Vol.2012-HPC-133 No.30
2012/3/27

k 段飛ばし共役勾配法: 通信を回避することで大規模並列計算で有効な対称正定値疎行列連立 1 次方程式の反復解法

本谷 徹[†1] 須田礼仁[†1]

連立一次方程式の反復解法として広く使われている共役勾配法を大規模に並列化した際に律速となるのは，頻繁に現われる内積計算の通信遅延である．内積計算は計算機全体の集団通信を必要とすることから，大規模なアーキテクチャでの通信遅延は大きくなる．また強スケーリングにおいては計算粒度が小さくなり，通信遅延は相対的が大きくなってしまう困難を抱えている．物理的制約を超えての通信遅延削減は不可能なため，アルゴリズム側のアプローチによる通信遅延の削減が必要とされている．

本稿では，共役勾配法の k+1 反復分の内積計算に必要な通信を 1 回で済ませることで集団通信を回避し，通信遅延を削減する k 段飛ばし共役勾配法を提案，実装した．

k-skip Conjugate Gradient Methods: Communication Avoiding Iterative Symmetric Positive Definite Sparse Linear Solver For Large Scale Parallel Computings

Toru MOTOYA[†1] and Reiji SUDA[†1]

The rate limiting factor of the conjugate gradient method parallelized for supercomputers, which is widely used as a sparse linear solver, is the communication latencies of inner products that frequently occur. Because the inner product needs the collective communication of whole processors, the larger the architectures become, the bigger the communication latencies of inner products are. In addition, the communication latencies of the strong scaling will be relatively big because the granularity will be small. Therefore, reducing the communication latencies from the viewpoint of improving algorithms is necessary because they cannot be reduced beyond the physical limitations,

In this paper,we propose and implement a new algorithm k-skip conjugate gradient method that avoids communication by skipping the collective communication of inner products of k times out of k+1 iterations.

1. はじめに

大規模数値計算においては今後ますます多くのプロセッサが使用されるようになり，それに伴い困難が多く生じると予想されている．その中でも重要なのが通信遅延の肥大化である．リダクション，ブロードキャスト，同期など，全体のプロセッサ間での集団通信はエクサスケールにもなると 30 段もの通信回数が必要となり，そこから生じる通信遅延は非常に大きくなる．それに加えて強スケーリングでの計算粒度は非常に細かくなり，通信遅延が相対的に大きくなる困難が発生すると考えられており課題である．

強スケーリングとは問題サイズを一定にしてプロセッサ数を増やしていくことで高速化を図る並列化手法であり，現在はプロセッサ数に対する問題サイズを一定に保って並列化を進める弱スケーリングが主流である．弱スケーリングにおいては並列化可能部分がほぼ一定の割合で存在するため，アムダールの法則から並列化が比較的容易であった．しかしプロセッサ数が非常に多くなる将来，弱スケーリングに従って並列化を進めると問題が不必要に大きくなるほか，解を得るまでの時間が問題サイズに依存する場合に非現実的な時間を要することとなり，そのような点において不利となる．そのため，問題の大きさが弱スケーリングでの限界に達した後には強スケーリングで高速化を進めていく方針が有力である．強スケーリングにおける困難とは，プロセッサ数を増やすにつれ並列化可能部分が減っていき，計算に対する通信の割合が大きくなっていくことである．

通信遅延は物理的制約により一定以下にできないことから，アルゴリズムの観点からの通信遅延の削減が必要であり，計算量の割に通信量が多くなるような問題では極めて重要である．そのような理由から，通信回避アルゴリズムがこれまでにいくつか提案されている．[1)2)] 通信回避アルゴリズムとは，計算量の増大やそれに伴う誤差の増大を許容してでも通信回数を削減し，結果として大規模計算における高速化を図ろうとするものである．

対称正定値疎行列を係数行列に持つ連立一次方程式の反復解法として広く利用されている共役勾配法 (以下 CG 法)[3)] も，並列化されれば通信が必要になる点において例外ではなく，図 1 にも示されるように 1 つの反復につき 2 回の内積と 1 回の行列ベクトル積が現れ，そこで通信が発生する．特に内積においては計算機全体の集団通信を必要とすることから，

†1 東京大学情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo

アーキテクチャの増大に伴いその通信時間も単調に増加していく．よって内積の通信遅延を削減することの意義は大きい．

以上のような背景から，本稿では k+1 回分の反復で必要な内積計算に必要な通信を 1 回に済ませることで回避できる k 段飛ばし共役勾配法 (以下 k-skip CG 法) を提案，実装した．k-skip CG 法は CG 法からの数学的等価な変換により導出される派生の 1 つであり，通信回数を削減する代償として計算量と誤差が増大する通信回避アルゴリズムである．これまでにも CG 法の通信回避アルゴリズムがいくつか提案されてきたが，その中でも k-skip CG 法は計算量の増加が少ない点で有利である．また，アルゴリズムの変換により精度が若干改善されることが実験で明らかになっており，k-skip CG 法の派生を複数示す．

以下，2 章では CG 法における内積の通信回避アルゴリズムの先行研究について，3 章では k-skip CG 法について述べる．4 章では先行研究を含めた実験とその評価を行い，5 章ではまとめと今後の課題を述べる．

