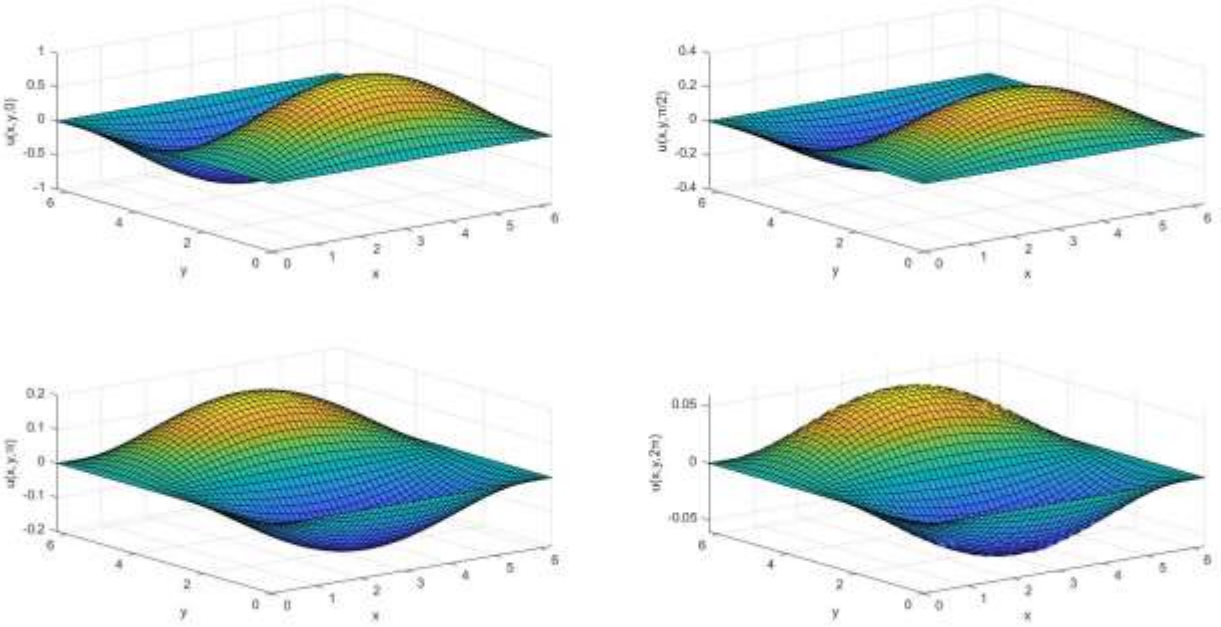
Εργαλεία προγραμματισμού - OpenMP - Σετ ασκήσεων

Ελεάννα Χωραΐτη Σιδέρη

ΑΕΜ 4406

1.

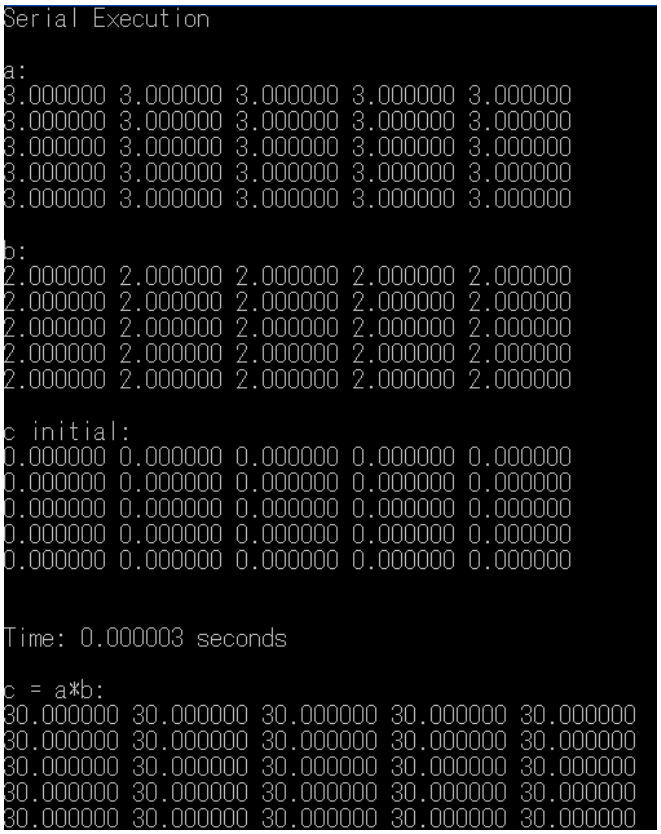
Α. Ο κώδικας που αντιστοιχεί στο ερώτημα είναι ο exe1.c. Επιλύει τη εξίσωση που δόθηκε και εξάγει τα αποτελέσματα για 4 συγκεκριμένες χρονικές στιγμές. Τα αποτελέσματα της λύσης σε 2 διαστάσεις σχεδιάστηκαν στα παρακάτω διαγράμματα.

