ΥΠΟΛΟΓΙΣΤΙΚΑ ΕΡΓΑΛΕΙΑ - OpenMP

ΣΕΤ ΑΣΚΗΣΕΩΝ

ΑΣΚΗΣΗ 1: Να βρεθεί η λύση του ελλειπτικού προβλήματος συνοριακών τιμών που καθορίζεται από τη διαφορική εξίσωση

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

με συνορικές συνθήκες $u(0, y) = u(1, y) = u(x, 0) = 0$ και $u(x, 1) = 1$. Χρησιμοποιείστε την επαναληπτική μέθοδο Liebmann, σε ένα ορθοκανονικό πλέγμα με όρια $(0, 1)\times(0, 1)$ το οποίο θα διακριτοποιήσετε με $N \times M = 400 \times 400$ σημεία. Ως αρχική τιμή για τις επαναλήψεις χρησιμοποιείστε την $u(x, y) = 0$ σε όλο το πλέγμα. Μετά από κάθε επανάληψη, ελέγξτε τον μέσο όρο του αθροίσματος της απόλυτης μεταβολής από μια επανάληψη έως την επόμενη (χρησιμοποιώντας μόνο τα εσωτερικά σημεία)

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

Η επαναληπτική μέθοδος να τερματίζει όταν $\text{tolerance} \leq 10^{-7}$. α) Εκτελέστε το πρόγραμμα για 1, 2, 4, 8 threads και δείξτε σε έναν πίνακα τον αριθμό των επαναλήψεων που απαιτήθηκαν, την τελική τιμή στο κέντρο του πλέγματος $u(N/2, M/2)$ και το χρόνο εκτέλεσης. β) Υπολογίστε την *επιτάχυνση παράλληλης επεξεργασίας* και την *απόδοση παράλληλης επεξεργασίας* για κάθε εκτέλεση και δημιουργήστε διαγράμματα αυτών ως προς τον αριθμό των threads . γ) Σχεδιάστε ως επιφάνεια την λύση $u(x, y)$.

ΑΣΚΗΣΗ 2: Επαναλάβετε την Άσκηση 1, αλλά αυτή τη φορά χρησιμοποιείστε τη μέθοδο Successive Over-Relaxation (SOR) (στη μορφή του αλγόριθμου red-black), με παράγοντα επιτάχυνσης $\omega = 1.0$, $\omega = 1.95$ και $\omega = 1.99$. α) Συγκρίνετε τον αριθμό επαναλήψεων, το χρόνο εκτέλεσης και την επιτάχυνση παράλληλης επεξεργασίας για τις παραπάνω περιπτώσεις με αυτόν της μεθόδου Liebmann. β) Δημιουργείστε ένα διάγραμμα $\log - \log$ για την tolerance ως συνάρτηση του αριθμού των επαναλήψεων (στο ίδιο διάγραμμα δείξτε καμπύλες για τη μέθοδο Liebmann και για την SOR με τις τρεις τιμές του ω που δίνονται παραπάνω). γ) Σχολιάστε αναλυτικά τα παραπάνω αποτελέσματα.

ΣΗΜΕΙΩΣΗ: α) Αναφέρετε τον επεξεργαστή που χρησιμοποιήσατε και τον αριθμό των φυσικών πυρήνων (cores). β) Ως *επιτάχυνση* παράλληλης επεξεργασίας (parallel speedup) ορίζουμε τον λόγο του χρόνου εκτέλεσης σε ένα πυρήνα προς τον χρόνο εκτέλεσης σε πολλούς πυρήνες. Η ιδανική επιτάχυνση είναι ίση με τον αριθμό των πυρήνων. γ) Ως *απόδοση* παράλληλης επεξεργασίας (parallel efficiency) ορίζουμε τον χρόνο εκτέλεσης ως ποσοστό του χρόνου εκτέλεσης σε ένα πυρήνα διαιρεμένο με τον αριθμό των πυρήνων. Η ιδανική απόδοση είναι 100%.

ΘΕΩΡΙΑ : Εξίσωση Poisson σε δύο διαστάσεις :

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

Μέθοδος Liebmann (ή point Jacobi):

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

Μέθοδος Gauss-Seidel:

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

Μέθοδος SOR (μόνο για σειριακή εκτέλεση):

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

Μέθοδος SOR (παράλληλος αλγόριθμος red-black):

1ο βήμα (μόνο για $i + j =$ περιττός):

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

2ο βήμα (μόνο για $i + j =$ άρτιος):

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

ΒΙΒΛΙΟΓΡΑΦΙΑ :

α) H.G. Im, "Numerical Methods for Elliptic Equations", online lectures on Computational Fluid Dynamics I, University of Michigan (2001)

β) D.J. Evans, "Parallel S.O.R. Iterative Methods", *Parallel Computing*, **1**, 3-18 (1984)

ΥΠΟΒΟΛΗ ΤΟΥ ΣΕΤ ΑΣΚΗΣΕΩΝ :

Δημιουργήστε ένα λογαριασμό στο Github και ανεβάστε τους κώδικες (μόνο πηγαίο κώδικα, όχι τα εκτελέσιμα αρχεία) καθώς και την εργασία σε ένα **private** repository. Δώστε δικαίωμα ανάγνωσης στο χρήστη niksterg1@gmail.com και ειδοποιείστε με μέσω email.