Μαρία Αναστασία Ρούνη

08.03.2019

Άσκηση 1

Θέλουμε να βρούμε τη λύση του ελλειπτικού προβλήματος συνοριακών τιμών που καθορίζεται από τη διαφορική εξίσωση

$$u_{xx} + u_{yy} = -10(x^2 + y^2 + 5)$$

με συνοριακές συνθήκες $u(0, y) = u(1, y) = u(x, 0) = 0$ και $u(x, 1) = 1$.
Αρχικά χρησιμοποιούμε τη μέθοδο *Liebmann*, σε ορθοκανονικό πλέγμα με όρια $(0, 1)X(0, 1)$ το οποίο διακριτοποιούμε σε $NXM = 400X400$ σημεία.

Στην ουσία το πρόβλημα που μελετάμε είναι η εξίσωση *Poisson* σε δύο διαστάσεις :

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = S(x, y)$$

ενώ η μέθοδος *Liebmann* ορίζει πως κατά την επανάληψη $n + 1$, η τιμή της συνάρτησης στο σημείο (i, j) εξαρτάται από τις τιμές της συνάρτησης στα σημεία $(i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1)$ και από την τιμή της S στο σημείο αυτό, όπως υπολογίστηκαν κατά την προηγούμενη επανάληψη n :

$$f_{i,j}^{n+1} = \frac{1}{4}[f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j+1}^{n} + f_{i,j-1}^{n} - h^2 S_{i,j}]$$

h είναι η απόσταση των σημείων κατά τη διεύθυνση i , $h = (xmax - xmin)/(N - 1)$. Αντίστοιχα ορίζουμε βήμα k κατά τη διεύθυνση j.

Ο κώδικας που υλοποιήσαμε δημιουργεί δύο διανύσματα όπου κρατούνται οι τιμές των διαδοχικών επαναλήψεων. Αρχικά με μια επαναληπτική διαδικασία δίνουμε τις συνοριακές τιμές όπως ορίζει το πρόβλημα που θέσαμε, ενώ κάθε ενδιάμεση τιμή παίρνει την τιμή μηδέν.
Στη συνέχεια με μια δεύτερη επαναληπτική διαδικασία, όσο το σφάλμα είναι μεγαλύτερο από μια αποδεκτή ακρίβεια, υπολογίζουμε την λύση της εξίσωσης *Poisson* σε κάθε σημείο του πλέγματος. Το σφάλμα σε κάθε επανάληψη ορίζεται ως ο μέσος όρος του αθροίσματος της απόλυτης μεταβολής από μια επανάληψη έως την επόμενη για όλα τα σημεία του πλέγματος:

$$tolerance = \frac{1}{(N-2)(M-2)} \sum_{i,j} |u_{i,j}^{new} - u_{i,j}^{old}|$$

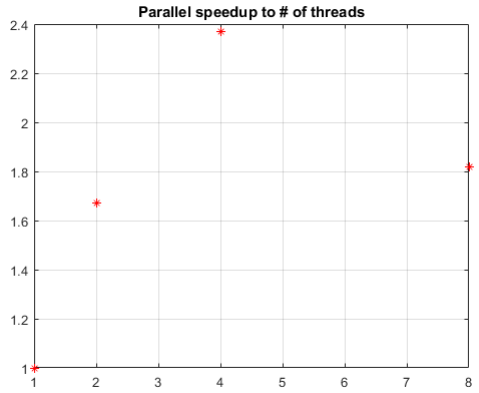
Στόχος μας ακρίβεια της τάξης 10^{-7}.