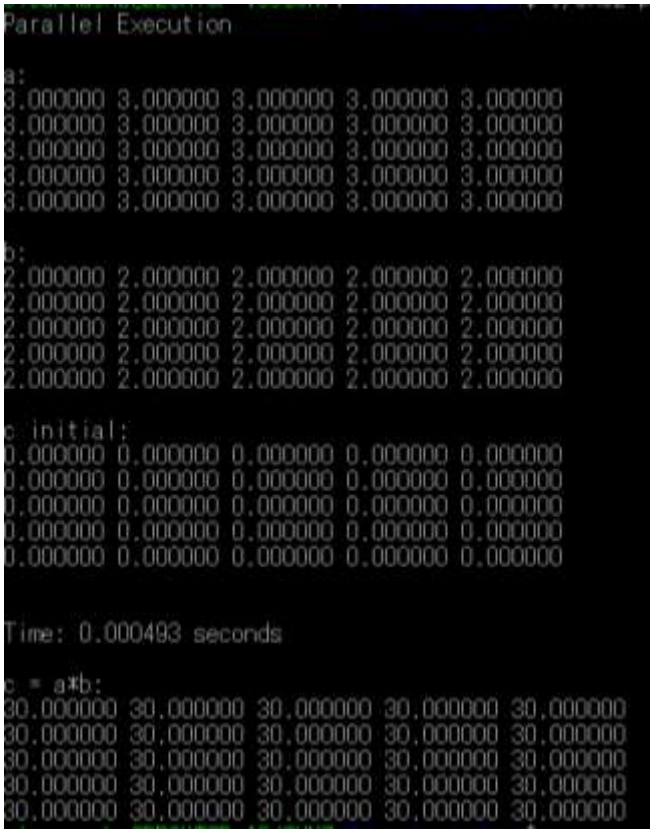


2.

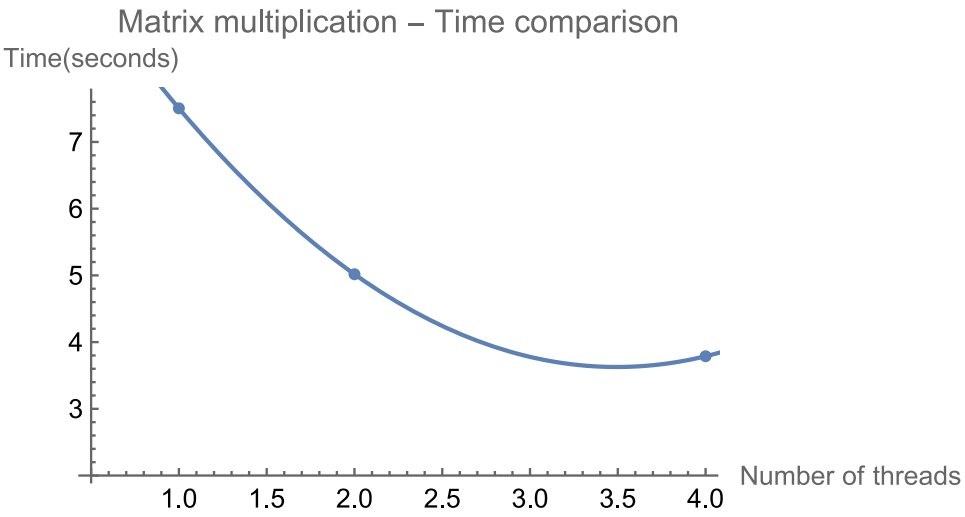
Οι κώδικες που αντιστοιχούν στην άσκηση είναι οι exe2-serial.c και exe2-parallel.c που περιέχουν το πρόγραμμα πολλαπλασιασμού πινάκων σειριακά και παράλληλα αντίστοιχα.

