIB (Network Library)

Είναι μια βιβλιοθήκη δικτύου με ηλεκτρονικά διαθέσιμα υποπρογράμματα των LINPACK, EISPACK, LAPACK μαζί με κάποιους άλλους τύπους λογισμικού για υπολογισμούς πινάκων)

5. NAG (Numerical Algorithms Group)

Αυτή η ομάδα έχει αναπτύξει μια μεγάλη βιβλιοθήκη λογισμικού που περιέχει προγράμματα για πολλά υπολογιστικά περιβάλλοντα, όπως

- Προβλήματα αριθμητικής γραμμικής άλγεβρας
- Αριθμητικά προβλήματα διαφορικών εξισώσεων (συνήθων και μερικών)
- Προβλήματα βελτιστοποίησης
- Προβλήματα ολοκληρωτικών εξισώσεων
- Στατιστικά προβλήματα, κ.α.

6. IMSL (International Mathematical and Statistical Library)

Η βιβλιοθήκη αυτή περιέχει προγράμματα για σχεδόν όλους τους μαθηματικούς και στατιστικούς υπολογισμούς

7. ACM Library (Association for Computing Machinery)

Η βιβλιοθήκη αυτή περιέχει προγράμματα για βασικές πράξεις πινάκων, γραμμικά συστήματα και σχετικά προβλήματα με μη γραμμικά συστήματα (ρίγες πολυωνύμων κ.α.)

8. ITRACK (Iterative Software Package)

Σχεδιάστηκε για την επίλυση γραμμικού συστήματος $Ax=b$ με διάφορες επαναληπτικές μεθόδους (όπου ο πίνακας A είναι μεγάλος και αραιός).

Πιο σύγχρονα περιβάλλοντα για την επίτευξη επιστημονικών υπολογισμών έχουν αναπτυχθεί και προσφέρουν στο χρήστη μαζί με το πακέτο και γλώσσα προγραμματισμού. Όλα έχουν πλέον ευνοηθεί στην δυνατότητα εκτέλεσης πράξεων μεταξύ διανυσμάτων και πινάκων αυξάνοντας έτσι την ταχύτητα εκτέλεσης αλγορίθμων που γράφονται σε διανυσματική μορφή (vectorized algorithms). Ένα πολύ ενδιαφέρον χαρακτηριστικό αφορά το κατά πόσον τα περιβάλλοντα προσφέρουν ελεύθερη χρήση ή απαιτείται η αγορά άδειας.

Τα εμποδαιότερα περιβάλλοντα εκτέλεσης επιστημονικών υπολογισμών είναι τα ακόλουθα:

1. MATLAB (Matrix Laboratory)

Το MATLAB αποτελεί ένα διαδραστικό υπολογιστικό σύστημα για εύκολους υπολογισμούς όλων των επιστημονικών προβλημάτων που βασίζονται σε πίνακες. Εμφανίστηκε περί τα μέσα της δεκαετίας του 1980 και υπήρξε καινοτόμο για την εποχή του.

Το MATLAB εξασφαλίζει εύκολη προσπέλαση σε λογισμικό πινάκων σχεδιασμένο με το λογισμικό LINPACK και EISPACK.

Διαδέχει διηγή του γλώσσα προγραμματισμού, η οποία μπορεί να

χρησιμοποιηθεί για την κατασκευή αλγορίθμων και προγραμμάτων. Επίσης, διαθέτει δυνατότητες γραφικών και έχει πολλά πακέτα (toolboxes) για τη διαχείριση διάφορων εφαρμογών. Για τη χρήση του απαιτείται αρχικά άδεια. Εναλλακτικά, μπορεί να χρησιμοποιηθεί το OCTAVE το οποίο είναι ένα λογισμικό ανοικτού κώδικα.

• MATLAB codes and MATCOM

Το MATCOM είναι ένα διαλογικό πακέτο εργαλείων λογισμικού προς MATLAB για την υλοποίηση όλων των κυριότερων αλγορίθμων για το ίδιο πρόβλημα.

2. PYTHON

Η PYTHON είναι αντισειμενωγραφής γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί τόσο για επιστημονικές εφαρμογές όσο και για διαχείριση βελίδων στο διαδίκτυο. Αναπτύχθηκε περί το 2012 και συνδυάζει τις δυνατότητες της γλώσσας προγραμματισμού C καθώς και του MATLAB. Διαθέτει το πακέτο NumPy για επεξεργασία αριθμητικών εφαρμογών και είναι εμπλουτισμένη με διάφορες βιβλιοθήκες Python modules

3. JULIA

Η JULIA είναι μία υψηλού επιπέδου γλώσσα προγραμματισμού που αναπτύχθηκε περί το 2012 από ερευνητές του MIT. Ως γενική γλώσσα προγραμματισμού μπορεί να χρησιμοποιηθεί για τη δημιουργία εφαρμογών και ενδείκνυται για επιστημονικούς υπολογισμούς.

Η JULIA συνδυάζει $\left\{ \begin{array}{l} \text{Αριθμητική Ανάλυση} \\ \text{Υπολογισμούς με Πίνακες} \\ \text{Επεξεργασία εφάκτου} \\ \text{Γραφικά} \end{array} \right\}$ σε ένα περιβάλλον στο

οποίο τα προβλήματα και οι λύσεις τους εκφράζονται απευθείας όπως γράφονται στα μαθηματικά χωρίς τη χρήση παραδοσιακού προγραμματισμού. Έχει υψηλή αποδοτικότητα, αναχωρίζει βιβλιοθήκες από άλλες γλώσσες προγραμματισμού και χρησιμοποιεί εντατική αξιολόγηση και συνεχή αναβάθμιση των δυνατοτήτων της. Είναι εμπλουτισμένη με πολλά πακέτα εφαρμογών για υπολογισμούς μηχανής υποδιαστολής, γραμμικής άλγεβρας, γεννήτριες τυχαίων αριθμών και πληθώρα άλλων. Αξίζει να σημειωθεί ότι το 2019 το διακεκριμένο βραβείο

"J. H. Wilkinson Prize for Numerical Software" απονεμήθηκε στους ερευνητές J. Bezanson, S. Karpinski και V.B. Shah, για τη συνεισφορά τους στην ανάπτυξη της γλώσσας προγραμματισμού JULIA.

1. Βασική Αριθμητική Υπολογιστή

1.1: Ανάλυση εφάλματος των αριθμών σε παράσταση με κινητή υποδιαστολή (floating point)

- Κάθε πραγματικός αριθμός x αναπαρίσταται με κινητή υποδιαστολή με την ακόλουθη κανονικοποιημένη επιστρεφόμενη μορφή

$$x = \pm (0.a_1a_2 \dots a_k a_{k+1} \dots)_\beta \cdot \beta^e, a_1 \neq 0, (\beta \text{ άρτιος})$$

Π.χ. $0.003789 = \underbrace{0.3789}_{\bar{x} \rightarrow \text{mantissa}} \times 10^{-2}$

- Υπολογιστική λέξη: πεπερασμένου μήκους με k ψηφία (bits) στη mantissa

$\pm$	mantissa (ψηφιακό τμήμα) $\bar{x}$	e
$0, 1$	k bits	ειδικός

Ο πραγματικός αριθμός x αντικαθίσταται με τον πλησιέστερο αριθμό μηχανής $fl(x)$ με την τεχνική της στρογγύλευσης ως εξής:

$$\begin{array}{c} \begin{array}{|c|c|c|} \hline + & & + \\ \hline x' & \frac{x'+x''}{2} & x'' \\ \hline \end{array} \end{array} \quad \text{όπου } fl(x) = \begin{cases} x', & \text{αν } x < \frac{x'+x''}{2} \\ x'', & \text{αν } x \geq \frac{x'+x''}{2} \end{cases}$$

όπου x', x'' είναι δύο διαδοχικοί αριθμοί μηχανής

Συμβολίζουμε με $fl(x)$ τον αριθμό μηχανής κινητής υποδιαστολής που είναι πλησιέστερος προς τον x . Έχουμε:

$$\text{Απόλυτο σφάλμα: } |x - fl(x)| \leq \frac{1}{2} \beta^{-k+e}$$

$$\text{Απόλυτο σχετικό σφάλμα: } \frac{|x - fl(x)|}{|x|} \leq \frac{1}{2} \beta^{-k+1} = u \quad \left(\begin{array}{l} \text{μονάδα σφάλματος} \\ \text{εροχγύλευσης} \\ \text{ή μονάδα μηχανής.} \end{array} \right)$$

Υπενθυμίζουμε ότι: k = γιγία στη μαντίσα
 e = εκθέτης

Σε περίπτωση αποκοπής έχουμε ότι $\frac{|x - fl(x)|}{|x|} \leq \beta^{-k+1}$.

Παρατηρήσεις:

- Η ακρίβεια που πετυχαίνουμε με την τεχνική της εροχγύλευσης είναι πολύ καλύτερη από εκείνη της αποκοπής αφού το φράγμα του σχετικού σφάλματος είναι το μισό από εκείνο της αποκοπής
- Το σφάλμα στην τεχνική της εροχγύλευσης αλλάζει πρόσημο (πότε είναι θετικό και πότε αρνητικό) με αποτέλεσμα την πιθανή απώλεια του, ενώ στην τεχνική της αποκοπής το πρόσημο στο σφάλμα είναι πάντα θετικό.

$$\begin{aligned} \text{Θέτουμε } \varepsilon &= \frac{fl(x) - x}{x} \Leftrightarrow x\varepsilon = fl(x) - x \Leftrightarrow fl(x) = x + x\varepsilon \Leftrightarrow \\ &\Leftrightarrow fl(x) = x(1 + \varepsilon), \quad \text{όπου } |\varepsilon| \leq u \end{aligned}$$

Συμβολίζουμε με x την πραγματική αριθ., με $1 + \varepsilon$ την διαταραχή και με ε τη μονάδα μηχανής, δηλ. το μικρότερο θετικό αριθμό τέτοιον ώστε $1 + \varepsilon \neq 1$.

Συμπέρασμα: Η εφοχγυλευθείσα τιμή $fl(x)$ είναι μία ελαφριά διατάραξη της θεωρητικής τιμής x .

Παραδείγματα:

(i) $\beta=2, k=23 \Rightarrow u = \frac{1}{2} \beta^{-k+1}$ δηλ. $u = \frac{1}{2} \cdot 2^{23+1} \simeq 1.19 \cdot 10^{-7}$

(ii) $\beta=2, k=52 \Rightarrow u = \frac{1}{2} \cdot 2^{1-52} \simeq 2.22 \cdot 10^{-16}$

Εκτίμηση του u .

Το μοναδιαίο σφάλμα εφοχγύλευσης u προσδιορίζεται από τα χαρακτηριστικά της αριθμητικής κάθε υπολογιστή. Η τιμή του προσδιορίζει επίσης και την "ακρίβεια" της μηχανής (machine epsilon). Είναι χρήσιμο να γνωρίζουμε την τιμή του u της μηχανής δεδομένου ότι σχετίζεται με την ακρίβεια των εκτελούμενων υπολογισμών.

