ISSIQLIK TENGLAMASINI PARALLEL USUL VA CHEKLI AYIRMALAR USULI BILAN SONLI YECHISH: ALGORITMIK TAHLIL VA SOLISHTIRMA

Tuychiyeva Dilrabo Mansur qizi

FarDU amaliy matematika mutaxassisligi magistranti

Annotatsiya: Ushbu maqolada bir o'lchamli issiqlik tenglamasini sonli yechishda qo'llaniladigan ikki usul — parallel hisoblash usuli va chekli ayirmalar usuli (CAU) — algoritmik jihatdan tahlil qilinadi va o'zaro solishtiriladi. Har ikkala usulning vaqt murakkabligi, xotira sarfi, barqarorlik shartlari va masshtablanish imkoniyatlari ko'rib chiqiladi. Tahlil shuni ko'rsatadiki, parallel usul katta o'lchamli masalalar uchun sezilarli darajada samaraliroq bo'lib, p protsessor soni ortishi bilan hisoblash vaqti proporsional kamayadigan bo'lsa, CAU sodda amalga oshirilishi bilan kichik o'lchamli masalalar uchun afzalroqdir.

Kalit so'zlar: issiqlik tenglamasi, parallel hisoblash, chekli ayirmalar usuli, vaqt murakkabligi, CFL sharti, masshtablanish.

1. Kirish

Issiqlik tarqalishini tavsiflaychi differensial tenglamalar fizika, muhandislik va amaliy matematikaning ko'plab sohalarida muhim ahamiyat kasb etadi. Bunday tenglamalarni analitik usulda yechish aksariyat hollarda qiyin yoki mumkin emas bo'lganligi sababli, sonli usullar keng qo'llaniladi.

Bir o'lchamli issiqlik tenglamasi quyidagi ko'rinishda ifodalanadi:

$\partial u/\partial t = \alpha \cdot \partial^2 u/\partial x^2, \quad (x,t) \in [0,L] \times [0,T]$

bu yerda u(x,t) — temperatura, α > 0 — issiqlik o'tkazuvchanlik koeffitsiyenti, L — fazoviy soha uzunligi, T — vaqt oralig'i.

Ushbu maqolada ikkita sonli yondashuv tahlil qilinadi: (1) chekli ayirmalar usuli (CAU) — klassik ketma-ket algoritm, va (2) parallel hisoblash usuli — tarqatilgan hisoblash arxitekturasi asosidagi yondashuv. Algoritmlarning samaradorligi nazariy tahlil orqali solishtiriladi.

2. Masalaning matematik qo'yilishi va metodlar

2.1. Masalaning matematik qo'yilishi

Quyidagi boshlang'ich-chegara masalasi ko'riladi:

$\partial u/\partial t = \alpha \cdot \partial^2 u/\partial x^2, \quad 0 < x < L, \quad 0 < t \leq T$

$u(x, 0) = f(x), \quad 0 \leq x \leq L \quad (boshlang'ich\ shart)$

$u(0, t) = u(L, t) = 0, \quad 0 < t \leq T \quad (chegara\ shartlari)$

Fazoviy sohani N ta, vaqt oralig'ini M ta kichik bo'laklarga bo'lamiz: h = L/N (fazoviy qadam), τ = T/M (vaqt qadami). Setka nuqtalari: $x_i = i \cdot h$, $t_n = n \cdot \tau$.

2.2. Chekli ayirmalar usuli (CAU)

CAU yorituvchi sxemasi (явная схема) asosida qurilgan. Vaqt hosilasi oldinga chekli ayirma, fazoviy hosilalar esa markaziy chekli ayirma bilan almashtiriladi:

$(u_i^{n+1} - u_i^n) / \tau = \alpha \cdot (u_{i-1}^n - 2u_i^n + u_{i+1}^n) / h^2$

Yechim iterativ ko'rinishda ifodalanadi:

$u_i^{n+1} = u_i^n + r \cdot (u_{i-1}^n - 2u_i^n + u_{i+1}^n), \quad r = \alpha \cdot \tau / h^2$

Barqarorlik sharti (CFL): $r = \alpha \cdot \tau / h^2 \leq 1/2$. Bu shart bajarilmasa, yechim divergensiyaga uchraydi.

Algoritmning vaqt murakkabligi: O(N·M) — har bir vaqt qadamida N ta nuqta ketma-ket hisoblanadi.

Xotira sarfi: O(N) — faqat ikki ketma-ket vaqt qatlamini saqlash zarur.