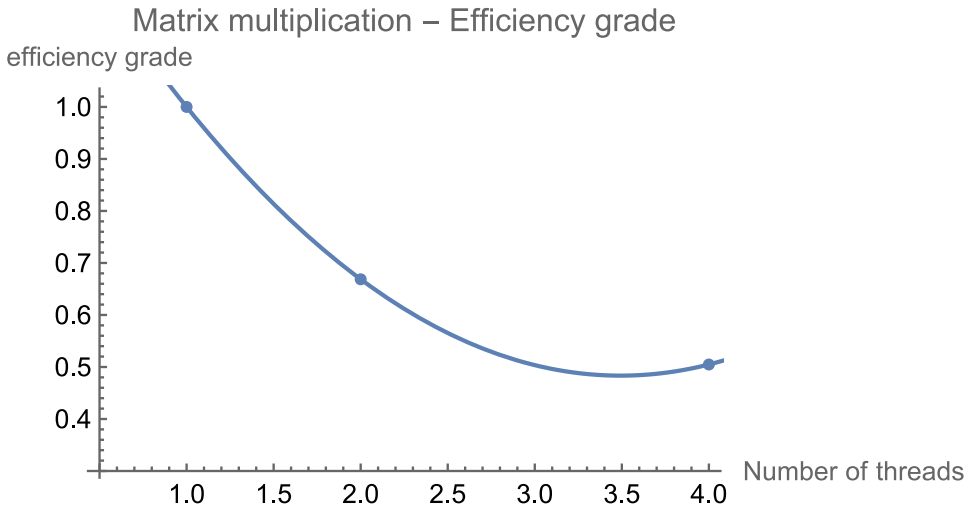
Τα τρέχουμε για 2 πίνακες a και b 5*5 ώστε να ελέγξουμε το αποτέλεσμα. Τα αποτελέσματα παρουσιάζονται στις παρακάτω εικόνες:





Αφού σιγουρευτήκαμε πως το παράλληλο πρόγραμμα έχει τα ίδια αποτελέσματα με το σειριακό εκτελούμε το παράλληλο πρόγραμμα για 2 πινακες a, b μεγέθους 1000*1000 ώστε να ελέγξουμε την απόδοση για διαφορετικούς αριθμούς threads. Εκτελούμε για αριθμό threads 1, 2 και 4. Στα παρακάτω γραφήματα παρουσιάζονται ο χρόνος και ο βαθμός παράλληλης απόδοσης. Ως βαθμό παράλληλης απόδοσης ορίζουμε το κλάσμα του χρόνου εκτέλεσης του παράλληλου προγράμματος σε 1 thread προς το χρόνο εκτέλεσης του παράλληλου σε παραπάνω threads. Δηλαδή αν η απόδοση είναι ίση με 0,5 τότε σημαίνει πως το παράλληλο πρόγραμμα εκτελεί την διαδικασία στο μισό χρόνο απ’ ότι θα έκανε σε 1 thread.


Matrix multiplication – Time comparison
Time(seconds)
7
6
5
4
3
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Number of threads


Matrix multiplication – Efficiency grade
efficiency grade
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Number of threads

3.

Ο κώδικας που αντιστοιχεί στην άσκηση είναι ο exe3.c. Γράψαμε ένα πρόγραμμα που να εκτελεί τη μέθοδο της διχοτόμησης για την εύρεση της ρίζας μια εξίσωσης. Κωδικά η μέθοδος της διχοτόμησης ακολουθεί τα παρακάτω βήματα για την εύρεση μιας ρίζας.

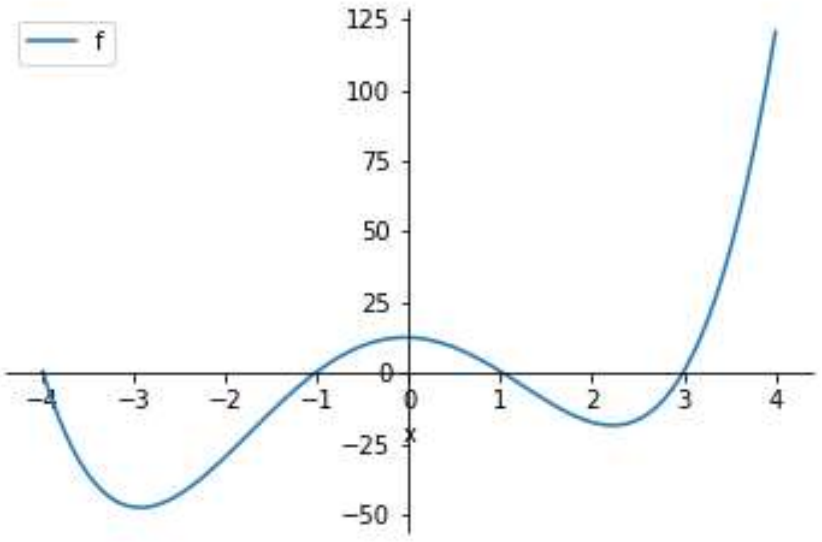
1. Εάν μια ρίζα βρίσκεται στο διάστημα [α,β] τότε θα ισχύει f(α)*f(β)<0. Βρίσκουμε το μέσο του διαστήματος $\mu = \frac{\alpha+\beta}{2}$.
2. Ελέγχουμε το πρόσημο του γινομένου f(α)*f(μ). Διακρίνουμε 3 περιπτώσεις¨
    - Αν f(α)*f(μ)<0 τότε η ρίζα βρίσκεται στο διάστημα [α,μ], οπότε θέτουμε β=μ.
    - Αν f(α)*f(μ)>0 τότε η ρίζα βρίσκεται στο διάστημα [μ,β], οπότε θέτουμε α=μ.
    - Αν f(α)*f(μ)=0 τότε η ρίζα ισούται με το μ
3. Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να επιτευχθεί μια ορισμένη ακρίβεια στον υπολογισμό της ρίζας.