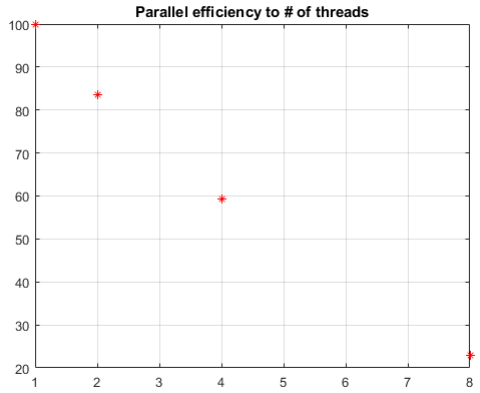
Το πρόγραμμα παραλληλοποιήθηκε με γραμμές εντολών *OpenMP* και έτρεξε σε 1,2,4 και 8 πυρήνες. Παρακάτω φαίνεται ο πίνακας των αποτελεσμάτων για κάθε αριθμό *threads* με τον αριθμό των επαναλήψεων που χρειάστηκαν, την τιμή του κεντρικού σημείου και την χρονική διάρκεια που έτρεξε το πρόγραμμα.

#threads	**#iterations**	**(N/2,M/2)**	**elapsed time (sec)**
1	207,846	4.329148	93.884
2	207,846	4.329148	56.207
4	207,846	4.329148	39.680
8	207,846	4.329148	51.572

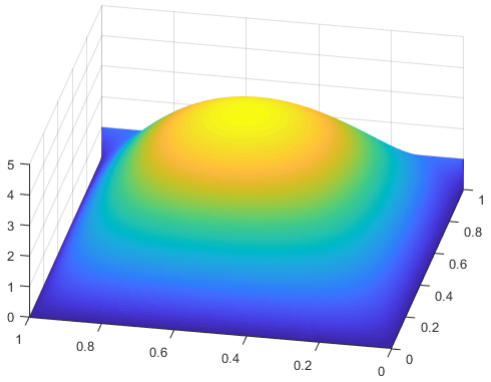
Υπολογίσαμε επίσης την επιτάχυνση της παράλληλης επεξεργασίας parallel speedup και την απόδοση της παράλληλης διαδικασίας parallel efficiency για κάθε εκτέλεση και δημιουργήσαμε τα αντίστοιχα διαγράμματα ως προς τον αριθμό των *threads*.

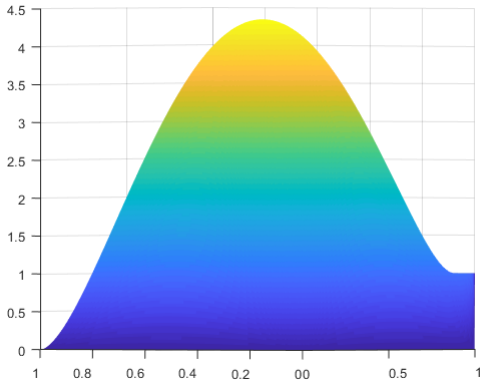
#threads	**speedup**	**efficiency %**
1	1	100
2	1.67	83.5
4	2.37	59.3
8	1.82	22.8

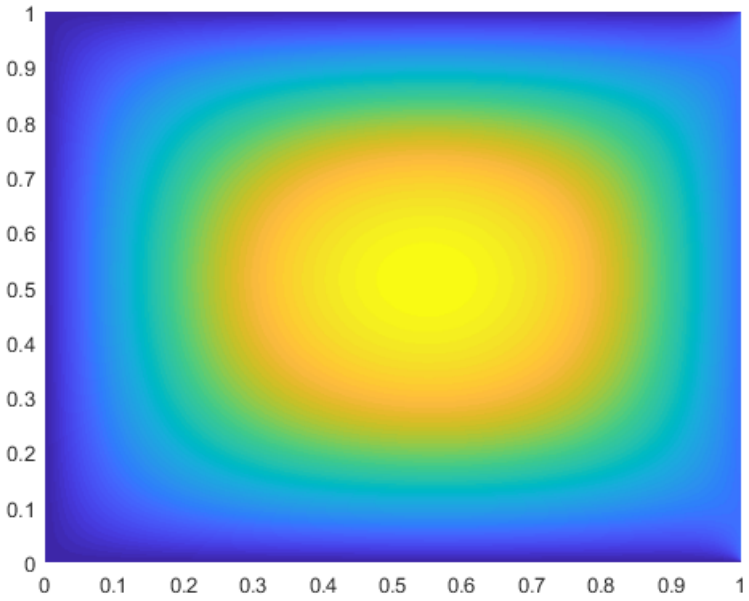




Τέλος, σχεδιάσαμε τη λύση $u(x, y)$ ως επιφάνεια :







