Liet. matem. rink., T. 43, spec. nr., 586–591

Lygiagretusis jungtinių gradientų metodas

Raimondas ČIEGIS, Galina ŠILKO (VGTU)
el. paštas: rc@fm.vtu.lt, gs@sc.vu.lt

1. Uždavinio formulavimas

Egzistuoja daug uždavinių, reikalaujančių išretintų tiesinių lygčių sistemų sprendimo. Dažnai tokie uždaviniai yra labai dideli ir negali būti išspręsti vienu kompiuteriu. Taikant lygiagrečiuosius skaičiavimus išretintų tiesinių lygčių sistemų sprendimui tikslinga naudoti iteracinius metodus. Jais daugelis lygčių sistemų išsprendžiamos greičiau nei tiesioginiais metodais, taip pat vartojama mažiau kompiuterio atminties [2]. Iteracinių metodų išlygiagretinimo algoritmas dažnai yra paprastesnis nei tiesioginių metodų [5].

Šiame straipsnyje nagrinėjamas lygiagretusis modifikuotas iteracinis jungtinių gradientų (*LMJG*) metodas, ištirtas jo efektyvumas taikant skirtingus modifikatorius, atlikta algoritmo maštabavimo analizė. Skaičiavimo eksperimentų rezultatai yra gauti naudojant asmeninių kompiuterių klasterį „vilkas.vtu.lt“.

MJG metodu sprendžiame tiesinių lygčių sistemą $AY = F$, kai matrica A yra simetrinė ir išretinta.

Jungtinių gradientų metodas [2].
$\mathbf{Y}_0,\ n = 0$,
$\mathbf{R}^0 = \mathbf{AY}^0 - \mathbf{F},\ \mathbf{W}^0 = \mathbf{V}^{-1}\mathbf{R}^0,\ \mathbf{P}^0 = \mathbf{W}^0.$
while $((\mathbf{R}^n, \mathbf{R}^n) > \varepsilon)$
$\mathbf{G}^n = \mathbf{AP}^n,\ \tau_{n+1} = \dfrac{(\mathbf{W}^n, \mathbf{R}^n)}{(\mathbf{G}^n, \mathbf{P}^n)},$
$\mathbf{Y}^{n+1} = \mathbf{Y}^n - \tau_{n+1}\mathbf{P}^n,$
$\mathbf{R}^{n+1} = \mathbf{R}^n - \tau_{n+1}\mathbf{G}^n,$
$\mathbf{W}^{n+1} = \mathbf{V}^{-1}\mathbf{R}^{n+1}.$
$\beta_n = \dfrac{(\mathbf{W}^{n+1}, \mathbf{R}^{n+1})}{(\mathbf{W}^n, \mathbf{R}^n)},$
$\mathbf{P}^{n+1} = \mathbf{W}^{n+1} + \beta_n\mathbf{P}^n,\ n = n + 1,$
end while.

Čia V yra speciali matrica. Parinkdami šią matricą siekiame dviejų tikslų [2, 6]. *Pirma*, matricos V spektras turi būti artimas matricos A spektrui, t.y.,

$$\tilde{\lambda}_1 V \leqslant A \leqslant \tilde{\lambda}_M V, \quad \frac{\tilde{\lambda}_1}{\tilde{\lambda}_M} \gg \frac{\lambda_1}{\lambda_M},$$

čia λ_1 ir λ_M yra matricos A mažiausia ir didžiausia tikrinės reikšmės, $\tilde{\lambda}_1$ ir $\tilde{\lambda}_M$ yra matricos $V^{-1}A$ mažiausia ir didžiausia tikrinės reikšmės, M yra nežinomųjų skaičius. Iteracijų skaičius priklauso nuo matricos $V^{-1}A$ sąlygotumo skaičiaus. Parinkdami $V = A$ tikslų sprendinį gausime po vienos iteracijos. Tačiau tokios iteracijos realizacija yra ekvivalenti pradinio uždavinio $AY = F$ sprendimui. Todėl *antrasis* reikalavimas, keliamas matricai V, yra tas, kad tiesinių lygčių sistema $VZ = F$ būtų lengvai išsprendžiama. Matricą V vadinsime modifikatoriumi (angl. *preconditioner*).

