Efficient Deflation for Communication-Avoiding Krylov Subspace Methods

Erin Carson
Nicholas Knight, James Demmel
Univ. of California, Berkeley

Monday, June 24, NASCA 2013, Calais, France
We derive the **Deflated Communication-Avoiding Conjugate Gradient** algorithm (Deflated CA-CG), demonstrating that deflation can be implemented while maintaining **asymptotic savings in data movement**.

1. **Background**
   - What is communication and why should it be avoided?
   - Communication-avoiding (s-step) conjugate gradient (CA-CG)
   - Deflated Conjugate Gradient method

2. **Derivation of Deflated CA-CG**

3. **Asymptotic communication and computation costs**

4. **Evaluating tradeoffs in practice**
   - Performance model and convergence results for model problem

5. **Extensions and future work**
What is Communication?

- Algorithms have two costs: **communication** and **computation**
  - **Communication**: moving data between levels of memory hierarchy (sequential), between processors (parallel)

![Diagram of memory hierarchy and communication]

- On modern computers, communication is expensive, computation is cheap
  - Flop time $\ll 1/$bandwidth $\ll$ latency
  - Communication a barrier to scalability (runtime and energy)
- **We must redesign algorithms to avoid communication**
How do Krylov Subspace Methods Work?

• A **Krylov subspace** is defined as

\[ \mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\} \]

• A **Krylov subspace method (KSM)** is a projection process onto the subspace \( \mathcal{K} \) orthogonal to \( \mathcal{L} \)
  
  • The choice of \( \mathcal{L} \) distinguishes the various methods
  
  • Examples: Conjugate Gradient (CG), Generalized Minimum Residual Methods (GMRES), Biconjugate Gradient (BICG)

KSMs for solving linear systems: in iteration \( m \), refine solution \( x_m \) to \( Ax = b \) by imposing the condition

\[ x_m = x_0 + \delta, \ \delta \in \mathcal{K}_m \ \text{and} \ \ r_0 - A\delta \perp \mathcal{L}_m, \]

where \( r_0 = b - Ax_0 \)
In each iteration, the projection process proceeds by:

1. Adding a dimension to the Krylov subspace $\mathcal{K}_m$
   - Requires **sparse matrix-vector multiplication (SpMV)**
     - Parallel: communicate vector entries with neighbors
     - Sequential: read $A$ (and $N$-vectors) from slow memory

2. Orthogonalizing with respect to $\mathcal{L}_m$
   - Requires inner products
     - Parallel: global reductions
     - Sequential: multiple reads/writes to slow memory

**Dependencies between communication-bound kernels in each iteration limit performance!**
Given: initial approximation $x_0$ for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $m = 0, 1, \ldots$, until convergence do

$$
\alpha_m = \frac{r_m^T r_m}{p_m^T Ap_m}
$$

$$
x_{m+1} = x_m + \alpha_m p_m
$$

$$
r_{m+1} = r_m - \alpha_m Ap_m
$$

$$
\beta_{m+1} = \frac{r_{m+1}^T r_{m+1}}{r_m^T r_m}
$$

$$
p_{m+1} = r_{m+1} + \beta_{m+1} p_m
$$

end for

SpMV and inner products require communication in each iteration!
CA-CG Derivation Overview

In iteration $m + s$ we have the relation

\[ p_{m+s}, r_{m+s} \in \mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m) \]
\[ x_{m+s} - x_m \in \mathcal{K}_s(A, p_m) + \mathcal{K}_{s-1}(A, r_m) \]

Let $V$ be a basis for $\mathcal{K}_{s+1}(A, p_m) + \mathcal{K}_s(A, r_m)$, and let Gram matrix $G = V^T V$.

For $1 \leq j \leq s$,

\[ p_{m+j} = V p_j' \quad r_{m+j} = V r_j' \quad x_{m+j} - x_m = V x_j' \]

where $p_j', r_j'$, and $x_j'$ are coordinates for $p_{m+j}, r_{m+j}$, and $x_{m+j} - x_m$ in basis $V$.