2. 関 連 研 究

内積の通信回数を削減する目的で，これまでにいくつかの関連手法が提案されてきた．CG 法のアルゴリズムに現れる内積計算に必ずしも通信は必要ないことを 1 番最初に示したのは Rosendale[4] であった．CG 法の反復に現れる等式 $r_{i+1} = r_i - \alpha_i A p_i$ の内積を取ることで，異なる反復のベクトルの内積の間に等式 $(r_{i+1}, r_{i+1}) = (r_i, r_i) - 2\alpha_i (r_i, Ap_i) + \alpha^2 (Ap_i, Ap_i)$ が成り立つことわかる．また (p_i, Ap_i) にも同様の等式が成り立つと Rosendale の報告の中では述べられている．これは 1 度計算された内積がその後の内積計算に再利用できることを示しており，すなわち集団通信を使ってのベクトルの要素自乗和から内積を計算する必要が無い．具体的なアルゴリズム，実験結果が存在しないものの，Rosendale の報告はその後の研究に大きな影響を与えた．

Saad[5] は $\alpha_i = (r_i, r_i)/(p_i, Ap_i) = (r_i, r_i)/(r_i, Ap_i)$ であることを用いて，

$$(r_{i+1}, r_{i+1}) = \alpha_i^2 (Ap_i, Ap_i) - (r_i, r_i) \quad (1)$$

を示した．ここから導出されるアルゴリズムは (p_i, Ap_i) と (Ap_i, Ap_i) の 2 つの内積が集団通信で計算され，かつその集団通信は重ね合わせることが可能である．(r_i, r_i) は 最初の $i = 0$ のみベクトルから計算され，それ以降は 式 (1) から順次求められていくこととなる．このアルゴリズムは CG 法と比較すると内積による通信遅延が半分になる反面，不安定になることがわかっている[6][7]．Meurant[7] は (p_i, Ap_i) と (Ap_i, Ap_i) に加えて (r_i, r_i) を 1 度の通信で計算し，式 (1) に適用することで精度が改善されることを示した．Saad と

入力: A: 正定値対称行列
入力: b: 0 でない右辺ベクトル
入力: x_0: 初期解
出力: x: 連立 1 次方程式 $Ax = b$ の数値解
$i \leftarrow 0$
$r_0 \leftarrow b - Ax_0$
$p_0 \leftarrow r_0$
repeat
  $\gamma_i \leftarrow (r_i, r_i)$
  if $i \neq 0$ then
    $\beta_i \leftarrow \gamma_i / \gamma_{i-1}$
    $p_i \leftarrow r_i + \beta_i p_{i-1}$
  end if
  $v_i \leftarrow Ap_i$
  $\sigma_i \leftarrow (p_i, v_i)$
  $\alpha_i \leftarrow \gamma_i / \sigma_i$
  $x_{i+1} \leftarrow x_i + \alpha_i p_i$
  $r_{i+1} \leftarrow r_i - \alpha_i v_i$
  $i \leftarrow i + 1$
until sqrt$(\gamma_i/(b,b))$ が十分小さい
return x_i

図 1 共役勾配法のアルゴリズム

Meurant は (r_i, r_i) を通信を使わずに計算することで内積の通信回数を削減したが，その一方で Eijkhout ら[8] は (p_i, Ap_i) の通信を回避するアルゴリズムを同様にして 3 つ提案し，その安定性を示した．

Chronopoulos と Gear は複数反復の内積を 1 回の集団通信から計算する s-step 法[9] を提案した．s-step 法は s 回分の反復のクリロフ部分空間の基底をまず最初に生成しておき，それらの内積計算を 1 度に行う．その計算結果から CG 法の反復 s 回分と等しいデータを得るために $O(s^4)$ の計算量を必要とする．s-step 法は動作するためには $s = 5$ 以下である必要があるとされ，不安定であることがわかっている．Toledo はその博士論文で Krylov Basis CG 法[10](以下 KBCG 法) を提案した．KBCG 法も s-step 法と同様にクリロフ部分空間の基底を最初に生成し，そのグラム行列を計算することで内積を 1 度の通信で計算する．KBCG 法においては τ 回の反復の内積を 1 度に計算するとすれば $O(\tau^3)$ の計算量が必要となる．Toledo の実験によると KBCG 法が安定動作する τ の値は 9 である．Hoemmen が提案した communication avoiding CG 法[11](以下 CACG 法) も同じく最初にクリロフ部分空間を生成，グラム行列を計算する．CACG 法は内積計算だけでなく行列ベクトル積の通信を回避することに主眼が置かれており，その複数反復に必要な通信を 1 度に行う．その代償としてアルゴリズムは非常に複雑となり，計算量が増大している．提案者による数値実験結果は存在しない．

3. k 段飛ばし共役勾配法 (k-skip CG 法)

k-skip CG 法は s-step 法や KBCG 法と同じく複数回の反復につき 1 回の集団通信で必要な内積を計算できるアルゴリズムである．ここでの k とは通信を回避できる反復の数とし，よって $k+1$ 回の反復につき 1 度の集団通信を必要とする．

3.1 アルゴリズムの導出

CG 法のアルゴリズムでは 1 つの反復に (r_i, r_i) と (p_i, Ap_i) の 2 つの内積が現れ，集団通信がそれぞれに必要となる．反復に跨ってそれらの通信を回避しようとすると，それぞれに式 (1) と同様の内積に関する漸化式が必要であることがわかる．(r_{i+1}, r_{i+1}) は式 (1) をそのまま用いて計算しその集団通信を回避する．(p_i, Ap_i) に加えて (p_i, A^2p_i) の集団通信を回避することが必要となるので，p_{i+1} どうしの A^j 内積を取る．

$$\begin{aligned}(p_{i+1}, A^jp_{i+1}) &= (r_{i+1}, A^jp_{i+1}) + \beta_i(p_i, A^jp_{i+1}) \\ &= (r_{i+1}, A^jp_{i+1}) + \beta_i(p_i, A^jr_{i+1}) + \beta_i^2(p_i, A^jp_i) \\ &= (r_{i+1}, A^jp_{i+1}) + \beta_i(p_i, A^jr_i) + \beta_i^2(p_i, A^jp_i) - \alpha_i\beta_i(p_i, A^{j+1}p_i) \quad (2)\end{aligned}$$