Άσκηση 2

Στο δεύτερο μέρος της εργασίας, λύσαμε την διαφορική εξίσωση με την μέθοδο $Gauss-Siedel$ που επιταχύνει την διαδικασία αφού χρησιμοποιεί όπου υπάρχουν, τις τιμές των σημείων που έχουν υπολογιστεί ήδη για την τρέχουσα επανάληψη. Το πρόβλημα που προκύπτει κατά την παράλληλη επεξεργασία είναι πως ο διαμερισμός των εργασιών στα $threads$ γίνεται 'γραμμικά' δηλαδή κάθε πυρήνας αναλαμβάνει ένα πλήθος γραμμών i του πλέγματος. Αυτό θα έχει σαν αποτέλεσμα, οι τιμές της τρέχουσας επανάληψης σε σημεία που χρειάζεται να γνωρίζει ένας πυρήνας, να μην έχουν υπολογιστεί από τον πυρήνα που επιφορτώθηκε με τον υπολογισμό τους.

Για τον λόγο αυτό, χρησιμοποιούμε τον αλγόριθμο $red-black$ κατά την εκτέλεση του οποίου με παράλληλη επεξεργασία υπολογίζονται τα κεντρικά σημεία των επιμέρους υπο-πλεγμάτων μεγέθους 5 στοιχείων (σταυρός), που αντιστοιχούν σε δείκτες (i, j) με άθροισμα περιττό και στη συνέχεια με μια δεύτερη παράλληλη επαναληπτική επεξεργασία, υπολογίζονται οι τιμές των υπολοίπων σημείων, χρησιμοποιώντας το διάνυσμα $uNew$ που δημιουργηθηκε στο προηγούμενο βήμα. Ο αλγόριθμος αναλύεται λοιπόν σε δυο βήματα:

$$1) f_{i.j}^{n+1} = (1-\omega) f_{i,j}^{n} + \frac{\omega}{4}[f_{i+1,j}^{n} + f_{i-1,j}^{n} + f_{i,j+1}^{n} + f_{i,j-1}^{n} - h^2 S_{i,j}] : (i + j\%2 == 1)$$

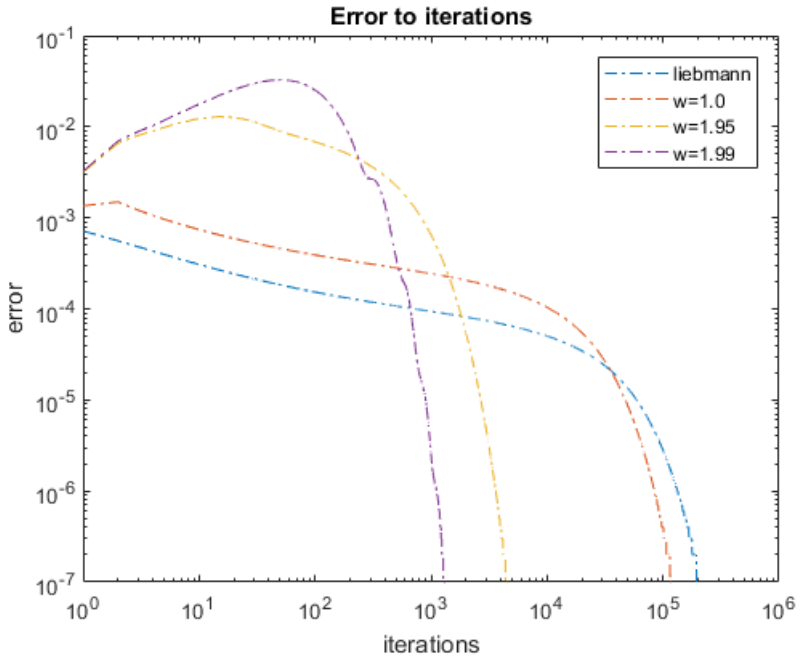
$$2) f_{i.j}^{n+1} = (1-\omega) f_{i,j}^{n} + \frac{\omega}{4}[f_{i+1,j}^{n+1} + f_{i-1,j}^{n+1} + f_{i,j+1}^{n+1} + f_{i,j-1}^{n+1} - h^2 S_{i,j}] : (i + j\%2 == 0)$$