The product $A p_{m+j-1}$ can be written:

\[ A p_{m+j-1} = A V p_j' = V T p_j' \]

and inner products can be written:

\[ r_{m+j}^T r_{m+j} = r_j'^T G r_j' \quad p_{m+j-1}^T A p_{m+j-1} = p_j'^T G T p_j' \]
Communication-Avoiding CG

• This formulation allows an $O(s)$ reduction in communication

• Main idea: Split iteration loop into outer loop ($k$) and inner loop ($j$)

**Outer iteration: 1 communication step**

• Compute $V_k$: read $A$/communicate vectors only once (for well-partitioned $A$) using matrix powers kernel (see, e.g., Hoemmen et al., 2007)

• Compute Gram matrix $G_k = V_k^T V_k$: one global reduction

**Inner iterations: $s$ computation steps**

• Perform iterations $sk + j$, for $0 \leq j < s$, with no communication

• Update $2s + 1$-vectors of coordinates of $p_{sk+j}, r_{sk+j}, x_{sk+j} - x_{sk}$ in $V_k$, replacing SpMVs and inner products

  • Quantities either local (parallel) or fit in fast memory (sequential)

Many CA-KSMs (or s-step KSMs) derived in the literature:
(Van Rosendale, 1983), (Walker, 1988), (Leland, 1989), (Chronopoulos and Gear, 1989),
(Chronopoulos and Kim, 1990, 1992), (Chronopoulos, 1991), (Kim and Chronopoulos, 1991),
Given: initial approximation $x_0$ for solving $Ax = b$

Let $p_0 = r_0 = b - Ax_0$

for $k = 0, 1, \ldots$, until convergence do

Calculate $P_k, R_k$, bases for $\mathcal{K}_{s+1}(A, p_{sk}), \mathcal{K}_s(A, r_{sk})$, resp.

Let $V_k = [P_k, R_k]$ and compute $G_k = V_k^T V_k$

Let $x'_0 = 0_{2s+1}$, $r'_0 = [0_{s+1}^T, 1, 0_{s-1}^T]^T$, $p'_0 = [1, 0_{2s}^T]^T$

for $j = 0, \ldots, s - 1$ do

$\alpha_{sk+j} = \frac{r'_j g_k r'_j}{p_j^T g_k T_k p_j'}$
$x'_{j+1} = x'_j + \alpha_{sk+j} p_j'$
$r'_{j+1} = r'_j - \alpha_{sk+j} T_k p_j'$

$\beta_{sk+j+1} = \frac{r'_{j+1} g_k r'_{j+1}}{r'_j g_k r'_j}$
$p_{j+1} = r'_{j+1} + \beta_{sk+j+1} p_j'$

end for

Compute $x_{sk+s} = V_k x'_s + x_{sk}$, $r_{sk+s} = V_k r'_s$, $p_{sk+s} = V_k p'_s$

end for
Deflation CG (Saad et al., 2000)

- Deflation: removing eigenvalues that are hard to converge to in order to increase convergence rate
- Convergence of CG governed by $\kappa(A) = \lambda_N / \lambda_1$
  - Where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ are eigenvalues of $A$
- Let $W$ be an $N \times c$ matrix to be used in deflation
- Deflated CG is equivalent to CG with system $H^T A H \tilde{x} = H^T b$ where $H = I - W(W^T A W)^{-1}(A W)^T$ is the matrix of the $A$-orthogonal projection onto $W^\perp_A$
  - When columns of $W$ are approximate eigenvectors of $A$ associated with $\lambda_1, \lambda_2, \ldots, \lambda_c$, $\kappa(H^T A H) \approx \lambda_N / \lambda_{c+1}$
  - Deflated CG should increase rate of convergence

Can deflation techniques be applied to CA-CG while maintaining asymptotic reduction in communication cost?
Deflated CG Algorithm (Saad et al., 2000)

Define $W$ to be a length $N \times c$ basis. Compute $W^TAW$.

Compute $x_0 = W(W^TAW)^{-1}W^Tb$

$r_0 = b - Ax_0$, $\mu_0 = (W^TAW)^{-1}W^TAr_0$, $p_0 = r_0 - W\mu_0$

for $m = 0, 1, \ldots$, until convergence do

- $\alpha_m = \frac{r_m^Tr_m}{p_m^TAp_m}$
- $x_{m+1} = x_m + \alpha_m p_m$
- $r_{m+1} = r_m - \alpha_m Ap_m$
- $\beta_{m+1} = \frac{r_{m+1}^Tr_{m+1}}{r_m^Tr_m}$

Solve $W^TAW\mu_{m+1} = W^TAr_{m+1}$ for $\mu_{m+1}$

$p_{m+1} = \beta_{m+1}p_m + r_{m+1} - W\mu_{m+1}$

end for
Avoiding Communication in Deflation Process

In Deflated CG, we have

\[ p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk}) + \mathcal{K}_{s-1}(A, W) \]

\[ x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk}) + \mathcal{K}_{s-2}(A, W) \]

To compute \( \mu_{sk+j+1} \), we also need

\[ A r_{sk+j+1} \in \mathcal{K}_{s+2}(A, p_{sk}) + \mathcal{K}_{s+1}(A, r_{sk}) + \mathcal{K}_s(A, W) \]

Let \( V_k \) be an \( N \times (2s + 3 + cs) \) matrix whose columns span this space, i.e.,

\[ V_k \in \mathcal{K}_{s+2}(A, p_{sk}) + \mathcal{K}_{s+1}(A, r_{sk}) + \mathcal{K}_s(A, W). \]

If we compute \( G_k = V_k^T V_k \), and extract \( Z_k = W^T V_k \) from \( G_k \), then

\[ W^T A r_{sk+j+1} = Z_k T_k r_{k,j+1}' \]

As in CA-CG, we compute inner products and mult. by \( A \) in the inner loop by updating length-(2s + 3 + cs) coordinate vectors in basis \( V_k \).
Define $W$ to be a length $N \times c$ basis. Compute $W^T A W$.

$$x_0 = W(W^T A W)^{-1} W^T b,\ r_0 = b - A x_0,\ \mu_0 = (W^T A W)^{-1} W^T A r_0,\ p_0 = r_0 - W \mu_0$$

Compute $\mathcal{W}$, a basis for $\mathcal{K}_s(A, W)$

for $k = 0, 1, \ldots$, until convergence do

Compute $P_k, R_k$, bases for $\mathcal{K}_{s+2}(A, p_{sk}), \mathcal{K}_{s+1}(A, r_{sk})$, resp.

Construct $T_k$ such that $A[P_k, R_k, \mathcal{W}] = [P_k, R_k, \mathcal{W}]T_k$

Let $V_k = [P_k, R_k, \mathcal{W}]$, compute $G_k = V_k^T V_k, Z_k = W^T V_k$

Compute $p_0' = [1,0^T_{2s+2+cs}]^T, r_0' = [0^T_{s+2}, 1, 0^T_{s+cs}]^T, x_0' = 0_{2s+3+cs}$

for $j = 0$ to $s - 1$ do

$$\alpha_{sk+j} = r_j'^T G_k r_j' / p_j'^T G_k T_k p_j'$$
$$x_{j+1}' = x_j' + \alpha_{sk+j} p_j'$$
$$r_{j+1}' = r_j' - \alpha_{sk+j} T_k p_j'$$
$$\beta_{sk+j+1} = r_{j+1}'^T G_k r_{j+1}' / r_j'^T G_k r_j'$$

Solve $W^T A W \mu_{sk+j+1} = Z_k T_k r_j'$ for $\mu_{sk+j+1}$

$$p_{j+1}' = \beta_{sk+j+1} p_j' + r_{j+1}' - [0^T_{2s+3}, \mu_{sk+j+1}^T, 0^T_c(s-1)]^T$$

end for

Solve $W^T A W \mu_{sk+s} = Z_k T_k r_s'$ for $\mu_{sk+s}$

$$x_{sk+s} = V_k x_s' + x_{sk}, \ r_{sk+s} = V_k r_s', \ p_{sk+s} = V_k p_s'$$

end for
## Model Problem (2D Laplacian), $s$ iterations of parallel algorithm

<table>
<thead>
<tr>
<th></th>
<th>Flops</th>
<th>Words moved</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CG</strong></td>
<td>$O\left(\frac{sN}{p}\right) + O(s)$</td>
<td>$O\left(s\sqrt{N/p}\right) + O(s)$</td>
<td>$O(s \log_2 p) + O(s)$</td>
</tr>
<tr>
<td><strong>CA-CG</strong></td>
<td>$O\left(\frac{s^2N}{p}\right) + O(s^3)$</td>
<td>$O\left(s\sqrt{N/p}\right) + O(s^2)$</td>
<td>$O(\log_2 p)$</td>
</tr>
<tr>
<td><strong>Deflated CG</strong></td>
<td>$O\left(\frac{csN}{p}\right) + O(c^2s)$</td>
<td>$O\left(s\sqrt{N/p}\right) + O(cs)$</td>
<td>$O(s \log_2 p) + O(s)$</td>
</tr>
<tr>
<td><strong>Deflated CA-CG</strong></td>
<td>$O\left(\frac{cs^2N}{p}\right) + O(c^2s^3)$</td>
<td>$O\left(s\sqrt{N/p}\right) + O(cs^2)$</td>
<td>$O(\log_2 p)$</td>
</tr>
</tbody>
</table>

Note: offline costs of computing and factoring $W^TAW$ omitted for Deflated CG and Deflated CA-CG (as well as computing $\mathcal{K}_s(A,W)$ for Deflated CA-CG)
Is This Efficient in Practice?

• In practice, evaluating tradeoffs between $s$ and $c$ is nontrivial
  • Larger $s$ means faster speed per iteration, but can potentially decrease convergence rate in finite precision
  • Larger $c$ gives better theoretical convergence rate, but can potentially decrease speed per iteration

• Performance modeling for a specific problem, method, and machine must take both
  1. How time per iteration changes with $s$ and $c$
  2. How the number of iterations required for convergence changes with $s$ and $c$, and

into account.

• We will demonstrate the tradeoffs involved for our model problem (2D Laplacian) on two large distributed memory machine models
Plot of modeled speedup per iteration relative to CG for 2 machines, for 2D Laplacian with $N = 262,144, p = 512$ where

$$\text{Time} = \gamma(\text{arithmetic operations}) + \beta(\text{words moved}) + \alpha(\text{messages sent})$$

Peta:
$$\gamma = 2 \cdot 10^{-11} \text{(s/flop)}, \quad \alpha = 10^{-5} \text{(s)}, \quad \beta = 2 \cdot 10^{-9} \text{(s/word)}$$

Grid:
$$\gamma = 10^{-12} \text{(s/flop)}, \quad \alpha = 10^{-1} \text{(s)}, \quad \beta = 25 \cdot 10^{-9} \text{(s/word)}$$
Convergence for Model Problem

Monomial Basis, \( s = 10 \)

\[
\rho_0(A) = 1, \\
\rho_j(A) = A \cdot \rho_{j-1}(A)
\]

Matrix: 2D Laplacian(512), \( N = 262,144 \). Right hand side set such that true solution has entries \( x_i = 1/\sqrt{n} \). Deflated CG algorithm (DCG) from (Saad et al., 2000).
Total speedup = (speedup per iteration) \times (number of iterations(Monomial))

- Since CA-CG method suffers delayed convergence with monomial basis, higher $s$ doesn’t always give better performance (convergence fails for $s > 10$).
- On Peta, since relative latency is not as bad as on Grid, speedups decrease for large $c$ values.
A better choice of basis leads to stability for higher $s$ values:

**Newton Basis, $s = 20$**

\[
\rho_0(A) = 1,
\rho_j(A) = (A - \theta_j I)\rho_{j-1}(A)
\]

where $\theta_j$ are Leja-ordered points on $\mathcal{F}(A)$

*For details on better bases for Krylov subspaces, see, e.g., Phillipe and Reichel, 2012.*

Matrix: 2D Laplacian(512), $N = 262,144$. Right hand side set such that true solution has entries $x_i = 1/\sqrt{n}$. Deflated CG algorithm (DCG) from (Saad, et al., 2000).
Total Speedup

Total speedup = (speedup per iteration) \times (number of iterations (Newton))

- **Peta**: Speedup decreases with increasing $c$; CA deflation doesn’t lead to significant overall performance improvements over CA-CG.
- **Grid**: since $O(s)$ speedup from CA techniques remains constant for increasing $c$, CA deflation increases overall speedup for all $s$ values!
Conclusions and Future Work

• **Summary**
  - **Proof-of-concept that deflation techniques can be implemented in CA-CG in a way that still avoids communication**
    - Asymptotic bounds confirm that Deflated CA-CG maintains the $O(s)$ reduction in latency over CG and Deflated CG
    - Performance modeling demonstrates **nontrivial tradeoffs between speed per iteration and convergence rate** for different methods

• **Future Work**
  - Extending other deflation techniques to CA methods
    - Solving (slowly-changing) series of linear systems (recycling Krylov subspaces)
    - Reorthogonalization to fix instability in Deflated CA-CG ($r \perp W$ fails in finite precision, set $r = (I - W(W^TW)^{-1}W^T)r$) (Saad et al., 2000)
    - Equivalent ‘augmented’ formulations (Gaul et al., 2013)
    - Claim: CA deflation can be applied to other deflated Krylov methods
      - GMRES, MINRES, BICG(STAB), QMR, Arnoldi, Lanczos, etc., see, e.g., (Gutknecht, 2012)
Thank you!

Erin Carson: ecc2z@cs.berkeley.edu
Nick Knight: knight@cs.berkeley.edu
Extra Slides
CA-CG Derivation Overview

In iteration $sk + j$, for $s > 0$, $0 \leq j \leq s$, we exploit the relation

$$p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$$

$$x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk})$$

Let $V_k$ be a matrix whose columns span $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$.

Then for iterations $sk + 1$ through $sk + s$, we can implicitly update the length n vectors $p_{sk+j}, r_{sk+j}$, and $x_{sk+j} - x_{sk}$ by updating their coordinates (length $2s + 1$ vectors) in basis $V_k$.

$$p_{sk+j} = V_k p'_{k,j} \quad r_{sk+j} = V_k r'_{k,j} \quad x_{sk+j} - x_{sk} = V_k x'_{k,j}$$
In iteration $sk + j$, for $s > 0$, $0 \leq j \leq s$, we exploit the relation

$$p_{sk+j}, r_{sk+j} \in \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$$

$$x_{sk+j} - x_{sk} \in \mathcal{K}_s(A, p_{sk}) + \mathcal{K}_{s-1}(A, r_{sk})$$

If we compute basis $V_k$ for $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$ and compute Gram matrix $G_k = V_k^T V_k$, then for iterations $0 \leq j < s$ we can implicitly update the length $n$ vectors $p_{sk+j}, r_{sk+j}$, and $x_{sk+j} - x_{sk}$ by updating their coordinates (length $2s + 1$ vectors) in basis $V_k$.

$$p_{sk+j} = V_k p'_{k,j} \quad r_{sk+j} = V_k r'_{k,j} \quad x_{sk+j} - x_{sk} = V_k x'_{k,j}$$

The product $A p_{sk+j}$ can be represented implicitly in basis $V_k$ by $B_k p'_{k,j}$, since

$$A p_{sk+j} = A V_k p'_{k,j} = V_k B_k p'_{k,j}$$

and we can write dot products as

$$r_{sk+j}^T r_{sk+j} = r'_{k,j}^T G_k r'_{k,j} \quad p_{sk+j}^T A p_{sk+j} = p'_{k,j}^T G_k B_k p'_{k,j}$$
CA-CG Derivation Overview

The product $A p_{sk+j}$ can be represented implicitly in basis $V_k$ by $B_k p'_{k,j}$, since

$$A p_{sk+j} = AV_k p'_{k,j} = V_k B_k p'_{k,j}$$

If we compute $G_k = V_k^T V_k$, we can write dot products as

$$r_{sk+j}^T r_{sk+j} = r_{k,j}'^T G_k r_{k,j}'$$

$$p_{sk+j}^T A p_{sk+j} = p_{k,j}'^T G_k B_k p'_{k,j}$$

$B_k$ and $G_k$ are small $O(s) \times O(s)$ matrices that fits in fast/local memory; Multiplication by $B_k$ and $G_k$ require no communication!
Related Work: $s$-step methods

<table>
<thead>
<tr>
<th>Authors</th>
<th>KSM</th>
<th>Basis</th>
<th>Precond?</th>
<th>Mtx Pwrs?</th>
<th>TSQR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Rosendale, 1983</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Leland, 1989</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Walker, 1988</td>
<td>GMRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Gear, 1989</td>
<td>CG</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Kim, 1990</td>
<td>Orthomin, GMRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos, 1991</td>
<td>MINRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Kim and Chronopoulos, 1991</td>
<td>Symm. Lanczos, Arnoldi</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>de Sturler, 1991</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Related Work: s-step methods

<table>
<thead>
<tr>
<th>Authors</th>
<th>KSM</th>
<th>Basis</th>
<th>Precond?</th>
<th>Mtx Pwrs?</th>
<th>TSQR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joubert and Carey, 1992</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>No</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Kim, 1992</td>
<td>Nonsymm. Lanczos</td>
<td>Monomial</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bai, Hu, and Reichel, 1991</td>
<td>GMRES</td>
<td>Newton</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Erhel, 1995</td>
<td>GMRES</td>
<td>Newton</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>de Sturler and van der Vorst, 1995</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>General</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Toledo, 1995</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Swanson, 1996</td>
<td>CGR, Orthomin</td>
<td>Monomial</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Kinkaid, 2001</td>
<td>Orthodir</td>
<td>Monomial</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Convergence in Finite Precision

• CA-KSMs are mathematically equivalent to classical KSMs

• But have different behavior in finite precision!
  • Roundoff error causes delay of convergence
  • Bounds on magnitude of roundoff error increase with $s$

• In solving practical problems, roundoff error can limit performance
  • If # iterations increases more than time per iteration decreases due to CA techniques, no speedup expected!

• To perform a practical performance comparison amongst CG, Deflated CG, CA-CG and Deflated CA-CG, we must combine speedup per iteration with the total number of iterations for each method
Detailed Complexity Analysis: CG

\[ \text{Flops}_{CG} = 2s - \frac{2s}{p} + \frac{19ns}{p} \]

\[ \text{Words}_{CG} = 2s - \frac{2s}{p} + 4s\sqrt{n/p} \]

\[ \text{Mess}_{CG} = 4s + 2s \log_2 p \]
Detailed Complexity Analysis: CA-CG

\[
\text{Flops}_{\text{CA CG}} = 49s - \frac{2s^2}{p} + 36s^2\sqrt{\frac{n}{p}} + 25n/p - 3s/p + 72s\sqrt{\frac{n}{p}} - \frac{1}{p} + 74s^2 + 36s^3 + 36\sqrt{\frac{n}{p}} + 36ns/p + 4ns^2/p + 12
\]

\[
\text{Words}_{\text{CA CG}} = 11s - \frac{2s^2}{p} - 3s/p + 8s\sqrt{\frac{n}{p}} - \frac{1}{p} + 6s^2 + 8\sqrt{\frac{n}{p}} + 5
\]

\[
\text{Mess}_{\text{CA CG}} = \log_2 p + 8
\]
**Detailed Complexity Analysis: DCG**

\[
\text{Flops}_{\text{DCG}} = 2s + 2c^2s - 2s/p - cs/p + 29ns/p + 4cns)/p
\]

\[
\text{Words}_{\text{DCG}} = 2s + cs - 2s/p + 8s\sqrt{n/p} - cs/p
\]

\[
\text{Mess}_{\text{DCG}} = 8s + 3\text{slog}_2 p
\]
Detailed Complexity Analysis: DCACG

\[ Flops_{DCACG} = 240s - 2s^2/p + 36s^2\sqrt{n/p} + 5cs + 60cs^2 + 2c^2s + 32cs^3 + 65n/p - 7s/p + 144s\sqrt{n/p} - 6/p + 184s^2 + 44s^3 + 2c^2s^2 + 8c^2s^3 + 144\sqrt{n/p} - 3cs/p + 44ns/p - 2cs^2/p + 4ns^2/p + 12cns/p + 4cns^2/p + 96 \]

\[ Words_{DCACG} = 23s - 2s^2/p + 3cs + 2cs^2 - 7s/p + 8s\sqrt{n/p} - 6/p + 6s^2 + 16\sqrt{n/p} - 3cs/p - 2cs^2/p + 22 \]

\[ Mess_{DCACG} = \log_2 p + 8 \]
Better Polynomial Bases

In general, columns $v_{i+1}$ of $V$ computed by the 3-term recurrence

$$v_{i+1} = \left((A - \hat{\alpha}_i I)v_i - \hat{\beta}_i v_{i-1}\right)/\hat{\gamma}_i$$

Scaled Monomial: For scalars $\{\sigma_i\}_{i=1}^S$,

$$\hat{\alpha}_i = 0, \hat{\beta}_i = 0, \hat{\gamma}_i = \sigma_i$$

Newton: For Leja-ordered Ritz values $\{\theta_i\}_{i=1}^S$ and scalars $\{\sigma_i\}_{i=1}^S$

$$\hat{\alpha}_i = \theta_i, \hat{\beta}_i = 0, \hat{\gamma}_i = \sigma_i$$

Chebyshev: Given bounding ellipse for spectrum with foci at $d \pm c$, the scaled and shifted polynomials are

$$\tilde{\tau}_i(z) = \tau_i((d - z)/c)/\sigma_i,$$

$$\hat{\alpha}_i = d, \quad \hat{\beta}_i = -\frac{c\sigma_i}{2\sigma_{i+1}}, \quad \hat{\gamma}_i = -\frac{c\sigma_{i+1}}{2\sigma_i}$$
Leja Ordering

Let $\mathbb{S}$ be a compact set in $\mathbb{C}$. (Note that here $\mathbb{S}$ is set of approx. Ritz values). Then a Leja ordering can be computed by

$$
\theta_1 = \arg\max_{z \in \mathbb{S}} |z|
$$

$$
\theta_{i+1} = \arg\max_{z \in \mathbb{S}} \prod_{k=0}^{i} |z - z_k|
$$

Leja (1957)

Many references for using different polynomials for Krylov subspace calculation:

Residual Replacement Strategy

• Van der Vorst and Ye (1999): Residual replacement used in combination with group-update to improve the maximum attainable accuracy
  • Given computable upper bound for deviation of residuals, replacement steps chosen to satisfy two constraints:
    1. Deviation must not grow so large that attainable accuracy is limited
    2. Replacement must not perturb Lanczos recurrence relation for computed residuals such that convergence deteriorates

• When the computed residual converges to level $O(\varepsilon)\|A\|\|x\|$ strategy reduces true residual, to level $O(\varepsilon)\|A\|\|x\|$

• We devise an analogous strategy for CA-CG and CA-BICG
  • Our strategy does not asymptotically increase communication or computation!
- Matrix: consph (FEM), SPD, $N = 8.3E4$, $NZ = 6E6$, $\kappa(A) \approx 9.7E3$.
- In all tests #replacements $\leq 5$.
- **Orders of magnitude improvement in accuracy** for little additional cost!
- But doesn’t fix slow convergence due to ill-conditioned basis.
Total Speedup, Monomial Basis

Total speedup = (speedup per iteration) × (number of iterations(Monomial))

Since the CA-CG method without deflation suffers delayed convergence, deflation results in performance improvements on both machines (note that delayed convergence also means that higher s doesn’t always give better performance for monomial). On Peta, since relative latency is not as bad as on Grid, speedups start to decrease for large c values.
Overlapping Communication and Computation

Plot of modeled speedup per iteration relative to CG for 2 machines, for 2D Laplacian with \( n = 262,144 \), \( p = 512 \) where

\[
\text{Time} = \max(\gamma(\text{arithmetic operations}), \beta(\text{words moved}) + \alpha(\text{messages sent}))
\]

Peta:
\[
\gamma = 2 \cdot 10^{-11}(\text{s/flop}), \quad \alpha = 10^{-5}(\text{s}), \quad \beta = 2 \cdot 10^{-9}(\text{words/s})
\]

Grid:
\[
\gamma = 10^{-12}(\text{s/flop}), \quad \alpha = 10^{-1}(\text{s}), \quad \beta = 25 \cdot 10^{-9}(\text{words/s})
\]
Total Speedup with Overlap (Newton)

Total speedup = (speedup per iteration) \times (number of iterations (Newton))

**Peta:** Overlapping communication and computation decreases cost of increasing c, so deflation results in performance improvement

**Grid:** Overlapping communication and computation doesn’t change anything, since extremely communication bound
Total Speedup with Overlap (Monomial)

Total speedup = (speedup per iteration) \times (number of iterations (Monomial))

**Peta:** Overlapping communication and computation decreases cost of increasing $c$; deflation results in speedup (but since convergence decreases $s$, best speedup at $s=8$).

**Grid:** Overlapping communication and computation doesn’t change anything, since extremely communication bound
The Matrix Powers Kernel

- Compute dependencies up front for computing $Av, A^2v, \ldots, A^s v$
  - $s$ steps of the transitive closure of $A$
  - Only need to read $A$ once assuming $A$ is well-partitioned

Figures: [MHDY09]