Kai nagrinėjame lygiagretųjį MJG algoritmą, papildomai reikalausime, kad tiesinių lygčių sistemos $VY = F$ sprendimas būtų efektyviai lygiagretinamas [4, 6].

Teorinę analizę atliksime nagrinėdami trimačio Puasono uždavinio

$$\begin{cases} -\left(\dfrac{\partial^2 u}{x_1^2} + \dfrac{\partial^2 u}{x_2^2} + \dfrac{\partial^2 u}{x_3^2}\right) = f(x_1, x_2, x_3), \quad (x_1, x_2, x_3) \in Q, \\ u(x) = \mu(x), \quad x \in \partial Q, \end{cases} \tag{1}$$

sprendimą, čia $Q = \{x = (x_1, x_2, x_3)\colon 0 < x_1 < 1, 0 < x_2 < 1, 0 < x_3 < 1\}$. Diferencialinių uždavinių spręsime baigtinių skirtumų metodu. Srityje Q apibrėžiame diskretųjį tinklą:

$$\omega_h = \{(x_{1i}, x_{2j}, x_{3k})\colon\ x_{1i} = ih,\ \ x_{2j} = jh,\ \ x_{3k} = kh,\ \ i, j, k = 1, \ldots, N-1\},$$

čia h yra parinktas taip, kad $x_{1N} = 1$, $x_{2N} = 1$, $x_{3N} = 1$. Šio tinklo paviršiaus mazgų aibę žymime $\partial\omega_h$. Diskrečiojo tinklo mazguose apibrėžiame skirtumų funkciją $y_{i,j,k} = y(x_{1i}, x_{2j}, x_{3k})$.

Diferencialinę lygtį ir kraštinę sąlygą aproksimuojame baigtinių skirtumų schema:

$$\begin{cases} -\big(y_{\bar{x}_1 x_1} + y_{\bar{x}_2 x_2} + y_{\bar{x}_3 x_3}\big) = f_{ijk}, \quad (x_{1i}, x_{2j}, x_{3k}) \in \omega_h, \\ y(x) = \mu(x), \quad x \in \partial\omega_h. \end{cases} \tag{2}$$

Baigtinių skirtumų schema apibrėžia N^3 eilės tiesinių lygčių sistemą:

$$\mathbf{AY} = \mathbf{F}, \tag{3}$$

kurios matrica yra juostinė ir kiekvienoje eilutėje yra nedaugiau nei septyni nenuliniai elementai.

2. Modifikatoriai

Nagrinėsime du svarbus matricos A modifikatorius.

Diagonalinis modifikatorius

Išskaidykime matricą A į apatinės trikampės, įstrižainės ir viršutinės trikampės matricų sumą $A = L + D + L^T$, L – trikampė apatinė matrica. Tada $V = D$.

Įvertinę pagrindines MJG algoritmo operacijas gauname, kad vienos iteracijos sudėtingumas yra $W = 14N^3\gamma$, čia γ yra bazinės operacijos $a = b + cd$ vykdymo laikas. Naudojant diagonalinį modifikatorių iteracijų skaičius didėja $\mathrm{O}(N)$ greičiu.

$MIC(0)$ modifikatorius

Modifikatorius yra toks [5, 6]:

$$V = (L + D_0)D_0^{-1}(L^T + D_0),$$

kur $D_0 = diag(d_1, d_2, \ldots, d_{N^3})$ yra įstrižainė matrica, o jos koeficientai apskaičiuojami iš matricos A koeficientų, taip kad būtų išpildytos lygybės:

$$\mathbf{AE} = \mathbf{VE},$$

čia $\mathbf{E}$ yra matrica, kurios visi elementai yra lygūs vienetui.

Įvertinę pagrindines MJG algoritmo operacijas, gauname vienos iteracijos sudėtingumo įvertį $W = 22N^3\gamma$. Naudojant $MIC(0)$ modifikatorių daugeliui matricų iteracijų skaičius didėja tik $\mathrm{O}(\sqrt{N})$ greičiu [1].

3. Lygiagrečiojo algoritmo sudėtingumo analizė

Tarkime turime p procesorių. Nagrinėsime MJG lygiagretųjį algoritmą. Naudosime duomenų lygiagretumo metodiką, kai užduočių paskirstymas tarp procesorių gaunamas automatiškai, nurodžius kaip skirstomos matricos $\mathbf{A}$ eilutės. Galime naudoti skirtingas duomenų paskirstymo tarp p procesorių schemas.

$1D$ duomenų paskirstymas atliekamas vienos diskrečiojo tinklo koordinatės atžvilgiu, kiekvienam procesoriui tenka $N \times N \times \frac{N}{p}$ duomenų. Nagrinėdami MJG algoritmą matome, kad atlikdami dvi operacijas procesoriai privalo pasikeisti duomenimis. Skaičiuodami sandaugą $\mathbf{A}X$ jie komunikuoja su dviem kaimyniniais procesoriais ir nusiunčia kiekvienam iš jų po N^2 duomenų. Tokios operacijos kaštai yra įvertinami formule [3]

$$T_{mvs} = 2(\alpha + \beta N^2),$$

čia α – starto laikas, β – vieno skaičiaus siuntimo laikas. Antroji operacija yra skaliarinės dviejų vektorių sandaugos skaičiavimas. Jos vykdymo metu atliekama grupinė duomenų persiuntimo operacija, kurios kaštai blogiausių atveju gali būti:

$$T_{ss}(p) = c_1 p.$$

Panašiai yra nagrinėjami $2D$ ir $3D$ duomenų paskirstymai. Jie svarbūs tuo, kad didėjant procesorių skaičiui mažėja tarp dviejų procesorių persiunčiamų duomenų kiekis.

Diagonalinio modifikatoriaus atveju uždavinio $\mathbf{V}Y = F$ sprendimas atliekamas lygiagrečiai ir procesoriams nereikia komunikuoti tarpusavyje. Todėl lygiagrečiojo algoritmo vykdymo laiką įvertiname formule:

$$T_p = \frac{14N^3}{p}\gamma + 2(\alpha + N^2\beta) + 3c_1 p.$$

$MIC(0)$ modifikatoriaus atveju tiek matricos $\mathbf{D}_0$ koeficientų skaičiavimas, tiek ir modikatoriaus realizavimas kiekvienoje iteracijoje yra nuoseklūs ir šį algoritmą labai sunku taip pertvarkyti, kad galėtume efektyviai naudoti daug procesorių [6]. Paprasčiausia išeitis yra naudoti tik lokalią kiekvieno procesoriaus informaciją, bet tokio modifikatoriaus iteracijų skaičius yra tos pačios eilės, kaip ir naudojant diagonalinį modifikatorių. Įvairių algoritmo modifikacijų palyginimas bus pateiktas atskirame darbe.

4. Maštabavimo analizė

Iš Amdahlo dėsnių žinoma, kad [2]:

a) lygiagrečiųjų algoritmų efektyvumas mažėja, kai didiname procesorių skaičių sprendžiant fiksuoto dydžio uždavinį,

b) algoritmo efektyvumas didėja, kai, fiksavę procesorių skaičių, sprendžiame vis sunkesnius uždavinius.

Atliksime MJG algoritmo maštabavimo analizę, t.y., nustatysime kokiu greičiu turi didėti uždavinio dydis, kad taip pat efektyviai panauduotume vis daugiau procesorius [3]. Lygiagrečiojo algoritmo papildomus kaštus pažymėsime $T_0(W,p) = pT_p - W$. Rasime izoefektyvumo funkciją , t.y., uždavinio dydžio W priklausomybę nuo procesorių skaičiaus, garantuojančią norimą lygiagrečiojo algoritmo efektyvumą. Sprendžiame lygtį:

$$W = eT_0(W,p), \quad e = \frac{E_p}{1-E_p},$$

čia $E_p = \frac{W}{W+T_0(W,p)}$ yra algoritmo efektyvumas.