όπου ω είναι ο παράγοντας επιτάχυνσης της λεγόμενης $SuccessiveOverRelaxation(SOR)$ μεθόδου που στηρίζεται στην $Gauss - Siedel$.

Δοκιμάσαμε την μέθοδο για τιμές του παράγοντα επιτάχυνσης ω = 1, 1.95 και 1.99 για 1,2,4,8 πυρήνες. Παρακάτω φαίνονται συγκεντρωτικά τα αποτελέσματα σε σύγκριση με την μέθοδο *Liebmann* του πρώτου μέρους της εργασίας.

method	**#threads**	**#iterations**	**elapsed time (sec)**	**parallel speedup**
Liebmann	1	207,846	93.884	1
$\omega = 1$		121,645	98.303	1
$\omega = 1.95$		4,543	3.698	1
$\omega = 1.99$		1,318	1.079	1
Liebmann	2	207,846	56.207	1.67
$\omega = 1$		121,645	55.900	1.76
$\omega = 1.95$		4,543	2.108	1.75
$\omega = 1.99$		1,318	0.609	1.77
Liebmann	4	207,846	39.680	2.37
$\omega = 1$		121,645	38.122	2.58
$\omega = 1.95$		4,543	1.338	2.76
$\omega = 1.99$		1,318	0.391	2.76
Liebmann	8	207,846	51.572	1.82
$\omega = 1$		121,645	45.696	2.15
$\omega = 1.95$		4,543	1.814	2.04
$\omega = 1.99$		1,318	0.543	1.99

Δημιουργήσαμε ένα διάγραμμα $log - log$ του σφάλματος κάθε επανάληψης προς τον αριθμό των επαναλήψεων και για τις δυο μεθόδους $(Liebmann, SOR)$ και για τις τρεις τιμές του παράγοντα επιτάχυνσης. Παρακάτω φαίνονται οι τέσσερις καμπύλες:



Η μέθοδος SOR για συντελεστή επιτάχυνσης ω=1 ταυτίζεται για $(i+j\%2==1)$ με την μέθοδο $Gauss-Siedel$ και χρειάζεται περίπου το μισό πλήθος επαναλήψεων σε σύγκριση με την $Liebmann$ (207,000) για να δώσει λύση. Αυτό είναι κάτι σχετικά λογικό εφόσον κάθε επανάληψη κάνει δύο βήματα. Ο χρόνος εκτέλεσης για κάθε πλήθος πυρήνων που έτρεξε είναι και για τις δυο μεθόδους περίπου ίδιος
Για ω=1.95 οι επαναλήψεις της SOR πέφτουν απο 121,500 περιπου στις 4.500 και ο χρονος εκτελεσης απο 98 δευτερόλεπτα σε 3.5 δευτερόλεπτα.
Για ω=1.99 οι επαναλήψεις πέφτουν στις 1,300 και ο χρόνος εκτέλεσης στο 1 δευτερόλεπτο.

Παρατηρούμε πως το πλήθος των επαναλήψεων δεν επηρεάζεται από τον αριθμό των πυρήνων στους οποίους μοιράζονται οι υπολογισμοί. Για 8 πυρήνες ο χρόνος εκτέλεσης αντί να μειωθεί αυξάνεται, πράγμα που είναι λογικό αφού οι φυσικοί πυρήνες του επεξεργαστή ο οποίος έτρεξε το πρόγραμμα είναι 4. Τα χαρακτηριστικά του συστήματος στο οποίο έτρεξαν τα προγράμματα φαίνονται παρακάτω :