2.3. Parallel hisoblash usuli

Parallel usulda fazoviy setka p ta protsessor o'rtasida taqsimlanadi. Har bir protsessor N/p ta nuqtani boshqaradi. Chegaradosh protsessorlar o'rtasida aloqa (ghost cell almashish) zarur.

Har bir protsessor p_k uchun hisoblash quyidagicha:

$$u_i^{n+1} = u_i^n + r \cdot (u_{i-1}^n - 2u_i^n + u_{i+1}^n), \quad i \in [k \cdot N/p, (k+1) \cdot N/p)$$

Har bir vaqt qadamida chegaradosh protsessorlar orasida 2 ta qiymat almashiladi. Aloqa vaqti: t_comm = β + γ·2, bu yerda β — boshlang'ich kechikish, γ — bir so'z uzatish vaqti.

Algoritmning vaqt murakkabligi: O(N·M/p + M·t_comm) — ideal holda p marta tezlashish.

Xotira sarfi (har protsessorda): O(N/p) — umumiy xotira sarfi ketma-ket usul bilan bir xil, lekin har bir tugun uchun p marta kamroq.

Tezlashish koeffitsiyenti (Speedup): $S(p) = T_1/T_p \approx p / (1 + p \cdot t_comm/(N \cdot t_calc))$. Aloqa xarajati kichik bo'lsa, S(p) → p.

3. Tahlil va muhokama

Ikki usulning algoritmik solishtirmasi 1-jadvalda keltirilgan:

1-jadval. Parallel usul va CAU ning algoritmik solishtirmasi

Mezon	Parallel usul	Chekli ayirmalar	Izoh
Vaqt murakkabligi	O(N/p + T)	O(N·T)	Parallel usul tezroq
Xotira sarfi	O(N/p)	O(N)	Parallel usul samaraliroq

Barqarorlik	Shartli (CFL)	Shartli (CFL)	Bir xil
Amalga oshirish murakkabligi	Yuqori	O'rta	FDM soddaroq
Masshtablanish	Yaxshi (p oshsa)	Cheklangan	Parallel usul moslashuvchan

Tahlil natijalariga ko'ra quyidagi xulosalar chiqarish mumkin:

1. Vaqt murakkabligi jihatidan parallel usul aniq ustunlik ko'rsatadi. N ta nuqta va M vaqt qadami uchun CAU ning O(N·M) murakkabligiga nisbatan parallel usul O(N·M/p + M·t_comm) murakkabligiga ega. p protsessor soni ortishi bilan hisoblash vaqti deyarli p marta qisqaradi.

2. Xotira sarfi bo'yicha parallel usulda har bir protsessor faqat N/p ta nuqta uchun ma'lumot saqlaydi, bu esa katta o'lchamli masalalarni hal qilish imkonini beradi.

3. Barqarorlik sharti ikkalasida ham bir xil (CFL: $r \leq 1/2$). Bu shundan kelib chiqadiki, qadamlar tanlashda bir xil cheklov mavjud.

4. CAU amalga oshirish soddaligi va protsessorlar orasidagi aloqa xarajatisiz ishlashi bilan kichik o'lchamli masalalar uchun afzalroqdir. Agar N kichik bo'lsa va aloqa xarajati hisoblash vaqtidan katta bo'lsa, parallel usul zarar ko'radi (Amdahl qonuni).

Masshtablanish tahlili: $N >> p \cdot t_{comm}/t_{calc}$ shartida parallel usul samarali ishlaydi. Amalda bu shuni anglatadiki, N ta setka nuqtasi uchun optimal protsessorlar soni $p_{opt} \approx \sqrt{(N \cdot t_{calc}/t_{comm})}$ bo'ladi.

Xulosa qilib aytganda, yirik o'lchamli masalalar ($N > 10^4$) va ko'p protsessorli tizimlar uchun parallel usul aniq samaraliroqdir. Kichik masalalar uchun esa CAU ning soddaligi ustunlikdir.

Adabiyotlar:

[1] Samarskiy A.A. Farq sxemalari nazariyasi. — M.: Nauka, 1977. — 656 b.

[2] Leveque R.J. Finite Difference Methods for Ordinary and Partial Differential Equations. — SIAM, 2007. — 341 p.

[3] Pacheco P.S. An Introduction to Parallel Programming. — Morgan Kaufmann, 2011. — 392 p.

[4] Gropp W., Lusk E., Skjellum A. Using MPI: Portable Parallel Programming with the Message-Passing Interface. — MIT Press, 2014. — 400 p.

[5] Strikwerda J.C. Finite Difference Schemes and Partial Differential Equations. — SIAM, 2004. — 435 p.