Lygiagrečiojo MJG algoritmo papildomieji kaštai yra:

$$T_0(W,p) = 2p(\alpha + N^2\beta) + 3c_1p^2 = 2\alpha p + 2N^2\beta p + 3c_1p^2.$$

Asimptotinę izoefektyvumo funkciją randame įvertinę atskirų algoritmo sudedamųjų dalių poveikį:

- pirmojo papildomų kaštų nario: $13N^3 \propto 2e\alpha p \Rightarrow N \propto \sqrt[3]{\frac{2e\alpha}{13}p}$,

- antrojo papildomų kaštų nario: $13N^3 \propto 2eN^2\beta p \Rightarrow N \propto \frac{2e\beta}{13}p$,
- trečiojo papildomų kaštų nario: $13N^3 \propto ec_1p^2 \Rightarrow N \propto \sqrt[3]{\frac{3ec_1p^2}{13}}$.

Didžiausias yra antrojo papildomųjų kaštų nario įnašas, t.y., priklausomybė nuo β. Norėdami išlaikyti fiksuotą efektyvumą E_p, turime didinti N proporcingai procesorių skaičiui p.

5. Skaičiavimo eksperimentai

Skaičiavimo eksperimentų rezultatai gauti naudojant asmeninių kompiuterių klasterį „vilkas.vtu.lt“ (informacija apie klasterį `http://vilkas.vtu.lt/`). Matrica $\mathbf{A}$ buvo paskirstyta tarp p procesorių panaudojant $1D$ metodą. 1 lentelėje pateikti skaičiavimo laikai T_p, kai naudojome diagonalinį modifikatorių.

Visais atvejais didėjant procesorių skaičiui uždavinio sprendimo laikas trumpėjo.

Analizuojant lygiagrečiuosius algoritmus, labai svarbūs yra algoritmo spartinimo $S_p = \frac{W}{T_p}$ ir algoritmo efektyvumo $E_p = \frac{S_p}{p}$ koeficientai. Šių koeficientų reikšmės yra pateiktos 2 lentelėje. Skaičiavimo rezultatai patvirtina gautus teorinius maštabavimo įverčius, kad norint palaikyti fiksuotą efektyvumą E_p, N turime didinti proporcingai procesorių skaičiui p.

1 lentelė. MJG algoritmo su diagonaliniu modifikatoriumi skaičiavimo laikas T_p

p	$N = 40$	$N = 80$	$N = 160$
1	2,2152	34,5746	557,6322
2	1,4083	19,3189	281,3233
4	0,8257	10,7250	177,6106
8	0,5322	6,1007	95,4990
10	0,4856	5,1902	78,4305

2 lentelė. Algoritmo spartinimo $S_p = \frac{W}{T_p}$ ir efektyvumo $E_p = \frac{S_p}{p}$ koeficientai

p	$N = 40$		$N = 80$		$N = 160$	
	S_p	E_p	S_p	E_p	S_p	E_p
2	1,57	0,79	1,79	0,89	1,98	0,99
4	2,68	0,67	3,22	0,81	3,14	0,78
8	4,16	0,52	5,67	0,71	5,84	0,73
10	4,56	0,46	6,66	0,67	7,11	0,71

Literatūra

[1] T.C. Chan, H.A. van der Vorst, Approximate and incomplete factorizations, in: D.E. Keyes, A. Samed, V. Venkatakrishnan (Eds.), *Parallel Numerical Algorithms*, *ICASE/LaRC Interdisciplinary Series in Science and Engineering*, Vol. 4, Kluwer Academic, Dordecht (1997), pp. 167–202.
[2] R. Čiegis, *Diferencialinių lygčių skaitiniai sprendimo metodai*, Technika, Vilnius (2003).
[3] R. Čiegis, *Lygiagretieji algoritmai*, Technika, Vilnius (2003).
[4] I.S. Duff, H.A. van der Vorst, Developments and trends in the parallel solution of linear systems, *Parallel Comput.*, **25**, 1931–1970 (1999).
[5] A. Gupta, V. Kumar, A. Sameh, Performance and scalability of preconditioned Conjugate Gradient methods on parallel computers, *IEEE Transactions on Parallel and Distributed Systems*, **6**, 455–469 (1995).
[6] M. Magolu monga Made, H.A. van der Vorst, A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings, *Future Generation Comp. Systems*, **17**, 925–932 (2001).

Parallel conjugate gradient method

R. Čiegis, G. Šilko

We investigate a parallel version of the preconditioned conjugate gradient method. A scalability analysis is done for a finite difference scheme which approximates the 3D elliptic problem. Results of numerical experiments are presented which confirm theoretical conclusions.