Σημειώνεται πως για να γνωρίζουμε το διάστημα στο οποίο πρέπει να αναζητήσουμε τη ρίζα σχεδιάζουμε ένα γράφημα της συνάρτησης.

Για την άσκηση γράψαμε ένα πρόγραμμα που να εκτελεί την μέθοδο της διχοτόμησης παράλληλα. Το παράλληλο κομμάτι αφορά στο χωρισμό του διαστήματος αναζήτησης της ρίζας σε μικρότερα τμήματα ώστε η αναζήτηση να γίνεται παράλληλα από όλα τα threads. Το πρόγραμμα χωρίζει το διάστημα αναζήτησης σε τόσα τμήματα όσα και τα threads που επιλέγουμε. Η εξίσωση της οποίας επιλέγουμε να βρούμε τη ρίζα είναι η:

$$f(x) = (x+4) * (x+1) * (x-1) * (x-3) + 0.5$$

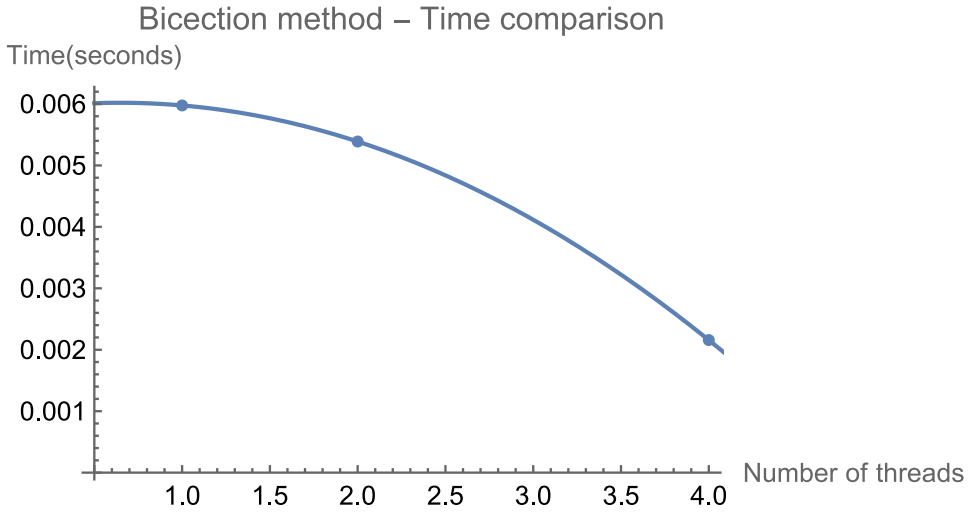
Είναι πολυωνυμική εξίσωση 4ου βαθμού και αν τη σχεδιάσουμε θα δούμε πως έχει 4 ρίζες μεταξύ του διαστήματος [-4,4]. Το γράφημα είναι το παρακάτω:



Για να ελέγξουμε αν η παράλληλη επεξεργασία του προγράμματος είναι πιο αποδοτική χρονικά τρέξαμε τη μέθοδο της διχοτόμησης για το διάστημα [-100,-2] ώστε να προσεγγίσει τη ρίζα που βρίσκεται κοντά στο χ=-4. Τρέξαμε τον κώδικα για 1, 2 και 4 threads και η εκτύπωση στην οθόνη ήταν η παρακάτω. Βλέπουμε πως το πρόγραμμα τυπώνει την ταυτότητα του κάθε thread, τα διαστήματα στα οποία έψαξε τη ρίζα, και ένα μήνυμα, είτε πως δεν υπάρχει ρίζα στο συγκεκριμένο διάστημα, είτε την τιμή της ρίζας, εφόσον την βρήκε. Επιπλέον τυπώνει τον χρόνο που πέρασε μέχρι να βρεθεί η κάθε ρίζα.

```
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 1
Interval is (-75.500000,-51.000000)

Thread num: 0
Interval is (-100.000000,-75.500000)
There is no root in the interval
There is no root in the interval

Thread num: 2
Interval is (-51.000000,-26.500000)
There is no root in the interval

Thread num: 3
Interval is (-26.500000,-2.000000)
Root is -3.995223

Time: 0.002158 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ export OMP_NUM_THREADS=2
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (-100.000000,-51.000000)

Thread num: 1
Interval is (-51.000000,-2.000000)
There is no root in the interval
Root is -3.995223

Time: 0.005389 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ export OMP_NUM_THREADS=1
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (-100.000000,-2.000000)
Root is -3.995223

Time: 0.005976 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$
```

Παρακάτω παρουσιάζεται ο χρόνος εκτέλεσης της παράλληλης περιοχής για τους διάφορους αριθμούς threads:



Παρόλο που οι χρόνοι είναι πολύ μικροί (εφόσον και η μέθοδος της διχοτόμησης είναι μια σχετικά απλή μέθοδος προγραμματιστικά) η μείωση του χρόνου είναι αισθητή όσο τα threads αυξάνονται.

Για την αποδοτικότητα της παράλληλης επεξεργασίας στη μέθοδο της διχοτόμησης πραγματοποιήσαμε και έναν ακόμη έλεγχο. Επιδιώξαμε να υπολογίσουμε τις 4 ρίζες της συνάρτησης ταυτόχρονα για 4 threads και ανά 2 για 2 threads. Κάτι τέτοιο δεν είναι εφικτό προφανώς για μία σειριακή επεξεργασία της διχοτόμησης καθώς για κάθε ρίζα είναι απαραίτητο να τρέξουμε εκ νέου τον κώδικα με διαφορετικά άκρα του αρχικού διαστήματος. Τρέξαμε έτσι το πρόγραμμα για 4 threads 1 φορα για το διάστημα [-4,4]. Έπειτα 2 φορές για 2 threads και για τα διαστήματα [-4,0] και [0,-4]. Και τέλος 4 φορές για 1 thread και για τα διαστήματα [-4,-2], [-2,0], [0,2], [2,4]. Τα αποτελέσματα παρουσιάζονται παρακάτω:

```
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ export OMP_NUM_THREADS=4
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 2
Interval is (0.000000,2.000000)

Thread num: 1
Interval is (-2.000000,0.000000)

Thread num: 3
Interval is (2.000000,4.000000)

Thread num: 0
Interval is (-4.000000,-2.000000)
Root is 1.024880

Root is 2.990999

Root is -1.020657

Root is -3.995223

Time: 0.016481 seconds
```

```
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ export OMP_NUM_THREADS=2
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (-4.000000,-2.000000)

Thread num: 1
Interval is (-2.000000,0.000000)
Root is -3.995223

Root is -1.020657

Time: 0.005489 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (0.000000,2.000000)

Thread num: 1
Interval is (2.000000,4.000000)
Root is 1.024880

Root is 2.990999

Time: 0.005470 seconds
```

```
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ export OMP_NUM_THREADS=1
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (-4.000000,-2.000000)
Root is -3.995223

Time: 0.000616 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (-2.000000,0.000000)
Root is -1.020657

Time: 0.000974 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (0.000000,2.000000)
Root is 1.024880

Time: 0.001030 seconds

eleannachs@DESKTOP-I6JOUN7:~/projects/set$ gcc exe3.c -lm -o exe3 -fopenmp
eleannachs@DESKTOP-I6JOUN7:~/projects/set$ ./exe3

Thread num: 0
Interval is (2.000000,4.000000)
Root is 2.990999

Time: 0.002374 seconds
```

Βλέπουμε πως το πρόγραμμα κατάφερε να υπολογίσει τις ρίζες ταυτόχρονα για 4 threads ωστόσο δεν μπορούμε να είμαστε σίγουροι πως το ίδιο θα επιτευχθεί για κάποια άλλη συνάρτηση της οποίας οι ρίζες δεν θα είναι τόσο ομοιόμορφα κατανεμημένες.