Μπορούμε να υπολογίσουμε μια καλή προσέγγιση του u ως εξής:

1. Αρχίζοντας λοιπόν με μία μεγάλη τιμή του x και ελαττώνοντας την σταδιακά έως ότου το $1+x$ να υπολογιστεί σαν 1 μπορούμε να πετύχουμε μια ικανοποιητική προσέγγιση του u . Επίσης, έτσι μπορούμε να προσδιορίσουμε και το λεγόμενο "μυδέν" της μηχανής, ϵ_{mach} , το οποίο θα είναι το πρώτο x που θα ικανοποιεί $fl(1+x)=1$. Αυτό υλοποιείται με τον παρακάτω αλγόριθμο. Δεδομένου ότι το IEEE 754 Standard έχει βάση αριθμητικώς $\beta=2$, η ελάττωση του x θα υλοποιείται με διαδοχικές διαιρέσεις με το 2.

```
x=1  
while (1+x != 1)  
    x = x/2  
end  
Zmach = x  
n = 2x.
```

Συνολικό αριθμητικό σφάλμα

Το συνολικό αριθμητικό σφάλμα (E_{total}) είναι το άθροισμα του σφάλματος δεροχύλωσης (E_{round}) και του σφάλματος αποκοπής (E_{trunc}), γιατί προκύπτει από δύο διακριτές, διαδοχικές πηγές ατέλειας κατά την επίλυση ενός προβλήματος σε υπολογιστή. Είναι λοιπόν:

$$E_{total} = E_{round} + E_{trunc}.$$

1. Σφάλμα αποκοπής (E_{trunc} - Truncation error)

- Προκύπτει από την απεικόνιση μιας αριθμικής μαθηματικής διαδικασίας (π.χ. άπειρη σειρά Taylor, άπειρη ολοκλήρωση) με μία πεπερασμένη αριθμητική πρόδεσσηση (π.χ. διακοπή της σειράς, χρήση πεπερασμένων διαφορών).
- Σχετίζεται με την ακρίβεια της μεθόδου, συνήθως είναι μια συνάρτηση των παραμέτρων του αλγορίθμου και συχνά μπορεί να θεωρηθεί ότι είναι γνωστό και σταθερό για μια δεδομένη εφαρμογή.
- Υπάρχει, επειδή οι Η/Υ δεν μπορούν να εκτελέσουν άπειρες πράξεις.
- Μειώνεται όσο μικραίνει το βήμα (π.χ. h) της μεθόδου αλλά αυξάνει την υπολογιστική προσπάθεια.

2. Σφάλμα στρογγύλευσης (Round - Round off error)

- Προκύπτει από την περιορισμένη ακρίβεια (πεπερασμένα ψηφία) των Η/Υ στην αναπαράσταση πραγματικών αριθμών (floating-point arithmetic)
- Κάθε αριθμητική πράξη (+, -, *, /) όταν Η/Υ στρογγυλοποιεί το αποτέλεσμα για να χωρέει στη μνήμη (32/64-bit), συσσωρεύοντας σφάλματα
- Αυξάνεται όσο αυξάνονται τα υπολογιστικά βήματα, καθώς τα μικρά σφάλματα κάθε βήματος συσσωρεύονται

3. Διασπορά του αθροίσματος ($E_{total} = E_{trunc} + E_{round}$)

- Το τελικό αριθμητικό αποτέλεσμα (A_{num}) διαφέρει από την αληθινή λύση (L_{true}) επειδή η μέθοδος δεν είναι τέλεια (αποκοπή) και επειδή ο Η/Υ δεν είναι ακριβής (στρογγύλευση)
- Συνολικό σφάλμα = $|L_{true} - A_{num}|$
- Αν προσπαθήσουμε να μειώσουμε το σφάλμα αποκοπής (π.χ. κάνοντας το βήμα ή πολύ μικρό), αναγκάζουμε τον υπολογιστή να κάνει χιλιάδες περισσότερες πράξεις
- Αυτό έχει σαν αποτέλεσμα τεράστια αύξηση του σφάλματος στρογγύλευσης

Συμπέρασμα: Το συνολικό σφάλμα είναι η συνισταμένη αυτών των δύο πηγών. Η αριθμητική ανάλυση στοχεύει στην εύρεση μιας χρυσής τομής (optimal step size) όπου το αθροιστικό $E_{trunc} + E_{round}$ ελαχιστοποιείται.

Κανόνες: Αναδιάταξη των αριθμητικών παραστάσεων ώστε

- Να αποφεύγεται η αφαίρεση δύο παραπλήσιων αριθμών ή η πρόσθεση δύο αριθμών πολύ διαφορετικού μεγέθους
- Να αδροίζονται πρώτα οι μικρότεροι και μετά οι μεγαλύτεροι όροι μιας παράστασης
- Να μειώνεται κατά το δυνατόν ο αριθμός των πράξεων

Αριθμητικά αποτελεσματικοί αλγόριθμοι

Ο αλγόριθμος είναι μια πεπερασμένη ακολουθία από αριθμητικές και λογικές πράξεις που οδηγούν στη λύση των αριθμητικών προβλημάτων λόγω των ιδιοτήτων της αριθμητικής μηχανής υποδιαστολής, ένα υψηλής ποιότητας μαθηματικό λογισμικό πρέπει να αναπτύσσεται από αριθμητικά αποτελεσματικούς (numerically effective) αλγόριθμους. Ένας αλγόριθμος πινάκων (matrix algorithm) είναι αριθμητικά αποτελεσματικός αν έχει τα ακόλουθα χαρακτηριστικά.

1. Γενικού σκοπού (General Purpose): Ο αλγόριθμος πρέπει να λειτουργεί για μια ευρεία κλάση πινάκων
2. Αξιόπιστος (Reliable): Ο αλγόριθμος πρέπει να δίνει διαγνωστικά μηνύματα αφ' ενός μεν όποτε υπάρχει ο κίνδυνος εμφάνισης προβλημάτων εξαιτίας των εφαλμάτων στρογγύλευσης, αφ' ετέρου δε όταν δεν ικανοποιούνται κάποια καθορισμένα κριτήρια σύγκλισης. Υπάρχουν αλγόριθμοι που παράχουν τελείως εσφαλμένα αποτελέσματα χωρίς να εμφανίζουν κανένα διαγνωστικό μήνυμα. Η μέθοδος απαλοιφής Gauss χωρίς οδηγίες είναι ένας τέτοιος μη αξιόπιστος αλγόριθμος.

3. Ευσταθής (Stable): Η τελική λύση του προβλήματος, λόγω των δυναμικών εφαλμάτων ερογχύλευσης που συσσωρεύονται στον αλγόριθμο, αποτελεί την ακριβή λύση ενός ελαφρά διαταραχμένου αρχικού προβλήματος.

4. Αποτελεσματικός (Efficient): Η αποτελεσματικότητα του αλγόριθμου μετράται από το χρόνο που χρειάζεται ο υπολογιστής για την εκτέλεσή του. Ο αριθμός των πράξεων μινιμής υποδιαστολής που χρειάζονται για την εφαρμογή του αλγόριθμου καθορίζει την πολυπλοκότητά του. Όσο μικρότερη η πολυπλοκότητα, τόσο πιο αποτελεσματικός θεωρείται ο αλγόριθμος.

Ορισμός: Ονομάζουμε floating point operation ή flop την πράξη μινιμής υποδιαστολής που συνδιάζει την εκτέλεση μιας πρόσθεσης/αφαίρεσης και ενός πολλαπλασιασμού/διαίρεσης, δηλαδή είναι της μορφής $c = a + t \cdot b$. Αυτός ο ορισμός του flop χρησιμοποιήθηκε αρχικά στο πολύ δημοφιλές παλιό λογισμικό LINPACK.

Ο αριθμός των απαιτούμενων flops που χρειάζεται ένας αλγόριθμος καθορίζει την ταχύτητα εκτέλεσής του και επομένως αποτελεί κριτήριο της αποτελεσματικότητάς του. Η ταχύτητα με την οποία ένας υπολογιστής εκτελεί flops εκφράζεται σε Gflop/sec (giga-flops per second). Συνήθιστες ταχύτητες εμφανίζονται στο εύρος των 1-10 Gflop/sec, δηλ. δισεκατομμύρια flops ανά δευτερόλεπτο.

Ορισμός: Ένας αλγόριθμος που περιλαμβάνει υπολογισμούς με πίνακες $n \times n$, θα καλείται αποτελεσματικός (efficient) αν δεν χρειάζεται περισσότερο από n^3 flops για την εκτέλεσή του. Ορίσουμε τότε την πολυπλοκότητα του αλγορίθμου να είναι της τάξης (Order) του n^3 και την συμβολίζουμε $O(n^3)$

Για την εκτίμηση της πολυπλοκότητας ενός αλγορίθμου γενικά εκτιμούμε την τάξη της μεγαλύτερης δύναμης του n (διάσταση των δεδομένων) που εμφανίζεται στον υπολογισμό του συνολικού αριθμού των απαιτούμενων flops. (Ο ιστορικός μαύνας του Craymer για την επίλυση ενός γραμμικού συστήματος διάστασης n δεν είναι αποτελεσματικός αφού απαιτούνται $O(n!)$ flops για την εκτέλεσή του).

Υπάρχει ένα σημείο που πρέπει να τονίσουμε εδώ. Ένας αλγόριθμος μπορεί να είναι αποτελεσματικός αλλά να είναι ασταθής (unstable). Για παράδειγμα η μέθοδος απλοϊφούς Gauss χωρίς οδηγό απαιτεί $n^3/3$ flops για ένα $n \times n$ πίνακα. Κατά συνέπεια, παρόλο που είναι αποτελεσματική, αποδεικνύεται ότι είναι αναξιόπιστη και ασταθής για έναν αυθαίρετο πίνακα.

5. Οικονομικός σε χρήση μνήμης: Ένας πυκνός πίνακας τάξης n χρειάζεται n^2 θέσεις μνήμης για την αποθήκευση των στοιχείων του. Κατά συνέπεια, αν ένας αλγόριθμος κατά τη διάρκεια της εκτέλεσής του απαιτεί αποθήκευση διαφόρων πινάκων, θα χρειασθεί μεγάλος αριθμός θέσεων μνήμης ακόμα και αν το n μεταβάλλεται.

Παρόλο που η διαφορετική αποδοτικότητα της μνήμης ενός υπολογιστή ανέρχεται σε πολλά gigabytes = 10^9 bytes, είναι εμφανές να δίνουμε ιδιαίτερη προσοχή κατά το σχεδιασμό ενός αλγορίθμου στην οικονομική διαχείριση της μνήμης. Με αναγκαζάρες ενός αλγορίθμου μπορούμε να πετύχουμε ελάττωση στις αποδοτικότερες απαιτήσεις με αποτέλεσμα την ταχύτερη εκτέλεσή του. Γενικά, αν ένας πίνακας του παράχεται κατά την εκτέλεση του αλγορίθμου δεν πρόκειται να χρησιμοποιηθεί μελλοντικά, θα πρέπει να επιαλυφθεί από άλλα υπολογιζόμενα στοιχεία.

Ένας $n \times n$ πίνακας απαιτεί για την αποθήκευσή του n^2 θέσεις μνήμης. Στην Αριθμητική Γραμμική Άλγεβρα συχνά αντιμετωπίζουμε άνω ή κάτω τριγωνικούς πίνακες οι οποίοι απαιτούν $\frac{n(n+1)}{2}$ αντί για n^2 θέσεις μνήμης. Στην περίπτωση αυτή οι επιπλέον θέσεις μπορούν να χρησιμοποιηθούν για αποθήκευση άλλων στοιχείων. Ένας $n \times n$ τριδιαγώνιος πίνακας απαιτεί μόνο $3n-2$ θέσεις μνήμης για την καταχώριση των στοιχείων του.

Ο συμβολισμός $a \equiv b$ θα σημαίνει ότι το b επιαλύεται το a .

Ανάλυση πολυπλοκότητας και μνήμης βασικών αλγορίθμων

Είναι πολύ σημαντικό να μπορούμε να εκτιμούμε την απαιτούμενη πολυπλοκότητα ενός αλγορίθμου καθώς και τις απαιτούμενες θέσεις μνήμης που χρειάζεται για την εκτέλεσή του. Στη συνέχεια αναλύουμε την πολυπλοκότητα και τη μνήμη που χρειάζονται ορισμένοι πολύ βασικοί αλγόριθμοι, οι οποίοι αποτελούν τον πυρήνα πολλών άλλων συνθετότερων υπολογισμών.

Σημείωση: Για τον υπολογισμό της πολυπλοκότητας είναι απαραίτητοι οι ακόλουθοι τύποι:

$$1+2+3+\dots+n = \frac{n(n+1)}{2}$$

$$1^2+2^2+3^2+\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$$

Παράδειγμα 1: Υπολογισμός εσωτερικού γινομένου (inner product, ip)

Έστω $x, y \in \mathbb{R}^n$. Να δοθεί αλγόριθμος που υπολογίζει το εσωτερικό του γινομένου $(x, y) = x^T y = \sum_{i=1}^n x_i y_i = z \in \mathbb{R}$.

Αλγόριθμος ip

$z = 0$

for $i = 1, 2, \dots, n$

$z = z + x_i y_i$

} Για κάθε i απαιτείται 1 flop, έτσι ο αλγόριθμος για την εκτέλεσή του χρειάζεται n flops (γραμμική πολυπλοκότητα).

Πολυπλοκότητα: $O(n)$ flops

Μνήμη: $2n+1$ θέσεις (n θέσεις για το x , n θέσεις για το y και 1 θέση για το άθροισμα).

Παράδειγμα 2: Υπολογισμός γινομένου διανύσματος με πίνακα
(matrix vector product, mvp)

Έστω A ένας $m \times n$ πίνακας και $x \in \mathbb{R}^n$. Να δοθεί αλγόριθμος που υπολογίζει το γινόμενο $y = Ax$.

Αλγόριθμος mvp:
$$\begin{cases} \text{for } i=1, 2, \dots, m \\ y_i = \sum_{k=1}^n a_{ik} x_k \end{cases}$$
 Για τον υπολογισμό κάθε y_i απαιτούνται n flops. Αφού το i τρέχει από 1 έως m , ο συνολικός αριθμός πολλαπλασιασμών που απαιτούνται είναι $m \cdot n$. Αν $m=n$ ο

αλγόριθμος για την εκτέλεση του χρειάζεται n^2 flops (τετραγωνική πολυπλοκότητα)

Πολυπλοκότητα: $O(mn)$ flops

Μνήμη: $m+n+2n$ θέσεις

Παρατήρηση: Οι υπολογισμοί εσωτερικών γινομένων και γινομένων διανυσμάτων πίνακων αποτελούν τον πυρήνα σχεδόν κάθε αλγορίθμου Αριθμητικής Γραμμικής Άλγεβρας. Κατά συνέπεια, πολλές φορές επιφέρεται η αποτελεσματικότητα ενός αλγορίθμου από το συνολικό αριθμό των απαιτούμενων ip's και mvp's.

- 1967 -

Παράδειγμα 3: Υπολογισμός γινομένου δύο τριγωνικών πινάκων

Έστω $U=(u_{ij})$ και $V=(v_{ij})$ δύο άνω τριγωνικοί πίνακες τάξης n .

Να δοθεί αλγόριθμος που υπολογίζει το γινόμενο $C=UV$, επιμαρτυρώντας τον πίνακα U με το γινόμενο UV .

Αλγόριθμος uppertriang:

for $i=1, 2, \dots, n$

for $j=i, i+1, \dots, n$

$$u_{ij} = c_{ij} = \sum_{k=i}^j u_{ik} v_{kj}$$

Ο υπολογισμός των c_{ij} απαιτεί $j-i+1$ flops

(Πραγματι, πχ αν $i=2, j=5$ έχουμε

$$c_{25} = \sum_{k=2}^5 u_{2k} v_{k5} =$$

$$= u_{22} v_{25} + u_{23} v_{35} + u_{24} v_{45} + u_{25} v_{55}$$

δηλ 4 πολλαπλασιασμοί $= 5-2+1 = j-i+1$)

Αφού το j τρέχει από το i ως το n και το i από το 1 ως το n , ο συνολικός αριθμός των flops είναι:

$$\sum_{i=1}^n \sum_{j=i}^n (j-i+1) = \sum_{i=1}^n (1+2+\dots+(n-i+1)) = \sum_{i=1}^n \frac{(n-i+1)(n-i+2)}{2} \approx \frac{n^3}{6}$$

Επομένως, το γινόμενο δύο τριγωνικών πινάκων απαιτεί $n^3/6$ flops

Πομπηλουμετα : $O(n^3/6)$ flops

Μνημη : $n(n+1)$ (γιατί το πλήθος των μη μηδενικών στοιχείων που βρίσκονται πάνω από την κύρια διαγώνιο σε κάθε πίνακα είναι $n(n+1)/2$.
Πχ για $n=3$ είναι $(3 \cdot 4)/2 = 6$ στοιχεία.)

Παράδειγμα 4: Υπολογισμός γινομένου πινάκων (matrix product)

Έστω A ένας $m \times n$ πίνακας και B ένας $n \times p$ πίνακας. Ο ακόλουθος αλγόριθμος υπολογίζει το γινόμενο $C = AB$.

Αλγόριθμος matprod:

for $i = 1, 2, \dots, m$

for $j = 1, 2, \dots, p$

$$C_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

Για τον υπολογισμό κάθε C_{ij} απαιτούνται n flops. Αφού το j τρέχει από 1 έως p και το i από το 1 έως m , ο συνολικός αριθμός πολλαπλασιασμών που απαιτούνται είναι mnp .

Πολυπλοκότητα: $O(mnp)$ flops

Μνήμη: $m + n + p$ θέσεις

Εάν έχουμε δύο τετραγωνικούς πίνακες τάξης n τότε απαιτούνται συνολικά n^3 flops. Παρατηρούμε λοιπόν ότι το γινόμενο πινάκων απαιτεί τη μέγιστη επιτρεπτή πολυπλοκότητα, επομένως θεωρείται πολύ δαπανηρή πράξη και θα πρέπει να χρησιμοποιείται με φειδώ.

Όσον αφορά τις απαιτούμενες θέσεις μνήμης, κάθε πίνακας χρειάζεται n^2 θέσεις. Για οικονομία στις απαιτούμενες θέσεις μνήμης μπορούμε να επικαλύπτουμε τον B με το γινόμενο AB , υποθέτοντας ότι μία επιπλέον βελάνη έχει προτεθεί στον B και η οποία θα χρησιμοποιήσει ως προσωρινό διάνυσμα εργασίας. Έτσι ο παραπάνω αλγόριθμος μπορεί να χρησιμοποιηθεί για τον υπολογισμό του γινομένου δύο $n \times n$ πινάκων με οικονομία στην απαιτούμενη μνήμη.

for $j=1,2,\dots,n$

$$h_i = \sum_{k=1}^n a_{ik} b_{kj}, \quad i=1,2,\dots,n$$

$$b_{ij} \equiv h_i, \quad i=1,2,\dots,n$$

Πολυπλοκότητα: $O(n^3)$ flops

Μνήμη: $2n^2+n$ θέσεις

(μνήμη για τα αρχικά διανύσματα n^2+n^2 , μνήμη για το προσωρινό διάνυσμα n , μνήμη για το άθροισμα 1, άρα σύνολο $n^2+n^2+n \cdot 1 = 2n^2+n$)

Προέχηση της αποτελεσματικότητας αλγορίθμων

Όταν προλαβούμε να μετρήσουμε τα flops που εμπεριέχονται ε'έναν αλγόριθμο μέχρι στιγμής είδαμε ότι εμφανίζονται διπτυπία άθροισματα (διπλά ή τριπλά) για τα οποία ενδιαφερόμαστε να υπολογίσουμε τον κυρίαρχο όρο (τη μεγαλύτερη εμφανιζόμενη δύναμη). Αυτός ο όρος μπορεί να υπολογισθεί αν απεικονίσουμε το άθροισμα με ολοκληρώματα προσαρμόζοντας κατάλληλα τα όρια ολοκληρώσεως

Για παράδειγμα, έστω το διπλό άθροισμα

$$\sum_{i=1}^n \sum_{j=1}^i 1 = \sum_{i=1}^n i = \frac{n(n+1)}{2} \approx \frac{n^2}{2}$$

Ο κυρίαρχος όρος του προηγούμενου άθροισματος μπορεί να προσεγγισθεί ως εξής:

$$\int_0^n \int_0^i 1 \, dj \, di = \int_0^n i \, di = \frac{n^2}{2}$$

Η μετρική των αριθμητικών πράξεων που περιέχονται, είναι
αλγόριθμο μπορεί να θεωρηθεί και δυνατότητα πρόβλεψης
του χρόνου εκτέλεσης του αλγορίθμου. Συγκεκριμένα, αν
πολλαπλασιάσουμε τον αριθμό των πράξεων με το χρόνο που
χρειάζεται για την εκτέλεση η συγκεκριμένη πράξη έχουμε
τη συνολική εκτίμηση του απαιτούμενου χρόνου. Πχ αν α
είναι ο απαιτούμενος χρόνος για την εκτέλεση μιας πράξης
ή ενός πολλαπλασιασμού τότε η επίλυση ενός άνω τριγωνικού
υστήματος θα απαιτούσε $\frac{n^2}{2}$ α χρόνο. Στην πράξη όμως αυτό δεν
πολογίζεται. Όταν ο αλγόριθμος αναφέρεται στο i, j στοιχείο ενός
πίνακα, θα πρέπει να το προσπελάσει από τη μνήμη και να
λέγξει τις τιμές των δεικτών i και j . Κατά συνέπεια, απαιτείται
προσθετός χρόνος για την άλλη εκτέλεση μιας πράξης. Έτσι
η εκτίμηση του συνολικού αριθμού των flops για την εκτέλεση
ένος αλγορίθμου μας επιτρέπει τα εξής:

- Πρόβλεψη της αύξησης του χρόνου εκτέλεσης του αλγορίθμου
συναρτήσει της διάστασης του n , γραμμικά, εκθετικά κλπ
- Καθιέρωση ενός μέτρου σύγκρισης μεταξύ διαφορετικών
αλγορίθμων. Πάντα θα προτιμάται ένας αλγόριθμος που έχει
υψίστο όρο μικρότερης δύναμης.

1.2. Θεωρία ανάλυσης εφέλματος

Βοηθητικά Λήμματα από την Ανάλυση

Λήμμα 1: Αν $0 \leq u < 1$ και αν $n = 1, 2, 3, \dots$ τότε

$$1 - nu \leq (1 - u)^n$$

$$\text{πχ } u = 10^{-10} \in (0, 1)$$

Λήμμα 2: Αν $n = 1, 2, 3, \dots$ και $0 \leq nu \leq 0.01$ τότε

$$(1 + u)^n \leq 1 + 1.01 nu$$

Συμείωση:

Στην εφαρμογή αυτού του Λήμματος, n είναι η τάξη ενός πίνακα και u είναι η μονάδα του εφέλματος στρογγύλευσης. Η υπόθεση $nu \leq 0.01$ είναι ρεαλιστική και ικανοποιείται σε όλα τα πρακτικά προβλήματα. Για παράδειγμα, αν $u = 10^{-15}$, τότε για να μην ικνύει αυτή η αυτεότιτα θα πρέπει $n > 10^{14}$. Εάν αρχίσουμε να προδίδουμε αριθμούς σε μία μηχανή με δυνατότητα πρόδεσης $1 \mu\text{sec} = 10^{-6} \text{sec}$, τότε ο χρόνος που θα χρειασθεί για να προδίδουμε 10^{14} αριθμούς είναι

$$10^8 \text{ sec} = 3,2 \text{ χρόνια!}$$

Λήμμα 3: Αν $|ε_i| \leq u$, για $i=1, 2, \dots, n$ και αν $nu \leq 0.01$, τότε

$$1 - nu \leq \prod_{i=1}^n (1 + \varepsilon_i) \leq 1 + 1.01 nu.$$

Πράγματι, αφού

$$|ε_i| \leq u \Leftrightarrow -u \leq \varepsilon_i \leq u \Leftrightarrow 1 - u \leq 1 + \varepsilon_i \leq 1 + u \Leftrightarrow (1 - u)^n \leq \prod_{i=1}^n (1 + \varepsilon_i) \leq (1 + u)^n$$

Επίσης $1 - nu \stackrel{\text{Λήμμα 1}}{\leq} (1 - u)^n \leq \prod_{i=1}^n (1 + \varepsilon_i) \leq (1 + u)^n \stackrel{\text{Λήμμα 2}}{\leq} 1 + 1.01 nu$

Ορίσουμε: $k_1: \underbrace{\frac{1}{2} \beta^{1-k_1}}_{u_1} = 1.01 \underbrace{\frac{1}{2} \beta^{1-k}}_u \stackrel{(*)}{\Rightarrow} u_1 = 1.01 u \quad (*')$

όπου u_1 μια νέα μονάδα εφάλματος εροχγύλευσης με k_1 γιφία 6ει mantissa

Λογαριθμώντας και τα δύο μέλη της $(*)$ έχουμε:

$$k_1 = k - \log_{\beta} 1.01.$$

όπου $\log_{\beta} 1.01 < 0$ και $k_1 \in \mathbb{Z}$

Με τη βοήθεια αυτών των οριβών μπορούμε να αναπαράβουμε ανιόττες της μορφής

$$(1 - u)^n \leq 1 + \varepsilon \leq (1 + u)^n$$

με την απλούστερη ανιόττα $|\varepsilon| \leq nu_1$, όπου n το μέγεθος του ηροβλήματος (π.χ. το ηλίδος των αριθμών που ηροβθύνεται).

Συμείωση: Αν είχα k γιφία 6ει mantissa, τότε $u = \frac{1}{2} \beta^{1-k}$ } $\Leftrightarrow u_1 < u$.
 Αν είχα $k_1 > k$ γιφία 6ει mantissa, τότε $u_1 = \frac{1}{2} \beta^{1-k_1}$

Συνοψίζοντας τα προηγούμενα έχουμε:

$$x \rightarrow \underbrace{fl(x)}_{\text{αριθμός μηχανής}}, \quad \left| \frac{x - fl(x)}{x} \right| \leq \frac{1}{2} \beta^{1-k}, \text{ όπου } k \text{ πλινθος ψηφίων στη μονάδα}$$

$$\text{ή } |e| \leq u. \quad (1)$$

$$\text{Επίσης: } (1-u)^n \leq 1+E \leq (1+u)^n \Leftrightarrow |E| \leq n \cdot u_1, \quad u_1 = 1.01u \quad (2)$$

$$\text{όπου } 1+E = \prod_{i=1}^n (1+\epsilon_i).$$

Στη συνέχεια εξετάζουμε τη διάδοση εφαλμάτων (propagation of errors) όταν έχουμε ακολουθίες αριθμητικών πράξεων.

1.2.1 Υπολογισμός γινομένου η αριθμών $x_1, \dots, x_n$ (σε παράσταση με κινητή υποδιαστολή (floating point))

Θέλουμε να υπολογίσουμε το γινόμενο P_n που φτίζει ως εξής:

$$P_n = fl(x_1 x_2 \dots x_n)$$

όπου $x_1, x_2, \dots, x_n$ είναι αριθμοί κινητής υποδιαστολής και οι πολλαπλασιασμοί εκτελούνται από αριστερά προς τα δεξιά.

Ος γνωστό ισχύει $x \rightarrow fl(x) = x \underbrace{(1+\epsilon)}_{\text{διασάραξη}}$, (ϵ μονάδα μηχανής).

Το υπολογιζόμενο γινόμενο των αριθμών $x_1, x_2, \dots, x_n$ είναι:

$$\begin{aligned} P_n = fl(x_1 x_2 \dots x_n) &= x_1 x_2 \dots x_n (1+E) \\ &= (x_1 x_2 \dots x_n) + \underbrace{(x_1 x_2 \dots x_n E)}_{E'} \quad (1') \\ &= \underbrace{(x_1 x_2 \dots x_n)}_{\text{πραγματική τιμή}} + \underbrace{E'}_{\text{εφαλμα}} \end{aligned}$$

Ορίζουμε αναγωγικά τις ποσότητες

$$P_1 = x_1$$

$$P_r = \underbrace{\text{fl}(P_{r-1} x_r)}_{\text{προυχθέντο} \cdot x_r} = P_{r-1} x_r (1 + \varepsilon_r), \quad r = 2, 3, \dots, n, \quad \text{όπου } |\varepsilon_r| \leq u$$

Με διαδοχικές αντικαταστάσεις στον παραπάνω τύπο, για $r = 2, 3, \dots, n$ προκύπτουν:

$$P_2 = \text{fl}(P_1 x_2) = P_1 x_2 (1 + \varepsilon_2) = x_1 x_2 (1 + \varepsilon_2)$$

$$P_3 = \text{fl}(P_2 x_3) = P_2 x_3 (1 + \varepsilon_3) = x_1 x_2 x_3 (1 + \varepsilon_2)(1 + \varepsilon_3)$$

$\vdots$

$$P_n = \text{fl}(P_{n-1} x_n) = P_{n-1} x_n (1 + \varepsilon_n) = x_1 x_2 x_3 \dots x_n (1 + \varepsilon_2)(1 + \varepsilon_3) \dots (1 + \varepsilon_n)$$

$$= x_1 x_2 x_3 \dots x_n (1 + E), \quad \text{όπου}$$

$$1 + E = (1 + \varepsilon_2)(1 + \varepsilon_3) \dots (1 + \varepsilon_n)$$

$$\text{Αφού } |\varepsilon_r| \leq u \Leftrightarrow 1 - u \leq 1 + \varepsilon_r \leq 1 + u, \quad r = 2, 3, \dots, n$$

$$\text{Επομένως } \left. \begin{array}{l} 1 - u \leq 1 + \varepsilon_2 \leq 1 + u \\ 1 - u \leq 1 + \varepsilon_3 \leq 1 + u \\ \vdots \\ 1 - u \leq 1 + \varepsilon_n \leq 1 + u \end{array} \right\} \begin{array}{l} (*) \\ \Rightarrow (1 - u)^{n-1} \leq \prod_{i=2}^n (1 + \varepsilon_i) = 1 + E \leq (1 + u)^{n-1} \end{array}$$

$$\text{όπου λόγω της (2) έχουμε } |E| \leq (n-1)u_1 = 1.01(n-1)u$$

Επομένως, το σφάλμα στον υπολογισμό γινομένου είναι:

$$\text{Σφάλμα} = E' = x_1 x_2 \dots x_n E$$

$$\text{Απόλυτο σφάλμα} = |E'| = |x_1| |x_2| \dots |x_n| |E| \leq |x_1| |x_2| \dots |x_n| 1.01(n-1)u$$

$$\text{Απόλυτο σχετικό σφάλμα} = \frac{|E'|}{|x_1| |x_2| \dots |x_n|} \leq 1.01(n-1)u$$

Αρα, στον πολλαπλασιασμό ΔΕΝ παίζει ρόλο η σειρά των παραγόντων για το σφάλμα.

Σχετικό σφάλμα γινομένου

Για το γινόμενο από την (1') έχουμε:

$$\begin{aligned} fl(x_1 x_2 \dots x_n) &= x_1 x_2 \dots x_n + x_1 x_2 \dots x_n E \\ &= x_1 x_2 \dots x_n + E' \end{aligned}$$

$$\text{με } |E'| = |x_1| |x_2| \dots |x_n| |E|$$

Η ανισότητα $(1-u)^{n-1} \leq 1+E \leq (1+u)^{n-1}$ αντιστρέφεται από την ανισότητα $|E| \leq (n-1)u$, οπότε

$$|E'| \leq |x_1| |x_2| \dots |x_n| (n-1)u_1.$$

Το σχετικό σφάλμα δίνεται από τον τύπο

$$\frac{|fl(x_1 x_2 \dots x_n) - (x_1 x_2 \dots x_n)|}{|x_1 x_2 \dots x_n|} \leq (n-1)u_1 \frac{|x_1| |x_2| \dots |x_n|}{|x_1 x_2 \dots x_n|}$$

$$\Rightarrow \frac{|fl(x_1 x_2 \dots x_n) - (x_1 x_2 \dots x_n)|}{|x_1 x_2 \dots x_n|} \leq (n-1)u_1.$$

Παρατήρηση: Επειδή πάντοτε το $(n-1)u_1$ είναι πολύ μικρό, ο υπολογισμός του γινομένου n αριθμών είναι πάντοτε ευεταδής.