ここで (r_i, A^jp_i) の通信を回避する等式が必要となったため，同様にして等式を導く．

$$\begin{aligned}(r_{i+1}, A^jp_{i+1}) &= (r_{i+1}, A^jr_{i+1}) + \beta_i(r_{i+1}, A^jp_i) \\ &= (r_{i+1}, A^jr_{i+1}) + \beta_i(r_i, A^jp_i) - \alpha\beta_i(p_i, A^{j+1}p_i) \quad (3)\end{aligned}$$

更に同様にして r_i の A^j 内積を導く．

$$(r_{i+1}, A^jr_{i+1}) = (r_i, A^jr_i) - 2\alpha_i(r_i, A^{j+1}p_i) + \alpha_i^2(p_i, A^{j+2}p_i) \quad (4)$$

以上の連立漸化式 (2)，(3)，(4) により，α_i，β_i，(p_i, A^jp_i)，$(p_i, A^{j+1}p_i)$，$(p_i, A^{j+2}p_i)$，(r_i, A^jp_i)，$(r_i, A^{j+1}p_i)$，(r_i, A^jr_i) から (p_{i+1}, A^jp_{i+1})，(r_{i+1}, A^jp_{i+1})，(r_{i+1}, A^jr_{i+1}) が計算されることがわかる．すなわち集団通信をせずに次の反復における内積が求まることとなる．(p_{i+1}, Ap_{i+1})，(p_{i+1}, A^2p_{i+1}) から (r_{i+1}, r_{i+1})，α_{i+1}，β_{i+1} が計算され，そこからベクトル和を計算すれば CG 法における反復が進むこととなる．この連立漸化式を再帰的に計算し i の値を進めていくことにより，複数の反復の内積の値を集団通信をせずに求めていくことが可能である．

この連立漸化式における内積はスカラー計算から求まる値に他ならないため，次の様に記号を導入することとする．$\gamma_i = (r_i, r_i)$，$\delta_{i,j} = (r_i, A^jr_i)$，$\eta_{i,j} = (r_i, A^jp_i)$，$\zeta_{i,j} = (p_i, A^jp_i)$．まとめると連立漸化式は以下のようになる．

$$\begin{aligned}\alpha_i &= \gamma_i/\zeta_{i,1} \\ \beta_i &= \alpha_i\zeta_{i,2}/\zeta_{i,1} - 1 \\ \gamma_{i+1} &= \beta_i\gamma_i \\ \delta_{i+1,j} &= \delta_{i,j} - 2\alpha_i\eta_{i,j+1} + \alpha_i^2\zeta_{i,j+2} \\ \eta_{i+1,j} &= \delta_{i+1,j} + \beta_i\eta_{i,j+1} - \alpha_i\beta_i\zeta_{i,j+1} \\ \zeta_{i+1,j} &= \eta_{i+1,j} + \beta_i\eta_{i,j} + \beta_i^2\zeta_{i,j} - \alpha_i\beta_i\zeta_{i,j+1}\end{aligned}$$

この連立漸化式で反復を 1 回分進めるためには j の値を 2 ずつ増やしていく必要がある．欲しい数値が $\zeta_{n+k,1}$，$\zeta_{n+k,2}$ だとすれば，$i = n$ においては $j = 2k+2$ が必要となる．つまり，$i = n$ から漸化式を適用していくとすれば，p_i，r_i それぞれのクリロフ部分空間 $p_i, Ap_i, \cdots A^{k+1}p_i$, $r_i, Ar_i, \cdots A^kr_i$ とそれらの内積が漸化式の初期値として必要であることを示している．

連立漸化式を解く際には動的計画法[12] を用いて進めていく必要がある．動的計画法とは計算結果を表に記録しておき再利用することで計算量を抑える手法である．例えばフィボナッチ数列 $a_{n+2} = a_{n+1} + a_n$ をただの再帰関数として計算すると計算量が指数爆発してしまう．しかし動的計画法を用いて途中結果を再利用すると $O(n)$ になる，といった具合で高速化が可能である．k-skip CG 法における連立漸化式の表と計算結果の再利用の様子

表 1 $k+1$ 反復における k-skip CG 法と関連手法の比較

	行列ベクトル積	内積	内積の通信回数	スカラー計算量	ベクトル和
CG 法	$k+1$	$2k+2$	$2k+2$	$O(k)$	$3k+3$
s-step 法	$k+1$	$2k+2$	1	$O(k^4)$	k^2+4k+3
KBCG 法	$2k+2$	$4k^2+16k+16$	1	$O(k^3)$	$6k+12$
k-skip CG 法	$3k+2$	$3k+3$	1	$O(k^2)$	$3k+3$

を図 2 に端的に示す．ここでの横軸は i の値，すなわちその反復回数での内積の値であり，縦軸は j の値，すなわち内積においてベクトルに挟まれている A の指数である．計算手順は，まず集団通信で内積を計算し 1 番左の列に計算結果を格納しておく．その後は反復を 1 回進める進める毎に，左の列にあるデータを参照して連立漸化式で計算した結果を右の列に埋め，得られた $\zeta_{i,1}$ と $\zeta_{i,2}$ を用いて CG 法における反復を進めることを繰り返す．これが k-skip CG 法のアルゴリズムの骨子である．$k+1$ 回の反復での内積計算を 1 回の通信で済ませた場合のスカラー計算量は，動的計画法を用いたことにより $O(k^2)$ である．これは関連手法の中で最も少ない計算量である．表 1 は k-skip CG 法と関連手法を比較したもので，全て $k+1$ 反復に 1 回集団通信した場合とする．

最終的に k-skip CG 法のアルゴリズムは図 3 となる．尚，$k=0$ とした場合の k-skip CG 法は Meurant が提案した Saad のアルゴリズムの修正版と一致する．

3.2 k skip CG 法の派生

後の実験結果で示されるように，前節で導いた k-skip CG 法は計算量は少ないが精度は関連手法の KBCG 法に劣る．この欠点を補うべく，k-skip CG 法に改造を施した派生を 2 つ示す．派生元の k-skip CG 法 を k-skip CG 法 $^{(1)}$ とする．

- γ_{i+1} の計算手順の変更

γ_{i+1} の計算 (式 (1)) には引き算が現れ，桁落ちが精度の悪化を招いていることが予想できる．Saad の不安定な手法も γ_{i+1} が原因であったことは既に述べた．よって γ_{i+1} を求める演算の順序を変更することにより桁落ちの回避を試みる．

$$\begin{aligned} tmp_0 &= \gamma_i - \alpha_i \eta_{i,1} \\ tmp_1 &= \eta_{i,1} - \alpha_i \zeta_{i,2} \\ \gamma_{i+1} &= tmp_0 - \alpha_i tmp_1 \end{aligned} \tag{5}$$

k-skip CG 法 $^{(1)}$ にこの僅かな変更を施した手法を k-skip CG 法 $^{(2)}$ とする．

- 2 項間漸化式による k-skip CG 法

数学的には等しくても計算手順が異なれば丸め誤差も異なるため，安定性に影響が現れる．

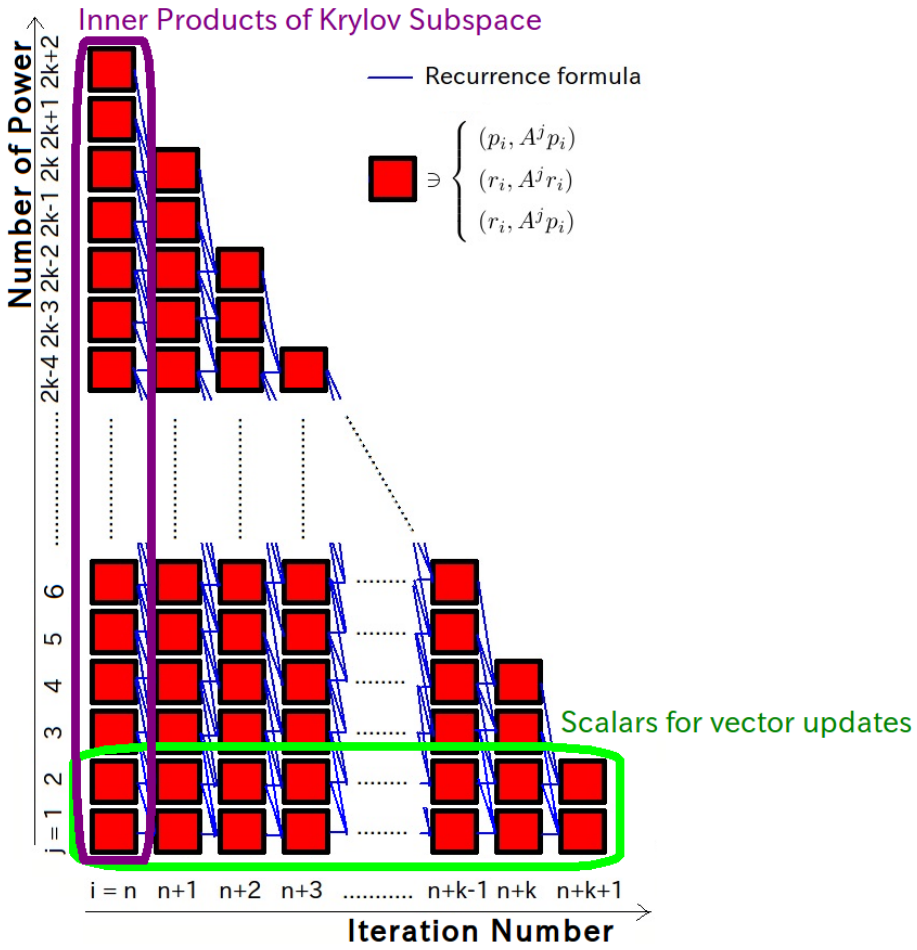

図 2 k-skip CG 法における連立漸化式の動的計画法

入力: A: 正定値対称行列, b: 0 でない右辺ベクトル, x_0: 初期解
入力: k: 内積通信を回避する反復の数
出力: x: 連立 1 次方程式 $Ax = b$ の数値解
$n \leftarrow 0$
$r_0 \leftarrow b - Ax_0$
$p_0 \leftarrow r_0$
repeat
  calculate $r_n, Ar_n, A^2r_n, \cdots A^kr_n$
  calculate $p_n, Ap_n, A^2p_n, \cdots A^kp_n, A^{k+1}p_n$
  /********** 1 度の通信で以下の内積を計算 **********/
  $\gamma_n \leftarrow (r_n, r_n)$
  $\delta_{n,1} \leftarrow (r_n, Ar_n), \cdots, \delta_{n,k} \leftarrow (r_n, A^kr_n)$
  $\eta_{n,1} \leftarrow (r_n, Ap_n), \cdots, \eta_{n,k} \leftarrow (r_n, A^kp_n), \eta_{n,k+1} \leftarrow (r_n, A^{k+1}p_n)$
  $\zeta_{n,1} \leftarrow (p_n, Ap_n), \cdots, \zeta_{n,k} \leftarrow (p_n, A^kp_n), \zeta_{n,k+1} \leftarrow (p_n, A^{k+1}p_n), \zeta_{n,k+2} \leftarrow (p_n, A^{k+2}p_n)$
  /**/
  for $i = n$ to $n + k$ do
    $\alpha_i \leftarrow \gamma_i/\zeta_{i,1}$
    $\beta_i \leftarrow \alpha_i\zeta_{i,2}/\zeta_{i,1} - 1$
    $\gamma_{i+1} \leftarrow \beta_i\gamma_i$
    for $j = 1$ to $2k - 2(i - n)$ do (動的計画法を用いて)
      $\delta_{i+1,j} \leftarrow \delta_{i,j} - 2\alpha_i\eta_{i,j+1} + \alpha_i^2\zeta_{i,j+2}$
      $\eta_{i+1,j} \leftarrow \delta_{i+1,j} + \beta_i\eta_{i,j+1} - \alpha_i\beta_i\zeta_{i,j+1}$
      $\zeta_{i+1,j} \leftarrow \eta_{i+1,j} + \beta_i\eta_{i,j} + \beta_i^2\zeta_{i,j} - \alpha_i\beta_i\zeta_{i,j+1}$
    end for
    $x_{i+1} \leftarrow x_i + \alpha_i p_i$
    $r_{i+1} \leftarrow r_i - \alpha_i Ap_i$
    $p_{i+1} \leftarrow r_{i+1} + \beta_i p_i$
  end for
  $n \leftarrow n + k + 1$
until sqrt($\gamma_n/(b, b)$) が十分小さい
return x_n

図 3 k 段飛ばし共役勾配法のアルゴリズム

式 (1) と 式 (5) の差は，式が不可分か可分かの違いである．ここで精度の向上を期待して，漸化式 (2)，(3)，(4) を不可分な漸化式に置き換えることを考える．すなわち，(r_i, A^jr_i)，(r_i, A^jp_i)，(p_i, A^jp_i) それぞれに CG 法でのベクトル更新の等式を 1 回だけ適用する．

$$(r_{i+1}, A^jr_{i+1}) = (r_i, A^jr_{i+1}) - \alpha_i(p_i, A^{j+1}r_{i+1})$$
$$(r_{i+1}, A^jp_{i+1}) = (r_{i+1}, A^jr_{i+1}) + \beta_i(p_i, A^jr_{i+1})$$
$$(p_{i+1}, A^jp_{i+1}) = (r_{i+1}, A^jp_{i+1}) + \beta_i(p_i, A^jp_{i+1})$$

ここでは新たに (r_i, A^jr_{i+1}), (p_i, A^jr_{i+1}), (p_i, A^jp_{i+1}) の漸化式を必要としている．

$$(r_i, A^jr_{i+1}) = (r_i, A^jr_i) - \alpha_i(p_i, A^{j+1}r_i)$$
$$(p_i, A^jr_{i+1}) = (r_i, A^jr_i) - \alpha_i(p_i, A^{j+1}p_i)$$
$$(p_i, A^jp_{i+1}) = (p_i, A^jr_{i+1}) + \beta_i(p_i, A^jr_i)$$

以上に示した 6 つの 2 項間漸化式を，k-skip CG 法 [(2)] の 3 つの漸化式に置き換えたものを k-skip CG 法 [(3)] とする．

4. 実験と評価

内積通信を回避する反復の数 k に着目して k-skip CG 法 [(1),(2),(3)] と KBCG 法の 4 つの手法で実験を行う．比較対象である KBCG 法は複数回の内積通信回避可能な関連手法で最も安定かつ高速であるとされている．本稿の実験における k の値は $0 \leq k < 30$ とする．

4.1 行列サンプルと実験環境

対角成分の異なる 6 つの 3 重対角行列で実験を行う．非対角成分は全て -1 で，行列サイズは全て 100 である．サンプル行列の特徴を表 2 に示す．一般に条件数が CG 法の収束速度に関係する．

連立 1 次方程式の初期解 x_0 は全て 0 ベクトル，右辺ベクトル b は全て 1，反復の終了条件は相対誤差 $1.0e-13$ 以下である．いずれも反復回数の上限を 1000 回とし，それ以上

表 2 サンプルの三重対角行列

	対角成分	条件数	最大固有値	最小固有値
T_0	25.0	1.17e+00	2.70e+01	2.30e+01
T_1	2.5	8.97e+00	4.50e+00	5.00e-01
T_2	2.05	7.94e+01	4.05e+00	5.10e-02
T_3	2.005	6.70e+02	4.00e+00	5.97e-03
T_4	2.0005	2.72e+03	4.00e+00	1.47e-03
T_5	2.0	4.13e+03	4.00e+00	9.67e-04

の反復を要する場合は発散とみなす．行列と CG 法の特性から，丸め誤差の影響が無ければ 100 回以内で収束するはずである．実験環境は cpu は Core2 U9400，精度は倍精度，コンパイラは GCC 4.4.3 を用いた．

4.2 実験結果

表 3，4，5 が実験結果を示す表である．k-skip CG 法 (1) を (1) と略記してある．結果は 3 通りに分かれる．以下はその詳細である．

- 数字:　収束した場合の反復数.
- 発散:　十分な回数の反復を繰り返したが，収束せず.
- 破綻:　計算途中で Not a Number が発生し，失敗した.

また，図 4，5 は T_3，T_4 における k-skip CG 法 (2) の収束履歴を示している．k は 20 までとしてある．横軸は反復回数，縦軸は対数スケールの相対残差 $sqrt(\gamma_i/(b,b))$ で，1000 反復まで追っている．破綻した場合にも収束履歴を得ることができた.

4.3 評　価

全体の傾向として，安定する k の値が連続でない結果となった．k の値が小さいうちに

表 3　左: T_0: 条件数 $1.17e+00$,　右: T_1: 条件数 $8.97e+00$

k	KBCG	(1)	(2)	(3)	k	KBCG	(1)	(2)	(3)
0	9	9	10	9	0	42	42	43	42
1	10	10	12	10	1	42	42	44	42
2	9	9	12	9	2	42	42	45	42
3	12	破綻	16	12	3	44	44	48	44
4	10	破綻	15	20	4	45	45	50	45
5	発散	破綻	破綻	破綻	5	42	42	48	42
6	発散	破綻	35	破綻	6	42	42	49	42
7	発散	破綻	24	破綻	7	48	48	56	48
8	破綻	破綻	27	破綻	8	45	破綻	54	45
9	発散	80	40	破綻	9	50	破綻	60	50
10	22	破綻	33	22	10	99	発散	110	44
11	192	破綻	48	発散	11	312	発散	192	発散
12	発散	発散	39	発散	12	611	発散	発散	破綻
13	発散	破綻	42	発散	13	発散	破綻	224	破綻
14	発散	破綻	45	300	14	120	発散	発散	発散
15	176	破綻	48	破綻	15	発散	発散	96	発散
16	85	破綻	51	破綻	16	85	発散	272	発散
17	発散	破綻	72	破綻	17	発散	破綻	発散	発散
18	破綻	破綻	76	発散	18	114	破綻	発散	発散
19	60	破綻	80	発散	19	820	発散	380	破綻
20	63	破綻	63	破綻	20	441	破綻	発散	破綻
21	198	破綻	66	破綻	21	352	破綻	発散	発散
22	713	破綻	92	発散	22	322	発散	発散	発散
23	発散	破綻	72	破綻	23	288	発散	発散	発散
24	発散	破綻	75	発散	24	200	発散	500	発散
25	発散	破綻	234	破綻	25	発散	発散	858	234
26	発散	発散	108	発散	26	発散	発散	発散	135
27	発散	発散	112	破綻	27	発散	破綻	560	112
28	発散	破綻	116	58	28	発散	破綻	発散	破綻
29	発散	破綻	120	発散	29	発散	発散	270	発散

表 4　左: T_2: 条件数 $7.94e+01$,　右: T_3: 条件数 $6.79e+02$

k	KBCG	(1)	(2)	(3)	k	KBCG	(1)	(2)	(3)
0	50	50	51	50	0	51	50	51	50
1	52	50	52	50	1	52	50	52	50
2	51	51	54	51	2	発散	発散	657	51
3	56	52	56	52	3	72	発散	96	80
4	55	55	55	50	4	95	100	95	95
5	66	発散	66	発散	5	354	828	186	240
6	189	発散	98	112	6	発散	発散	発散	182
7	発散	発散	96	破綻	7	発散	248	240	248
8	発散	117	207	破綻	8	225	発散	261	破綻
9	120	発散	120	120	9	230	発散	270	210
10	814	発散	396	275	10	374	発散	781	550
11	228	発散	発散	168	11	発散	発散	発散	672
12	発散	破綻	発散	発散	12	発散	破綻	発散	発散
13	発散	発散	発散	破綻	13	発散	発散	発散	破綻
14	発散	発散	発散	525	14	990	発散	発散	発散
15	400	破綻	発散	発散	15	発散	発散	発散	破綻
16	発散	破綻	発散	発散	16	629	発散	発散	発散
17	発散	発散	発散	発散	17	発散	破綻	発散	発散
18	950	発散	発散	破綻	18	発散	破綻	発散	破綻
19	発散	破綻	発散	発散	19	発散	発散	発散	破綻
20	発散	破綻	発散	破綻	20	発散	破綻	発散	発散
21	発散	発散	発散	破綻	21	発散	発散	発散	破綻
22	発散	発散	発散	破綻	22	発散	破綻	発散	発散
23	発散	発散	発散	発散	23	発散	発散	発散	発散
24	375	破綻	発散	発散	24	発散	発散	発散	破綻
25	発散	発散	発散	破綻	25	発散	発散	発散	破綻
26	発散	発散	発散	発散	26	発散	破綻	発散	破綻
27	発散	破綻	発散	発散	27	発散	発散	発散	破綻
28	667	破綻	発散	発散	28	発散	破綻	発散	破綻
29	発散	発散	発散	発散	29	発散	破綻	発散	発散

は収束するケースが連なるが，ところどころに破綻，発散が挟まる．k が大きくなると収束しなくなる傾向が強いが，収束する場合もところどころ現れる．丸め誤差の影響を受けやすい性格が浮き彫りとなった．行列の条件数に着目すると，最も大きい T_0 よりもむしろ 2 番目に大きい T_1 が収束しやすい．条件数が 1 に近い行列が収束しやすい理由として，行列の冪乗を係数にもつベクトルの内積計算に誤差が出にくいとの推測が成り立つ．

行列別に見ると，T_0 では k-skip CG 法 $^{(2)}$ が 1 番収束しやすい傾向にある．T_1 においては，k-skip CG 法 $^{(2)}$ と KBCG 法が収束しやすい傾向にあった．T_2 と T_3 では KBCG 法が大きな k でも収束を見せる一方，k-skip CG 法 $^{(2)}$ と k-skip CG 法 $^{(3)}$ は収束する k の値にまとまりを見せている．T_4 においては k-skip CG 法 $^{(2)}$ と k-skip CG 法 $^{(3)}$ が収束する場合が比較的多い．逆に T_5 においては KBCG 法が収束する傾向が強い．

アルゴリズム別に見ると，KBCG 法と k-skip CG 法 $^{(2)}$ は収束しない場合でも破綻に至るケースが少なく，逆に他の 2 つは発散しやすい傾向にある．KBCG 法は大きな k の値でもまばらに収束することがある．その反面，k-skip CG 法 $^{(2)}$ はまとまった k での収束傾向が強いものの，収束するまでの H ン復回数が早い段階で大きくなりがちであり，収束するも不安定であると言えよう．どの行列においても k-skip CG 法 $^{(1)}$ に比べると k-skip CG 法 $^{(2)}$ と k-skip CG 法 $^{(3)}$ は収束しやすく，比較的安定している．

以上の結果からは一概にどのアルゴリズムが優れているとは言い難い．適切な行列の場合に適切な k を選び実行すれば有効である．

収束履歴からは k-skip CG 法 $^{(2)}$ の挙動の 1 部が明らかになった．不安定な収束には 2 通りが観察される．まず発散する場合は無限大方向に向かうのではなく停滞する傾向にある

表 5 左: T_4: 条件数 $2.72e+03$, 右: T_5: 条件数 $4.13e+03$

k	KBCG	(1)	(2)	(3)	k	KBCG	(1)	(2)	(3)
0	51	50	51	50	0	51	50	51	50
1	52	50	52	50	1	52	50	52	50
2	発散	発散	93	51	2	90	破綻	破綻	破綻
3	296	172	164	204	3	760	発散	破綻	破綻
4	140	155	145	140	4	135	190	55	50
5	発散	発散	264	発散	5	発散	発散	破綻	破綻
6	発散	発散	231	378	6	発散	発散	破綻	破綻
7	発散	発散	464	272	7	発散	発散	破綻	破綻
8	発散	発散	破綻	発散	8	発散	発散	破綻	破綻
9	290	発散	480	310	9	820	発散	60	50
10	発散	発散	発散	528	10	発散	発散	破綻	破綻
11	発散	発散	発散	発散	11	発散	発散	破綻	破綻
12	発散	発散	発散	破綻	12	発散	発散	破綻	破綻
13	発散	発散	発散	発散	13	発散	発散	発散	発散
14	発散	発散	発散	発散	14	発散	発散	発散	発散
15	発散	発散	発散	発散	15	発散	発散	発散	発散
16	発散	破綻	発散	破綻	16	発散	発散	発散	破綻
17	発散	発散	発散	破綻	17	発散	発散	発散	破綻
18	発散	発散	発散	破綻	18	発散	発散	発散	発散
19	発散	発散	発散	破綻	19	発散	破綻	発散	破綻
20	発散	破綻	発散	発散	20	発散	発散	発散	破綻
21	発散	発散	発散	発散	21	発散	破綻	発散	破綻
22	発散	発散	発散	発散	22	発散	破綻	発散	破綻
23	発散	発散	発散	発散	23	発散	発散	発散	破綻
24	発散	破綻	発散	発散	24	発散	発散	発散	発散
25	発散	発散	発散	破綻	25	発散	発散	発散	発散
26	発散	破綻	発散	破綻	26	発散	発散	発散	発散
27	発散	発散	発散	発散	27	発散	破綻	発散	破綻
28	発散	発散	発散	発散	28	発散	発散	発散	破綻
29	発散	発散	発散	破綻	29	発散	発散	発散	破綻

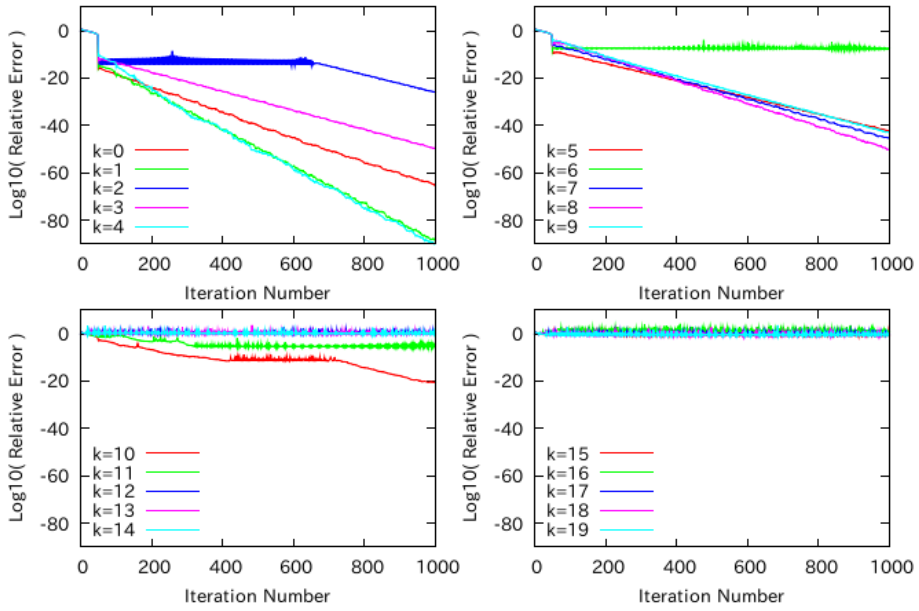

図 4 T_3 を係数行列とした場合の k-skip CG$^{(2)}$ 法の収束履歴

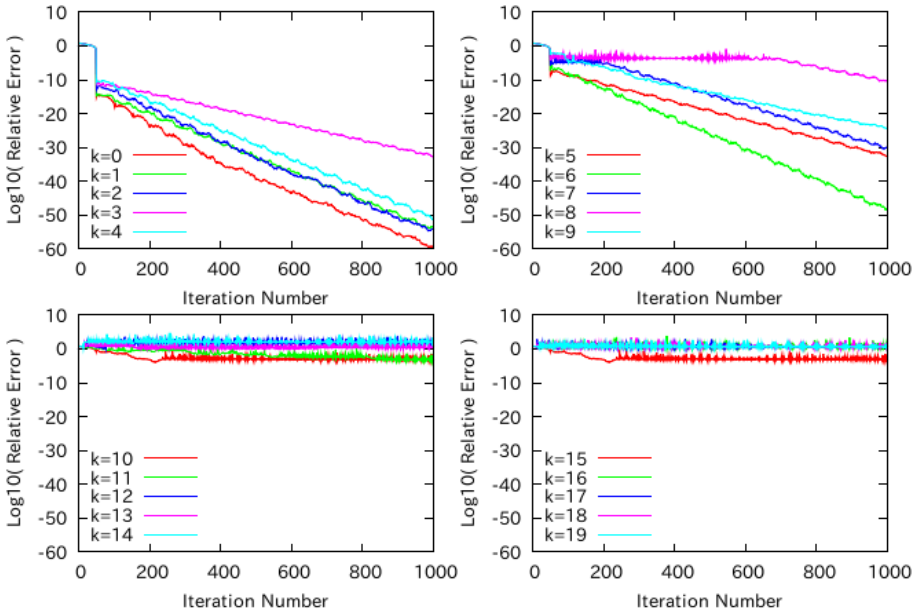

図 5 T_4 を係数行列とした場合の k-skip CG$^{(2)}$ 法の収束履歴

が，稀に発散の途中に息を吹き返して収束に向かう場合が存在する．また，CG 法本来の反復が終了した後に残差が一直線に 0 に向かう傾向が確認できるが，丸め誤差の影響が大きいとその過程で終了判定を満たすこととなる．その直線の傾きは k に依存するがこれも原因が定かでない．例えば図 5 の左上のグラフにおいて，$k = 3$ の場合のみ収束の様子が他から外れており傾きが最も緩やかである．

5. まとめと今後の課題

本稿では CG 法に頻出する内積の集団通信を回避する k-skip CG 法を 3 つ提案・実装し，先行研究の KBCG 法も含めて実験を行った．k-skip CG 法は $k+1$ 反復の内積を 1 回の集団通信で計算することにより $2k+1$ 回の集団通信を回避する反面，計算量の増大とそれに伴う誤差の増大を許容せざるを得ない．k 回の反復の内積通信を回避した場合の計算量の増大は $O(k^2)$ であり，関連手法の中では最小である．またアルゴリズムがシンプルであり，丸め誤差の影響を減少させるための計算手順の変更が容易であることも利点である．これらは動的計画法を用いたことによる．実験結果からは計算量のオーダーの低さが有効と言えるほどの k の値では収束しない場合が多く，収束したとしても反復回数が大きくなる傾向が大きい．加えて k の値により様々な挙動を示すことが明らかになっている．

総じて行列の特性と丸め誤差の影響を受けやすい性質が明らかとなっており，更に多種多様な行列での実験と丸め誤差の影響を受けにくいアルゴリズムの開発が課題である．

参 考 文 献

1) J.Demmel, L.Grigori, M.H. and Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations (2008).
2) J. W. DEMMEL, M. H. and MOHIYUDDIN, M.: Minimizing communication in sparse matrix solvers, *in Proceedings of the 2009 ACM/IEEE Conference on Supercomputing* (2009).
3) Hestenes, M.R. and Stiefel, E.: Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research of the National Bureau of Standards*, Vol.49, No.6, pp.408–436 (1952).
4) Rosendale, J.V.: MINIMIZING INNER PRODUCT DATA DEPENDENCIES IN CONJUGATE GRADIENT ITERATION, *NASA Contractor Report* (1983).
5) Saad, Y.: Practical use of polynomial preconditionings for the conjugate gradient method, *SIAM J. Sci. Statist. Comput.*, Vol.6, pp.865–881 (1985).
6) Saad, Y.: Krylov Subspace Methods on Supercomputers, *SIAM J. Sci. Statist. Comput.*, Vol.10, pp.1200–1232 (1989).
7) Meurant, G.: Multitasking the conjugate gradient method on the CRAY X-MP/48, *Prallel Comput.*, Vol.5, pp.267–280 (1987).
8) E.F.D'Azevedo, V. and C.H.Romine: Lapack Working Note 56 Conjugate Gradient Algorithms with Reduced Synchronization Overhead on Distributed Memory Multiprocessors (1999).
9) CHRONOPOULOS, A. and GEAR, C.: s-step iterative methods for symmetric linear systems, *Journal of Computational and Applied Mathematics*, Vol. 25, pp. 153–168 (1989).
10) Toledo, S.A.: Quantative Performance Modeling of Scientific Computations and Creating Locality in Numerical Algorithms, PhD Thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY (1995).
11) Hoemmen, M.: Communication-Avoiding Krylov Subspace Methods, PhD Thesis, University of California, Berkeley (2010).
12) Bellman, R.: The Theory of Dynamic Programming, *Bull. Amer. Math. Soc.*, Vol.60, No.6, pp.503–515 (1954).