Newton-Raphson

Η Μέθοδος Newton-Raphson (50 Μονάδες)
Η εύρεση ριζών σε πολυώνυμα 2ου βαθμού είναι κάτι που όλοι λίγο-πολύ περάσαμε στο σχολείο συνοδευόμενο από ευχάριστες ή τραυματικές αναμνήσεις. Ο υπολογισμός της διακρίνουσας ($\Delta = \beta^2 - 4\alpha\gamma$) μας εντυπώθηκε στην μνήμη (για μερικούς από εμάς μέχρι τις εξετάσεις και μετά διεγράφη) και η χρησιμότητά της ήταν ότι μας επέτρεπε να υπολογίσουμε τις ρίζες σε κλειστή μορφή για οποιοδήποτε πολυώνυμο 2ου βαθμού. Παρόλα αυτά, το να βρίσκεις ρίζες σε πολυώνυμα είναι δύσκολη υπόθεση. Η κλειστή μορφή για πολυώνυμα 3ου βαθμού είναι σαφώς πιο δύσκολη (για μια γρήγορη αναδρομή που περιέχει προδοσία, εγωισμούς, ζήλια, γεωμετρία και φανταστικούς αριθμούς δείτε εδώ). Για πολυώνυμα 5ου βαθμού και πάνω, είναι ακόμα χειρότερα - καθώς δεν υπάρχει κλειστή μορφή υπολογισμού!
Δυστυχώς (ή ευτυχώς ανάλογα με την οπτική), πολυώνυμα 5ου βαθμού και πάνω υπάρχουν στην φυσική, την μηχανική, και άλλα πεδία και πολύ συχνά πρέπει να υπολογίσουμε τις ρίζες τους. Η εύρεση ριζών σε αυτά τα πολυώνυμα ήταν κάτι άπιαστο για την ανθρωπότητα για το μεγαλύτερο μέρος της ιστορίας μας. Σήμερα, μπορούμε να κάνουμε κάτι για να λύσουμε τέτοια προβλήματα; Ευτυχώς ναι! Η απάντηση είναι στις προσεγγιστικές μεθόδους που προσφέρει η αριθμητική ανάλυση.
Η μέθοδος Newton-Raphson, γνωστή και πιο απλά ως μέθοδος Newton, προς τιμήν του γνωστού μας Νεύτωνα μας επιτρέπει να βρίσκουμε προσεγγιστικές λύσεις σε πραγματικές συναρτήσεις. Η μέθοδος Newton ορίζεται ως εξής: Δεδομένης μιας πραγματικής συνάρτησης $f(x)$ και της παραγώγου της $f\prime(x)$, ξεκινώντας με ένα τυχαίο x_0 μία καλύτερη προσέγγιση μιας ρίζας του πολυωνύμου x_1 δίνεται από την σχέση:

$$x_1 = x_0 - \frac{f(x_0)}{f\prime(x_0)}$$

Αντίστοιχα, η γενική αναδρομική σχέση της μεθόδου του Νεύτωνα είναι:

$$x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)}$$

όπου x_{n+1} είναι η προσεγγιστική τιμή μιας ρίζας της συνάρτησης $f(x)$ μετά από $n+1$ επαναλήψεις.

Για το ζητούμενο αυτής της άσκησης, καλείστε να γράψετε ένα πρόγραμμα που υπολογίζει αυτόματα τις ρίζες (τα x για τα οποία μηδενίζεται) οποιουδήποτε πολυωνύμου 5ου βαθμού μέσα σε δευτερόλεπτα. Σκεφτείτε πόσο χρόνο θα είχαμε γλυτώσει στο σχολείο αν είχαμε πρόσβαση σε ένα τέτοιο πρόγραμμα!Τεχνικές Προδιαγραφές