Παράδειγμα:

Να υπολογιστεί η εξής παράσταση

$$d = fl(x_1 x_2 \dots x_m / y_1 y_2 \dots y_n)$$

όπου $x_1, x_2, \dots, x_m, y_1, y_2, \dots, y_n$ είναι αριθμοί μοναχίας υποδιαστολής. Να εξετάσετε αν το σφάλμα επηρεάζεται από τη σειρά εκτέλεσης των πράξεων

Λύση

Αρχικά υπολογίζουμε την παράσταση $d = fl(x_1 x_2 \dots x_m / y_1 y_2 \dots y_n)$ με εκτέλεση των πράξεων από αριστερά προς τα δεξιά. Παρατηρούμε ότι

$$\begin{aligned} d &= fl(x_1 x_2 \dots x_m / y_1 y_2 \dots y_n) = \\ &= fl(fl(x_1 x_2 \dots x_m) / y_1 y_2 \dots y_n) = \\ &= fl\left(\left(fl(x_1 x_2 \dots x_m) / y_1\right) y_2 \dots y_n\right) = \\ &= fl\left(fl\left(\left(x_1 x_2 \dots x_m\right) \prod_{i=2}^m (1 + \varepsilon_i) / y_1\right) y_2 \dots y_n\right) = \\ &= fl\left(\frac{x_1 x_2 \dots x_m}{y_1} \prod_{i=2}^m (1 + \varepsilon_i) (1 + \varepsilon) y_2 \dots y_n\right) \\ &= \left(\frac{x_1 x_2 \dots x_m}{y_1}\right) \prod_{i=2}^m (1 + \varepsilon_i) (1 + \varepsilon) (y_2 \dots y_n) \prod_{i=2}^n (1 + \varepsilon_i) \quad , \quad |\varepsilon_i| \leq u, |\varepsilon| \leq u \end{aligned}$$

όπου

$$\left. \begin{aligned} (1-u)^{m-1} &\leq \prod_{i=2}^m (1+\varepsilon_i) \leq (1+u)^{m-1} \\ 1-u &\leq 1+\varepsilon \leq 1+u \\ (1-u)^{n-1} &\leq \prod_{i=2}^n (1+\varepsilon_i) \leq (1+u)^{n-1} \end{aligned} \right\} \begin{aligned} & \\ & \\ & \end{aligned} \quad \left(\begin{array}{c} * \\ \equiv \end{array} \right)$$

$$\Rightarrow (1-u)^{m-1} (1-u) (1-u)^{n-1} \leq \underbrace{\prod_{i=2}^m (1+\varepsilon_i) (1+\varepsilon) \prod_{i=2}^n (1+\varepsilon_i)}_{(\text{Θέτω}) \quad 1+E} \leq (1+u)^{m-1} (1+u) (1+u)^{n-1}$$

$$\Rightarrow (1-u)^{m+n-1} \leq 1+E \leq (1+u)^{m+n-1} \quad (3)$$

$$1 < \alpha_1 \quad \mathbb{P}(x_1 x_2 \dots x_m / y_1 y_2 \dots y_n) \equiv (x_1 x_2 \dots x_m / y_1 y_2 \dots y_n) (1+E)$$

Παρατηρούμε ότι η σειρά με την οποία εκτελούνται οι πράξεις, δεν επηρεάζει τα γράμματα του βφάλματος, όπως φαίνεται από την (3).

Επομένως το ίδιο βφάλμα προκύπτει αν εκτελεστούν οι πράξεις από δεξιά προς τα αριστερά.

1.2.2. Υπολογισμός αθροίσματος

Θέλουμε να υπολογίσουμε το άθροισμα s_n που ορίζεται ως εξής:

$$s_n = fl(x_1 + x_2 + \dots + x_n)$$

όπου $x_1, x_2, \dots, x_n$ είναι αριθμοί κινητής υποδιαστολής και οι προβάσεις εκτελούνται από αριστερά προς τα δεξιά. Οι παρόντες s_r ορίζονται αναγωγικά από τις σχέσεις

$$s_1 = x_1$$

$$s_r = fl(s_{r-1} + x_r) \equiv (s_{r-1} + x_r)(1 + \varepsilon_r), \quad |\varepsilon_r| \leq u, \quad r = 2, 3, \dots, n$$

Με διαδοχικές αντικαταστάσεις στον ανωτέρω τύπο για $r = 2, 3, \dots, n$ προκύπτει

$$s_2 = fl(s_1 + x_2) = (x_1 + x_2)(1 + \varepsilon_2)$$

$$s_3 = fl(s_2 + x_3) = (s_2 + x_3)(1 + \varepsilon_3) = (x_1 + x_2)(1 + \varepsilon_2)(1 + \varepsilon_3) + x_3(1 + \varepsilon_3)$$

$$s_4 = fl(s_3 + x_4) = \dots = (x_1 + x_2)(1 + \varepsilon_2)(1 + \varepsilon_3)(1 + \varepsilon_4) + x_3(1 + \varepsilon_3)(1 + \varepsilon_4) + x_4(1 + \varepsilon_4)$$

⋮

$$s_n = fl(s_{n-1} + x_n) = \dots = (x_1 + x_2)(1 + \varepsilon_2) \dots (1 + \varepsilon_n) + x_3(1 + \varepsilon_3) \dots (1 + \varepsilon_n) + \\ + x_4(1 + \varepsilon_4) \dots (1 + \varepsilon_n) + \dots + x_n(1 + \varepsilon_n)$$

$$= x_1(1 + \varepsilon_2) \dots (1 + \varepsilon_n) + x_2(1 + \varepsilon_2) \dots (1 + \varepsilon_n) \\ + x_3(1 + \varepsilon_3) \dots (1 + \varepsilon_n) \\ + \dots + \\ + x_n(1 + \varepsilon_n)$$

$$= x_1 \prod_{i=2}^n (1 + \varepsilon_i) + \sum_{r=2}^n \left\{ x_r \prod_{i=r}^n (1 + \varepsilon_i) \right\}, \quad |\varepsilon_i| \leq u, \quad i = 2, 3, \dots, n$$

Έστω $1+u_1 = \prod_{i=2}^n (1+e_i)$ και $1+u_r = \prod_{i=r}^n (1+e_i)$, $r=2,3,\dots,n$

Τότε $S_n = x_1(1+u_1) + x_2(1+u_2) + \dots + x_n(1+u_n)$, όπου

$$(1-u)^{n-1} \leq 1+u_1 \leq (1+u)^{n-1}$$

$$(1-u)^{n-(r-1)} \leq 1+u_r \leq (1+u)^{n-(r-1)}, \quad r=2,3,\dots,n$$

Οι δύο τελευταίες ανισότητες συνδυάζονται σε μία, δηλ.

$$(1-u)^{n-r+1} \leq 1+u_r \leq (1+u)^{n-r+1}, \quad r=1,2,3,\dots,n$$

Τέλος

$$f(x_1+x_2+\dots+x_n) \equiv x_1(1+u_1) + x_2(1+u_2) + \dots + x_n(1+u_n) \quad (4)$$

όπου $(1-u)^{n-r+1} \leq 1+u_r \leq (1+u)^{n-r+1}, \quad r=1,2,3,\dots,n \quad (5)$

Παρατήρηση: Παρατηρούμε ότι τα φράγματα των βφαλμάτων, όπως δίνονται από την (4) εξαρτώνται από τη σειρά με την οποία επιλέγεται η πρόθεση. Κατά συνέπεια το άνω φράγμα για το βφαλμα είναι μικρότερο αν οι προτιθέμενοι αριθμοί είναι διατεταγμένοι κατά αύξουσα σειρά μεγέθους. Αυτό δικαιολογείται από το γεγονός ότι ο μεγαλύτερος παραγοντας βφαλματος $\prod_{i=2}^n (1+e_i)$ θα συσχετιστεί με το μικρότερο x_i .

Σχετικό βήμα αδοσίματος.

Από την (4) έχουμε:

$$\begin{aligned} f(x_1+x_2+\dots+x_n) &= x_1(1+u_1) + x_2(1+u_2) + \dots + x_n(1+u_n) \\ &= x_1+x_2+\dots+x_n + x_1u_1+x_2u_2+\dots+x_nu_n \\ &= x_1+x_2+\dots+x_n + E \end{aligned}$$

$$\text{όπου } |E| = |x_1u_1+x_2u_2+\dots+x_nu_n| \leq |x_1||u_1|+|x_2||u_2|+\dots+|x_n||u_n| \quad (6)$$

Επίσης από την (5) έχουμε

$$(1-u)^{n-r+1} \leq 1+u_r \leq (1+u)^{n-r+1}, \quad r=1,2,3,\dots,n$$

η οποία λόγω της (2) γράφεται

$$|u_r| \leq (n-r+1)u_1 \quad (7)$$

Οπότε από την (6)

$$\begin{aligned} |E| &\leq |x_1|(n-1+1)u_1 + |x_2|(n-2+1)u_1 + \dots + |x_n|(n-n+1)u_1 \\ &= u_1(n|x_1|+(n-1)|x_2|+\dots+2|x_{n-1}|+|x_n|) \quad (8) \end{aligned}$$

Συνεπώς το σχετικό βήμα δίνεται από τον τύπο:

$$\begin{aligned} \frac{|f(x_1+x_2+\dots+x_n)-(x_1+x_2+\dots+x_n)|}{|x_1+x_2+\dots+x_n|} &= \frac{|(x_1+x_2+\dots+x_n)+E-(x_1+x_2+\dots+x_n)|}{|x_1+x_2+\dots+x_n|} \\ &= \frac{|E|}{|x_1+x_2+\dots+x_n|} \stackrel{(8)}{\leq} \frac{u_1(n|x_1|+(n-1)|x_2|+\dots+|x_n|)}{|x_1+x_2+\dots+x_n|} \leq \frac{u_1(n|x_1|+n|x_2|+\dots+n|x_n|)}{|x_1+x_2+\dots+x_n|} = \\ &= nu_1 \frac{|x_1|+|x_2|+\dots+|x_n|}{|x_1+x_2+\dots+x_n|} \end{aligned}$$

Παρατηρήσεις: Εάν $|x_1 + x_2 + \dots + x_n| \ll |x_1| + |x_2| + \dots + |x_n|$, τότε το

σχετικό σφάλμα του υπολογισμού του αθροίσματος ή αριθμών

ευδύναται να μην είναι μικρό. Η ποσότητα $k = \frac{|x_1| + |x_2| + \dots + |x_n|}{|x_1 + x_2 + \dots + x_n|}$

αποτελεί ένα δείκτη κατάστασης (condition number) ώστε, ανάλογα

με την τιμή που παίρνει, το σχετικό σφάλμα ευδύναται να είναι

μικρό ή μεγάλο. Όταν όλοι οι όροι του αθροίσματος είναι

θετικοί (ή όλοι αρνητικοί), τότε το k είναι μονάδα και ο υπολογισμός

του αθροίσματος είναι ευσταθής αφού το σχετικό σφάλμα θα

ικανοποιεί την ανισότητα $Rel \leq n u_1$.

Τα σφάλματα εροχχύλευσης συσσωρεύονται αργά καθώς προστίθενται

οι όροι του αθροίσματος. Σε αυτή την περίπτωση, το φράγμα n είναι

μια υπερεκτίμηση της τιμής που εμφανίζεται στην πράξη αφού τα

σφάλματα εροχχύλευσης τότε θα έχουν θετική και τότε αρνητική

τιμή, οπότε πολλά από αυτά θα αλληλοακυρώνονται.

Στην περίπτωση που η πρόσθεση εκτελείται με αποκοπή, τα

σφάλματα εροχχύλευσης τείνουν όλα στην ίδια κατεύθυνση (προς

τα πάνω) και πέφει το ένα το άλλο, οπότε ο παράγοντας n του

φραγματος είναι ρεαλιστικός.

Παράδειγμα: Έστω $t=4$. Να ελεγχθούν τα σφάλματα που προκύπτουν κατά την πρόθεση των παρακάτω αριθμών χρησιμοποιώντας τεροχγύλευση και αποκοπή

αριθμός	Στεροχγύλευση	Σφάλμα	Αποκοπή	Σφάλμα
1374.8	1375	-0.2	1374	+0.8
3856.4	3856	+0.4	3856	+0.4
5231.2	5231	+0.2	5230	+1.2

Παρατηρούμε ότι τα σφάλματα που προέκυψαν στη τεροχγύλευση έχουν αντίθετα πρόσημα κι έτσι στο τελικό άθροισμα έχουμε το μικρό σφάλμα +0.2, δηλαδή η αληθινή τιμή είναι κατά 0.2 μεγαλύτερη από την υπολογιζόμενη. Αντίθετα, στην αποκοπή τα σφάλματα έχουν το ίδιο πρόσημο και πιέζουν το ένα το άλλο με αποτέλεσμα να έχουμε το μεγαλύτερο σφάλμα +1.2.

Συμπέρασμα: Τα σφάλματα από την τεροχγύλευση τείνουν να ναιρεί το ένα το άλλο, ενώ στην αποκοπή συσσωρεύονται προδεδικά. Γι' αυτό, πάντα προτιμάται η τεχνική της τεροχγύλευσης.

1.2.3. Υπολογισμός εσωτερικού γινομένου

Θέλουμε να υπολογίσουμε το εσωτερικό γινόμενο i_{P_n} , το οποίο ορίζεται ως εξής:

$$i_{P_n} = fl(a_1 b_1 + a_2 b_2 + \dots + a_n b_n)$$

όπου a_i και b_i είναι αριθμοί μηχανής υποδιαστολής.

Γενικά γνωρίζουμε ότι

$$\left. \begin{array}{l} a = (a_1, \dots, a_n)^T \\ b = (b_1, \dots, b_n)^T \end{array} \right\} a \cdot b = a^T b = (a_1, \dots, a_n) \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

$$\text{Αφού } x \rightarrow \underbrace{fl(x)}_{\text{αριθμός μηχανής}} = x \underbrace{(1+\varepsilon)}_{\text{διαφορά}}, \quad |\varepsilon| \leq u, \quad \varepsilon \text{ μονάδα μηχανής}$$

θα ισχύει ότι

$$\begin{aligned} a \cdot b &\Rightarrow fl(a \cdot b) = fl(a_1 b_1 + a_2 b_2 + \dots + a_n b_n) \\ &= a_1 b_1 (1+\varepsilon_1) + a_2 b_2 (1+\varepsilon_2) + \dots + a_n b_n (1+\varepsilon_n) \quad (9) \end{aligned}$$

Πρώτα υπολογίζονται τα γινόμενα και μετά προστίθενται μεταξύ τους από αριστερά προς τα δεξιά. Ορίζουμε αναγωγικά τις ποσότητες i_{P_r} (εσωτερικό γινόμενο $\rightarrow$ περιλαμβάνει άθροιση) και t_r (για το κλασικό γινόμενο) όπως και z_r το σφάλμα του γινομένου και u_r το σφάλμα του αθροίσματος.

Γίνεται:

- 39 -

$$t_r = fl(a_r b_r) \quad \text{onore} \quad \begin{cases} t_1 = fl(a_1 b_1) = a_1 b_1 (1 + \gamma_1) \\ t_2 = fl(a_2 b_2) = a_2 b_2 (1 + \gamma_2) \\ t_3 = fl(a_3 b_3) = a_3 b_3 (1 + \gamma_3) \\ \vdots \\ t_n = fl(a_n b_n) = a_n b_n (1 + \gamma_n) \end{cases}$$

$$i_{p_1} = t_1 = a_1 b_1 (1 + \gamma_1)$$

$$\begin{aligned} i_{p_2} &= fl(i_{p_1} + t_2) = (i_{p_1} + t_2)(1 + \eta_2) = [a_1 b_1 (1 + \gamma_1) + a_2 b_2 (1 + \gamma_2)](1 + \eta_2) \\ &= a_1 b_1 (1 + \gamma_1)(1 + \eta_2) + a_2 b_2 (1 + \gamma_2)(1 + \eta_2) \end{aligned}$$

$$\begin{aligned} i_{p_3} &= fl(i_{p_2} + t_3) = (i_{p_2} + t_3)(1 + \eta_3) \\ &= [a_1 b_1 (1 + \gamma_1)(1 + \eta_2) + a_2 b_2 (1 + \gamma_2)(1 + \eta_2) + a_3 b_3 (1 + \gamma_3)](1 + \eta_3) \\ &= a_1 b_1 (1 + \gamma_1)(1 + \eta_2)(1 + \eta_3) + a_2 b_2 (1 + \gamma_2)(1 + \eta_2)(1 + \eta_3) + a_3 b_3 (1 + \gamma_3)(1 + \eta_3) \end{aligned}$$

$$\begin{aligned} i_{p_4} &= fl(i_{p_3} + t_4) = (i_{p_3} + t_4)(1 + \eta_4) = \dots = \\ &= a_1 b_1 (1 + \gamma_1)(1 + \eta_2)(1 + \eta_3)(1 + \eta_4) + a_2 b_2 (1 + \gamma_2)(1 + \eta_2)(1 + \eta_3)(1 + \eta_4) + \\ &\quad + a_3 b_3 (1 + \gamma_3)(1 + \eta_3)(1 + \eta_4) + a_4 b_4 (1 + \gamma_4)(1 + \eta_4) \end{aligned}$$

Αναγωγικά

$$\begin{aligned} i_{p_n} &= fl(i_{p_{n-1}} + t_n)(1 + \eta_n) = \dots = a_1 b_1 (1 + \gamma_1)(1 + \eta_2)(1 + \eta_3) \dots (1 + \eta_n) \\ &\quad + a_2 b_2 (1 + \gamma_2)(1 + \eta_2)(1 + \eta_3) \dots (1 + \eta_n) \\ &\quad + a_3 b_3 (1 + \gamma_3)(1 + \eta_3) \dots (1 + \eta_n) \\ &\quad + a_4 b_4 (1 + \gamma_4)(1 + \eta_4) \dots (1 + \eta_n) \\ &\quad + \dots \\ &\quad + a_n b_n (1 + \gamma_n)(1 + \eta_n) \end{aligned} \quad (10)$$

Από (9) και (10) έχουμε:

$$1+\varepsilon_1 = \overbrace{(1+z_1)(1+u_2)(1+u_3)\dots(1+u_n)}^{n-\text{παραγοντες}} \Rightarrow (1-u)^n \leq 1+\varepsilon_1 \leq (1+u)^n \quad (11)$$

$$1+\varepsilon_2 = \overbrace{(1+z_2)(1+u_2)(1+u_3)\dots(1+u_n)}^{n-\text{παραγοντες}} \Rightarrow (1-u)^n \leq 1+\varepsilon_2 \leq (1+u)^n$$

$$1+\varepsilon_3 = \overbrace{(1+z_3)(1+u_3)\dots(1+u_n)}^{n-1=n-3+2 \text{ παραγοντες}} \Rightarrow (1-u)^{n-1} \leq 1+\varepsilon_3 \leq (1+u)^{n-1}$$

$$1+\varepsilon_4 = \overbrace{(1+z_4)(1+u_4)\dots(1+u_n)}^{n-2=n-4+2 \text{ παραγοντες}} \Rightarrow (1-u)^{n-2} \leq 1+\varepsilon_4 \leq (1+u)^{n-2}$$

⋮

$$1+\varepsilon_r = \overbrace{(1+z_r)(1+u_r)\dots(1+u_n)}^{n-r+2 \text{ παραγοντες}} \Rightarrow (1-u)^{n-r+2} \leq 1+\varepsilon_r \leq (1+u)^{n-r+2} \quad (12)$$

$r=2,3,\dots,n$

$$1+\varepsilon_n = \overbrace{(1+z_n)(1+u_n)}^{(n-n+2) \text{ παραγοντες}} \Rightarrow (1-u)^2 \leq 1+\varepsilon_n \leq (1+u)^2$$

Αφού $u > 0$ ισχύει ότι $1-u < 1$ και $1+u > 1$, άρα από την (11) έχουμε

$$(1-u)^{n-1} \leq (1-u)^n \leq 1+\varepsilon_1 \leq (1+u)^n \leq (1+u)^{n+1} \quad (13)$$

και από (12), (13) έχουμε

$$(1-u)^{n-r+2} \leq 1+\varepsilon_r \leq (1+u)^{n-r+2}, \quad r=1,2,3,\dots,n$$

και λόγω της (2) έχουμε

$$|\varepsilon_r| \leq (n-r+2) 1.01 u. \quad (14)$$

Σχετικό φάρμα εσωτερικών γινόμενων.

Από την (9) έχουμε

$$\begin{aligned} fl(a_1b_1+a_2b_2+\dots+a_nb_n) &= \overbrace{a_1b_1(1+\varepsilon_1)+\dots+a_nb_n(1+\varepsilon_n)}^{\text{προσεγγιστική τιμή}} = \\ &= \underbrace{(a_1b_1+a_2b_2+\dots+a_nb_n)}_{\text{ακριβής τιμή}} + \underbrace{(a_1b_1\varepsilon_1+a_2b_2\varepsilon_2+\dots+a_nb_n\varepsilon_n)}_{\text{σφάλμα } E} \end{aligned}$$

Επομένως αφού γενικά ισχύει ότι το

Απόλυτο εφάλμα = αριθμός τιμών - προβλεπόμενη τιμή

έχουμε ότι

$$\begin{aligned}
 |E| &= |a_1 b_1 \varepsilon_1 + a_2 b_2 \varepsilon_2 + \dots + a_n b_n \varepsilon_n| \\
 &\leq |a_1| |b_1| |\varepsilon_1| + |a_2| |b_2| |\varepsilon_2| + \dots + |a_n| |b_n| |\varepsilon_n| \\
 &\leq |a_1| |b_1| (n-1+2) 1.01u + |a_2| |b_2| (n-2+2) 1.01u + \dots + |a_n| |b_n| (n-n+2) 1.01u \\
 &= 1.01u [(n+1)|a_1| |b_1| + n|a_2| |b_2| + \dots + 2|a_n| |b_n|] \\
 &\leq 1.01u [(n+1)|a_1| |b_1| + (n+1)|a_2| |b_2| + \dots + (n+1)|a_n| |b_n|] \\
 &= 1.01u (n+1) [|a_1| |b_1| + |a_2| |b_2| + \dots + |a_n| |b_n|] \\
 &= 1.01u (n+1) |a|^T |b|
 \end{aligned}$$

Επομένως, το απόλυτο σχετικό εφάλμα στον υπολογισμό του εσωτερικού γινομένου δίνεται από τη σχέση

$$\frac{|\ell(a^T b) - a^T b|}{|a^T b|} = \frac{|E|}{|a^T b|} \leq \frac{(n+1)1.01u |a|^T |b|}{|a^T b|} = (n+1)u_1 \frac{|a|^T |b|}{|a^T b|}$$

Παρατηρούμε ότι αν $|a^T b| \ll |a|^T |b|$, τότε το σχετικό εφάλμα στο $\ell(a^T b)$ ενδέχεται να μην είναι μικρό.

Όλα τα προηγούμενα συνοψίζονται στον παρακάτω πίνακα

Υπολογισμός	Σχετικό σφάλμα	Παρατηρήσεις
$fl(x_1 + x_2 + \dots + x_n)$	Όχι πάντοτε μικρό αν $ \sum_{i=1}^n x_i \ll \sum_{i=1}^n x_i $	Μικρότερο σφάλμα αν οι αριθμοί είναι διατεταγμένοι μετ' αύξουσα βερά
$fl(x_1 \cdot x_2 \cdot \dots \cdot x_n)$	Πάντοτε μικρό	Δεν παίζει ρόλο η βερά εκτέλεσης των πράξεων
$fl(a^T b), a, b \in \mathbb{R}^n$	Όχι πάντοτε μικρό, αν $ a^T b \ll a ^T b $	Οι πράξεις εκτελούνται ιεραρχικά.

Είναι αξιοσημείωτο να αναφερθεί ότι ο θεμελιωτής της θεωρίας σφάλματος είναι ο J. Wilkinson με το βιβλίο του Rounding Errors in Algebraic Processes. Ένα προς επίλυση πρόβλημα συνήθως έχει ένα συγκεκριμένο αριθμό τιμών εισόδου (δεδομένα) π.χ. $x_1, x_2, \dots, x_n$ και ένα συγκεκριμένο αριθμό τιμών εξόδου (αποτελέσματα) $y_1, y_2, \dots, y_n$ και κάποια (συνήθως πολυώνυμα) συναρτήσεις που περιγράφουν πως οι τιμές εξόδου σχετίζονται με τις τιμές εισόδου. Στα επόμενα, για λόγους απλότητας θεωρούμε την απλούστερη περίπτωση για μία τιμή εισόδου και μία τιμή εξόδου, δηλ. $y = f(x)$.

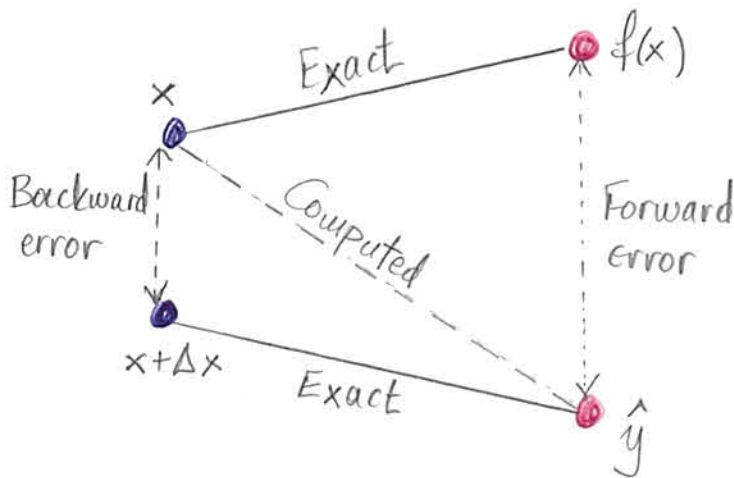
Η ανάλυση σφάλματος διακρίνεται σε 1) Forward (εκ των προτέρων) και 2) Backward (εκ των υστέρων) ανάλυση σφάλματος.

Εκ των προτέρων (Forward) και εκ των υστέρων (Backward) Ανάλυση Σφάλματος

1. Έστω $\hat{y}$ μια πρόβλεψη της ποσότητας $y=f(x)$ με αριθμητική ακρίβεια ϵ

2. Εκ των Προτέρων (Forward) ανάλυση σφάλματος: Εύρεση ενός άνω φράγματος για το Forward σφάλμα $|\Delta y| = |y - \hat{y}|$

3. Εκ των Υστέρων (Backward) ανάλυση σφάλματος: Εύρεση ενός άνω φράγματος για το Backward σφάλμα Δx τέτοιο ώστε $f(x + \Delta x) = \hat{y}$



Σύγκριση Forward και Backward Σφάλματος

→ Εκ των Προτέρων (Forward) ανάλυση σφάλματος

- πιο φυσική

- συνήθως είναι δύσκολο να υπολογιστεί διότι συνήθως είναι άγνωστη η λύση

→ Εκ των Υστέρων (Backward) ανάλυση σφάλματος

- ορίζεται θεωρώντας ότι το $\hat{y}$ είναι η λύση ενός "εχθρού" προβλήματος με τη διαταραχμένη είσοδο δεδομένων $\hat{x}$

- συνήθως είναι δύσκολο να υπολογιστεί διότι συνήθως είναι άγνωστη η λύση.

Γενικές έννοιες:

1. Πρόβλημα καλώς ορισμένο (well posed)

- Υπάρχει η λύση του
- Είναι μοναδική
- Εξαρτάται κατά συνεχή τρόπο από τα δεδομένα (ευστάθεια), δηλαδή μικρές μεταβολές στην είσοδο (δεδομένα) δεν παρουσιάζουν μεγάλες μεταβολές στην έξοδο (αποτελέσματα)

2. Πρόβλημα well-conditioned (καλώς κατάστασης)

- Καλώς ορισμένο
- Έχει μικρό μεταδιδόμενο σφάλμα

3. Σχετικός αριθμός συνθήκης (δεντικής κατάστασης - condition number)

$$k = \|A\| \cdot \|A^{-1}\|$$

$$k = \text{cond}_f(x) = \frac{\left\| \frac{f(\hat{x}) - f(x)}{\hat{x} - x} \right\|}{\frac{\|\hat{x} - x\|}{\|x\|}} = \frac{\frac{\|\Delta y\|}{\|y\|}}{\frac{\|\Delta x\|}{\|x\|}}$$

Προβλήματα

→ Καλώς Ορισμένα:

- Well Conditioned → "μικρό" $\left(\frac{\|\Delta y\|}{\|y\|} \right) / \left(\frac{\|\Delta x\|}{\|x\|} \right)$
- Ill Conditioned

→ Κακώς Ορισμένα

5. Αλγόριθμοι

→ Backward ευστάθεις ("μικρό" $\|\Delta x\| / \|x\|$)

→ Αεστάθεις

Αριθμός λύση: • Well Conditioned πρόβλημα
• Backward Ευστάθεις Αλγόριθμος

Διαδιδόμενο εφάλμα

Θεωρούμε μία πρόβλεψη $\tilde{x}$ στην τιμή x (τιμή εισόδου) - Θα μπορούσαμε να θεωρήσουμε ότι η πρόβλεψη οφείλεται σε λάθος μέτρησης της τιμής. Πρέπει να υπολογίσουμε την ποσότητα $\tilde{y} := f(\tilde{x})$. Το σχετικό εφάλμα για το $\tilde{y}$ είναι:

$$\epsilon_y := \frac{\tilde{y} - y}{y} = \frac{f(\tilde{x}) - f(x)}{f(x)} \stackrel{\text{ΘΜΤ}}{\approx} \frac{(\tilde{x} - x)f'(x)}{f(x)} \stackrel{*,1^{\circ}}{=} \frac{x f'(x)}{f(x)} \cdot \frac{\tilde{x} - x}{x} = c_f \cdot \epsilon_x \quad (15)$$

που $c_f := \frac{x f'(x)}{f(x)}$ ορίζεται ως ο αριθμός συνθήκης (condition number) της συνάρτησης f στο x . Ο αριθμός συνθήκης δείχνει πόσο ευαίσθητο είναι το πρόβλημα σε μικρές μεταβολές στα δεδομένα.

- Αν $|c_f|$ όχι $\gg$ (πολύ μεγαλύτερο) του 1 τότε το πρόβλημα είναι καλώς ορισμένο
- Αν $|c_f| \gg 1$, τότε το πρόβλημα είναι κακώς ορισμένο

Παράδειγμα

Έστω $f(x) = \frac{1}{x}$, τότε $c_f = \frac{x(-\frac{1}{x^2})}{\frac{1}{x}} = -\frac{x^2}{x^2} = -1 < 1$, οπότε $|c_f| = 1$ και συνεπώς το πρόβλημα είναι καλώς ορισμένο.

Τρόποι Μηχανής και Αλγόριθμοι σε Αριθμητική Μηχανή

Έστω x πραγματικός αριθμός με $x_{\min} \leq x \leq x_{\max}$, ο οποίος μπορεί να παρασταθεί στον υπολογιστή με τον αριθμό μηχανής $fl(x)$ έτσι ώστε

$$\left| \frac{fl(x) - x}{x} \right| \leq \epsilon_m, \text{ όπου } \epsilon_m \text{ είναι η μονάδα μηχανής (machine unit).}$$

Ένας αλγόριθμος σε αριθμητική μηχανή είναι μια συνάρτηση $\hat{y} = \hat{f}(\hat{x})$ η οποία

α) παίρνει σαν είσοδο τον αριθμό μηχανής $\hat{x}$

β) εκτελεί συγκεκριμένες πράξεις σε αριθμητική μηχανή ($\hat{f}$) και

γ) δίνει ως έξοδο τον αριθμό μηχανής $\hat{y}$.

Αντί να υπολογίζεται το $y = f(x)$ υπολογίζεται το $\hat{y} = \hat{f}(\hat{x})$ για $\hat{x} = fl(x)$.

να ενδιαφέρον ερώτημα που τίθεται είναι ποιά είναι η καλύτερη κρίβεια $|\frac{\hat{y}-y}{y}|$ που μπορούμε να πετύχουμε. Ένας ιδανικός αλγόριθμος

υπολογίζει απευθείας το $\tilde{y} := f(\hat{x})$ και μετά προερχτεί το $\tilde{y}$ με τον

αποτέλεσμα αριθμό μηχανής $\hat{y} := fl(\tilde{y})$ ή $\hat{f}(\hat{x}) := fl(f(\hat{x}))$, έτσι

κρίβεια

$$\left| \frac{\hat{x}-x}{x} \right| \leq \epsilon_M, \quad \left| \frac{\tilde{y}-y}{y} \right| \leq |c_f| \cdot \epsilon_M$$

και

$$\left| \frac{\hat{y}-y}{y} \right| = \left| \frac{(\hat{y}-\tilde{y}) + (\tilde{y}-y)}{y} \right| \leq \left| \frac{\hat{y}-\tilde{y}}{y} \right| + \left| \frac{\tilde{y}-y}{y} \right| = \left| \frac{fl(\tilde{y})-\tilde{y}}{fl(x)} \right| + \left| \frac{\tilde{y}-y}{y} \right| \leq$$

$$\leq \epsilon_M + |c_f| \epsilon_M. \quad (16)$$

Παράδειγμα — Εκ των προτέρων (Forward) Ανάλυση Σφάλματος

Έστω $y = f(x) := 1 - \cos x$, $x = 10^{-5}$ και χρησιμοποιούμε αριθμητική διπλής ακρίβειας. Τότε έχουμε: $c_f = \frac{x \sin x}{1 - \cos x}$.

Γαχνει ότι $1 - \cos x = 2 \sin^2\left(\frac{x}{2}\right)$. Άρα, $c_f = \frac{x \sin x}{2 \sin^2 \frac{x}{2}}$.

Πομένως,

$$|c_f| = \left| \frac{x \sin x}{2 \sin^2\left(\frac{x}{2}\right)} \right| \leq \frac{x^2}{2x^2/4} = 2$$

αφού $|\sin x| \leq |x|$. Συνεπώς η συνάρτηση είναι καλά ορισμένη και το ναυόφενκτο βφάλμα είναι $|c_f| \epsilon_M + \epsilon_M = 3 \cdot 10^{-16}$, όπου η μονάδα μηχανής είναι $\epsilon_M = 10^{-16}$.

1^{ος} Αλγόριθμος

Είναι $y := 1 - \cos x$. Θεω $y_1 := \cos x$, οπότε $y = 1 - y_1$ (17)

και κατά συνέπεια $\hat{x} := fl(x)$

$$\tilde{y}_1 := \cos \hat{x}$$

$$\hat{y}_1 := fl(\tilde{y}_1)$$

$$\tilde{y} = 1 - \hat{y}_1 \quad (18)$$

$$\text{και } \hat{y} = fl(\tilde{y})$$

Τα απόλυτα σχετικά βφάλματα είναι:

$$\text{Σφάλμα στρογγύλευσης: } \left| \frac{\hat{x} - x}{x} \right| \leq \epsilon_M \quad (19)$$

$$\text{και έχουμε και ότι: } \left| \frac{\tilde{y}_1 - y_1}{y_1} \right| \leq 4 \epsilon_M \quad (20)$$

όπου $C_1 = \left| \frac{x f'(x)}{f(x)} \right| = \left| \frac{x(-\sin x)}{\cos x} \right| = \left| \frac{x \sin x}{\cos x} \right| \stackrel{(*)}{\approx} \frac{x^2}{1} = (10^{-5})^2 = 10^{-10}$

(*) $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$

$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$

Άρα, $C_1 = 10^{-10}$ (21)

Από την (16) έχουμε: $\left| \frac{\hat{y}_1 - y_1}{y_1} \right| \leq C_1 \epsilon_M + \epsilon_M$ (22)

και επίσης ισχύει ότι:

$$\begin{aligned} \left| \frac{\tilde{y} - y}{y} \right| &\stackrel{(17)(18)}{=} \left| \frac{1 - \hat{y}_1 - (1 - y_1)}{1 - y_1} \right| = \left| \frac{y_1 - \hat{y}_1}{1 - y_1} \right| = \left| \frac{y_1 - \hat{y}_1}{y_1} \cdot \frac{y_1}{1 - y_1} \right| = \\ &= \left| \frac{\hat{y}_1 - y_1}{y_1} \cdot \frac{y_1(-1)}{1 - y_1} \right| = \left| \frac{\hat{y}_1 - y_1}{y_1} \right| \left| \frac{y_1(-1)}{1 - y_1} \right| \stackrel{(22)}{\leq} \stackrel{(24)}{C_2} (C_1 \epsilon_M + \epsilon_M) \quad (23) \end{aligned}$$

όπου $C_2 = \left| \frac{y_1(-1)}{1 - y_1} \right| = \left| \frac{\cos x}{1 - \cos x} \right| = \left| \frac{\cos x}{2 \sin^2 \frac{x}{2}} \right| \stackrel{(*)}{\approx} \frac{1}{2 \frac{x^2}{4}} = \frac{2}{x^2} = \frac{2}{(10^{-5})^2} =$

$$= \frac{2}{10^{-10}} = 2 \cdot 10^{10} \quad (24)$$

Άρα, το πρόβλημα είναι καίως ορισμένο και ο αλγόριθμος είναι αριθμητικά ασταθής. Τελικά έχουμε:

Συνολικό σφάλμα = Σφάλμα εσφαλμένου + Σφάλμα αποκοπής

$$\begin{aligned} &= \left| \frac{\hat{x} - x}{x} \right| + \left| \frac{\tilde{y} - y}{y} \right| \leq \epsilon_M + C_2 (C_1 \epsilon_M + \epsilon_M) \approx \epsilon_M + 2 \cdot 10^{10} \cdot 10^{-10} \epsilon_M + 2 \cdot 10^{10} \epsilon_M \\ &\approx 3 \epsilon_M + 2 \cdot 10^{10} \epsilon_M < 3 \cdot 10^{-16} + 2 \cdot 10^{10} \cdot 10^{-16} \approx 2 \cdot 10^{-6} \end{aligned}$$

2ος Αλγόριθμος

Είναι $y = f(x) := 1 - \cos x \stackrel{(*)}{=} 2 \sin^2 \frac{x}{2} \quad (25)$

$(*)$ Είναι $\cos 2x = 1 - 2 \sin^2 x \Leftrightarrow \cos x = 1 - 2 \sin^2 \left(\frac{x}{2}\right) \Leftrightarrow 1 - \cos x = 2 \sin^2 \left(\frac{x}{2}\right)$.

Οπότε με σύνθεση συναρτήσεων η (25) αναλύεται ως

$$y_1 := \frac{x}{2}, \quad y_2 := \sin y_1, \quad y_3 := y_2^2 \quad \text{και} \quad y_4 := 2y_3.$$

Ο πολλαπλασιασμός επί 2 και η διαίρεση με 2 είναι αριθμείς σε αριθμητική μηχανή, άρα το πρώτο και το τέταρτο βήμα δεν έχουν σφάλμα στρογγύλευσης

$$G = \frac{y_1 \cos y_1}{y_1} = \cos y_1 \stackrel{(*)}{\simeq} 1$$

και ανάδοχα με τον Αλγόριθμο 1 έχουμε

$$\left| \frac{\hat{x} - x}{x} \right| + \left| \frac{\tilde{y} - y}{y} \right| \leq G_2(G_1 \epsilon_M + \epsilon_M) + \epsilon_M \simeq 2(\epsilon_M + \epsilon_M) + \epsilon_M \simeq 5 \cdot 10^{-16}$$

Άρα, αυτός ο αλγόριθμος είναι αριθμητικά ευσταδής.

Παράδειγμα - Γε των Υγέρων (Backward) Ανάλυση Σφάλματος

Έστω $y = f(x)$ ο ακριβής υπολογισμός και $\hat{y} = \hat{f}(\hat{x})$ ο αλγόριθμος σε αριθμητική μηχανή. Αν μπορούσαμε να βρούμε έναν αριθμό x_* κοντά στο $\hat{x}$ τέτοιο ώστε $\hat{y} = f(x_*)$ τότε έχουμε:

$$\begin{aligned} \left| \frac{\hat{y} - y}{y} \right| &= \left| \frac{f(x_*) - f(x)}{f(x)} \right| \stackrel{\text{ΘΗΤ}}{\approx} \left| \frac{(x - x_*) f'(x)}{f(x)} \right| \stackrel{*, 1/x}{=} \left| \frac{x f'(x)}{f(x)} \frac{x - x_*}{x} \right| = \\ &= \left| \frac{x f'(x)}{f(x)} \right| \left| \frac{x - x_*}{x} \right| \leq |c_f| \cdot \left| \frac{x - x_*}{x} \right| \end{aligned}$$

Αρα $\left| \frac{\hat{y} - y}{y} \right| \leq |c_f| \cdot \left| \frac{x - x_*}{x} \right| \quad (26)$

Αν $\left| \frac{x - x_*}{x} \right| \leq \epsilon$, όπου το ϵ δεν είναι πολύ μεγαλύτερο από το ϵ_M , τότε ο αλγόριθμος είναι ευσταθής.

Παράδειγμα - Γε των Υγέρων (Backward) Ανάλυση Σφάλματος

Έστω $y = f(x) = x^2$. Ο απ'ευθείας υπολογισμός δίνει $\hat{x} = fl(x)$, $\tilde{y} = \hat{x}^2$, $\hat{y} = fl(\tilde{y})$, όπου $\hat{y} = \tilde{y}(1 + \epsilon)$ με $|\epsilon| \leq \epsilon_M$.

Αρα, $\hat{y} = x_*^2$ για $x_* = \sqrt{\tilde{y}} \sqrt{1 + \epsilon} = \hat{x} \sqrt{1 + \epsilon}$. Αφού $\left| \frac{\hat{x} - x}{x} \right| \leq \epsilon_M$

και $\sqrt{1 + \epsilon} \approx 1 + \frac{\epsilon}{2}$ έχουμε

$$\begin{aligned} \left| \frac{x_* - x}{x} \right| &= \left| \frac{\hat{x} \sqrt{1 + \epsilon} - x}{x} \right| \approx \left| \frac{\hat{x} (1 + \frac{\epsilon}{2}) - x}{x} \right| = \left| \frac{(\hat{x} - x) + \frac{\hat{x}}{2} \epsilon}{x} \right| \leq \\ &\leq \left| \frac{\hat{x} - x}{x} \right| + \left| \frac{\hat{x}}{x} \frac{\epsilon}{2} \right| \leq \epsilon_M + \frac{1}{2} \epsilon_M \end{aligned}$$

Επίσης

$$\left| \frac{\hat{x}}{x} \right| = \frac{f(x)}{x} = \frac{x(1+\epsilon_M)}{x} = 1+\epsilon_M$$

$$\text{και } (1+\epsilon_M) \frac{\epsilon}{2} \approx \epsilon_M \frac{\epsilon}{2} < \frac{\epsilon_M}{2}.$$

Το οποίο δεν είναι πολύ μεγαλύτερο από το ϵ_M , άρα ο
αλγόριθμος είναι αριθμητικά ευσταδής.