• C Filepath: newton.c
• Το πρόγραμμά θα πρέπει να παίρνει επτά ορίσματα από την γραμμή εντολών στην μορφή `./newton a0 a1 a2 a3 a4 a5 x0`. Για την σημασιολογία των ορισμάτων, δείτε παρακάτω την χρήση του προγράμματος. Για οποιαδήποτε είσοδο δεν είναι μέσα στις προδιαγραφές το πρόγραμμα πρέπει να επιστρέφει με κωδικό εξόδου (exit code) 1.
• Για ένα πολυώνυμο 5ου βαθμού $a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ και μια αρχική ρίζα x_0 το πρόγραμμα θα εκτελεστεί ως εξής:
`$ ./newton a_0 a_1 a_2 a_3 a_4 a_5 x_0`
• Όλες οι παράμετροι θα είναι σε δεκαδική μορφή τύπου double. Συνιστούμε να χρησιμοποιήσετε την συνάρτηση βιβλιοθήκης strtod για να κάνετε την μετατροπή από όρισμα σε double. Δεν χρειάζεται να ελέγξετε για συνθήκες σφάλματος κατά την μετατροπή σε double - εγγυώμαστε ότι όλα τα δεδομένα εισόδου θα είναι αριθμοί και ότι οι αριθμοί αυτοί θα είναι επαρκώς μικροί για να χωράνε σε double.
• Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας πρέπει να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):
gcc -O3 -Wall -Wextra -Werror -pedantic -o newton newton.c -lm
• README Filepath: newton/README.md
• Ένα αρχείο που να περιέχει στοιχεία εισόδου και ένα εξόδου διαφορετικά από αυτά της άσκησης. Συγκεκριμένα προτείνουμε να βάλετε έναν συνδυασμό που θεωρείται ότι είναι δύσκολος να γίνει σωστός.
– input Filepath: newton/test/input.
– output Filepath: newton/test/output.
Παράδειγμα που όμως δεν θα γίνει δεκτό από την άσκηση επειδή είναι ήδη στα παραδείγματα παρακάτω, για το input: 1.0 0.0 1.0 0.0 0.0 0.0 2.0 και για το output: incomplete.
• Όλα τα αποτελέσματα θα πρέπει να είναι δεκαδικοί με δύο ψηφία ακριβείας και πρόσημο.
• Συνθήκες τερματισμού: ο αλγόριθμος πρέπει να τερματίσει όταν:

1. Ο αλγόριθμος έχει συγκλίνει και ισχύει: $|x_n + 1 - x_n| < 10^{-6}$. Σε αυτήν την περίπτωση τυπώνουμε το αποτέλεσμα $x_n + 1$ με ακρίβεια δύο δεκαδικών ψηφίων και πρόσημο.
2. Ο αλγόριθμος αποκλίνει (π.χ., διαίρεση με 0). Σε αυτήν την περίπτωση τυπώνουμε το αποτέλεσμα nan.
3. Ο αλγόριθμος δεν τερματίζει μετά από 1000 επαναλήψεις. Σε αυτήν την περίπτωση τυπώνουμε το αποτέλεσμα incomplete.

• Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 10 δευτερόλεπτα.

Παρακάτω παραθέτουμε την αλληλεπίδραση με μια ενδεικτική λύση:

```
thanassis@linux14:~$ hostname
linux14
thanassis@linux14:~$ gcc -O3 -Wall -Wextra -Werror -pedantic -o newton
newton.c -lm
thanassis@linux14:~$ ./newton 1.0 2.0 3.0 4.0 5.0 6.0 1.0
-0.67
thanassis@linux14:~$ ./newton 1.0 0.0 1.0 0.0 0.0 0.0 2.0
incomplete
thanassis@linux14:~$ ./newton 1.0 0.0 1.0 0.0 0.0 0.0 0.0
nan
```

Μπορούμε να επιβεβαιώσουμε τα αποτελέσματα των υπολογισμών μας αποτιμώντας το πολυώνυμο. Για παράδειγμα, το $a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ για $x = -06$ κάνει 0.00112899 (σχεδόν 0 - μπορούμε να πάρουμε ρίζες που προσεγγίζουν το μηδέν καλύτερα αυξάνοντας την ακρίβεια). Δοκιμάστε και εσείς με τα δικά σας πολυώνυμα για να δείτε αν όντως το πρόγραμμά σας τα καταφέρνει!

Στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης.