Parallel Performance and Numerical Stability of Communication Hiding Pipelined Krylov Subspace Methods

SIAM Conference on Applied Linear Algebra (SIAM-ALA18), Hong Kong Baptist University, May 4-8, 2018

University of Antwerp* [Belgium], LBNL‡ [USA], INRIA Bordeaux† [France]

S. Cools*, J. Cornelis*, W. Vanroose*, T. Mijieux*, P. Ghysels‡, E. F. Yetkin†, E. Agullo†, L. Giraud†

*E-mail: siegfried.cools@uantwerp.be

Universiteit Antwerpen

Exascale systems projection

	Today's Systems	Predicted Exascale Systems*	Factor Improvement
System Peak	10^{16} flops/s	10^{18} flops/s	100
Node Memory Bandwidth	10^{2} GB/s	10^{3} GB/s	10
Interconnect Bandwidth	10^{1} GB/s	10^{2} GB/s	10
Memory Latency	10^{-7} s	$5 \cdot 10^{-8}$ s	2
Interconnect Latency	10^{-6} s	$5 \cdot 10^{-7}$ s	2

*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)

- Data movement (communication) is much more expensive than flops (computation) in terms of both time and energy
- Reducing time spent moving data/waiting for data will be essential for exascale applications

$\Rightarrow$ Communication avoiding / Communication hiding

Communication hiding vs. communication avoiding

Communication cost has motivated several approaches to reducing global synchronization cost in Krylov subspace methods:

Avoiding communication: *s*-step Krylov subspace methods *

[A. Chronopoulous, J. Demmel, M. Hoemmen, E. Carson, L. Grigori, J. Erhel, . . .]

- Compute iterations in blocks of s (change of Krylov subspace basis)
- Reduces number of synchronizations per iteration by a factor of $\mathcal{O}(s)$

Hiding communication: Pipelined Krylov subspace methods *

[P. Ghysels, W. Vanroose, S. C., P. Sanan, B. Gropp, I. Yamazaki, . . .]

- Introduce auxiliary vectors to decouple SpMV and inner products
- Enables overlapping of communication and computation

* Both equivalent to corresponding Krylov subspace methods in exact arithmetic

General concepts

Iteratively improve an approximate solution of linear system $Ax = b$,

$$x_i \in x_0 + \mathcal{K}_i(A, r_0) = x_0 + \mathrm{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

General concepts

Iteratively improve an approximate solution of linear system $Ax = b$,

$$x_i \in x_0 + \mathcal{K}_i(A, r_0) = x_0 + \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

- minimize certain error measure over Krylov subspace $\mathcal{K}_i(A, r_0)$
- Krylov subspace methods:

  Conjugate Gradients (CG), Lanczos, GMRES, MinRES, BiCG, CGS, BiCGStab, CGLS, ...
- Preconditioners:

  AMG & GMG, Domain Decomposition Methods, FETI, BDDC, Incomplete factorization, Physics based preconditioners, ...

General concepts

Iteratively improve an approximate solution of linear system $Ax = b$,

$$x_i \in x_0 + \mathcal{K}_i(A, r_0) = x_0 + \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

- minimize certain error measure over Krylov subspace $\mathcal{K}_i(A, r_0)$
- Krylov subspace methods:

  Conjugate Gradients (CG), Lanczos, GMRES, MinRES, BiCG, CGS, BiCGStab, CGLS, ...
- Preconditioners:

  AMG & GMG, Domain Decomposition Methods, FETI, BDDC, Incomplete factorization, Physics based preconditioners, ...
- usually in combination with sparse linear algebra/stencil application
- three algorithmic building blocks:
  i. dot-product
     - $\mathcal{O}$(N) flops
     - global synchronization (MPI_Allreduce)
  ii. SpMV
     - $\mathcal{O}$(nnz) flops
     - neighbor communication only
  iii. axpy
     - $\mathcal{O}$(N) flops
     - no communication

Classical CG

Algorithm CG

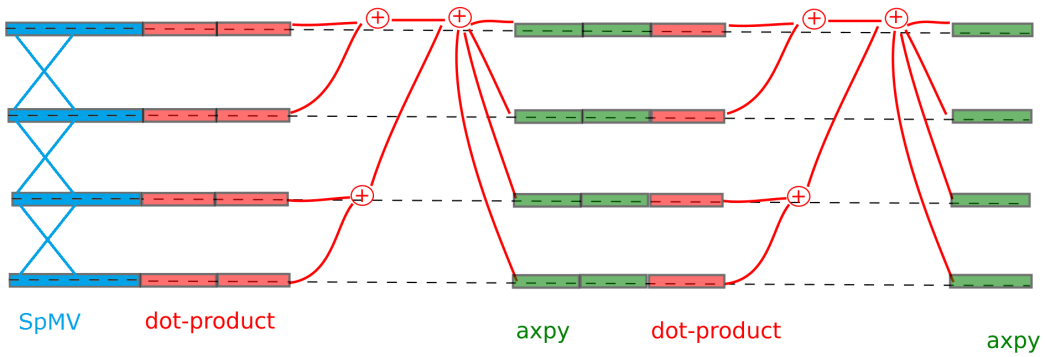
```
procedure CG(A, b, x_0)
    r_0 := b − A x_0; p_0 = r_0
    for i = 0, ... do
        s_i := A p_i                                  dot-pr
        α_i := (r_i, r_i) / (s_i, p_i)                SpMV
        x_{i+1} := x_i + α_i p_i                      axpy
        r_{i+1} := r_i − α_i s_i
        β_{i+1} := (r_{i+1}, r_{i+1}) / (r_i, r_i)
        p_{i+1} := r_{i+1} + β_{i+1} p_i
    end for
end procedure
```

Hestenes & Stiefel (1952)

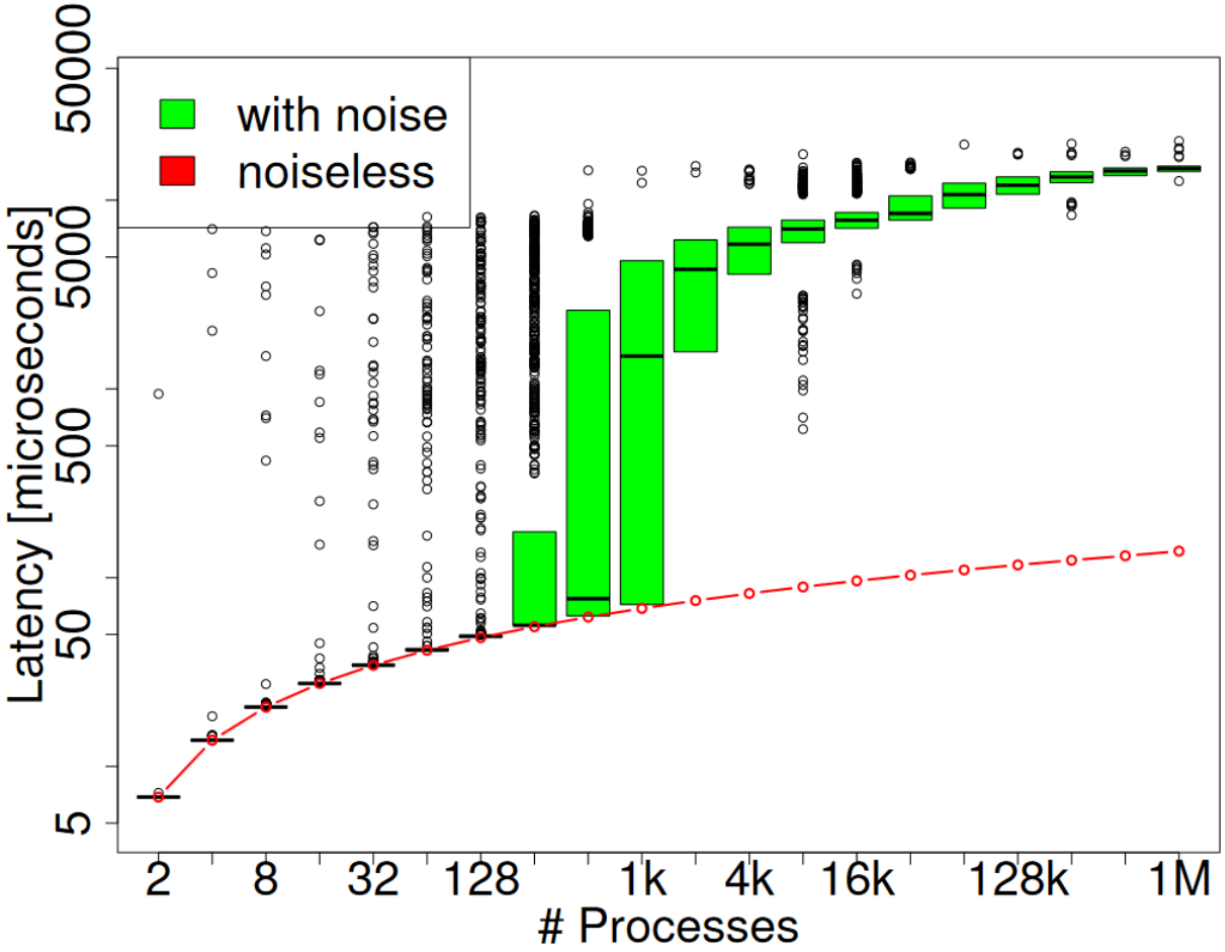
i. dot-products
- 2 global reductions: latency dominated
- scales as $log_2(\#partitions)$

ii. SpMV
- computationally expensive
- good scaling (minor comm.)

iii. axpy's
- vector operations (recurrence relations)
- perfect scaling (no comm.)



Global reduction latency



T. Hoeffler, T. Schneider and A. Lumsdaine, SC10, 2010

Pipelined CG

Algorithm Pipelined CG

procedure PIPE-CG(A, b, x_0)
  $r_0 := b - Ax_0$; $w_0 := Ar_0$
  for $i = 0, \ldots$ **do**
    $\gamma_i := (r_i, r_i)$
    $\delta := (w_i, r_i)$
    $q_i := Aw_i$
    if $i > 0$ **then**
      $\beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}$
    else
      $\beta_i := 0$; $\alpha := \gamma_i/\delta$
    end if
    $z_i := q_i + \beta_i z_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $p_i := r_i + \beta_i p_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $w_{i+1} := w_i - \alpha_i z_i$
  end for
end procedure

dot-pr
SpMV
axpy

Ghysels & Vanroose (2014)

Pipelined CG = re-engineered version of CG for improved parallel performance

Pipelined CG

Algorithm Pipelined CG

```
procedure PIPE-CG(A, b, x_0)
    r_0 := b − A x_0; w_0 := A r_0
    for i = 0, ... do
        γ_i := (r_i, r_i)
        δ := (w_i, r_i)
        q_i := A w_i
        if i > 0 then
            β_i := γ_i/γ_{i−1}; α_i := (δ/γ_i − β_i/α_{i−1})^{−1}
        else
            β_i := 0; α := γ_i/δ
        end if
        z_i := q_i + β_i z_{i−1}
        s_i := w_i + β_i s_{i−1}
        p_i := r_i + β_i p_{i−1}
        x_{i+1} := x_i + α_i p_i
        r_{i+1} := r_i − α_i s_i
        w_{i+1} := w_i − α_i z_i
    end for
end procedure
```

dot-pr
SpMV
axpy

Ghysels & Vanroose (2014)

Pipelined CG = re-engineered version of CG for improved parallel performance

- Re-ordering of operations requires new auxiliary variables: $s_i = Ap_i$, $w_i = Ar_i$, $z_i = As_i$

Pipelined CG

Algorithm Pipelined CG

```
procedure PIPE-CG(A, b, x_0)
    r_0 := b − A x_0; w_0 := A r_0
    for i = 0, ... do
        γ_i := (r_i, r_i)                                   dot-pr
        δ := (w_i, r_i)                                     dot-pr
        q_i := A w_i                                        SpMV
        if i > 0 then
            β_i := γ_i/γ_{i−1}; α_i := (δ/γ_i − β_i/α_{i−1})^{−1}
        else
            β_i := 0; α := γ_i/δ
        end if
        z_i := q_i + β_i z_{i−1}                            axpy
        s_i := w_i + β_i s_{i−1}                            axpy
        p_i := r_i + β_i p_{i−1}                            axpy
        x_{i+1} := x_i + α_i p_i                            axpy
        r_{i+1} := r_i − α_i s_i                            axpy
        w_{i+1} := w_i − α_i z_i                            axpy
    end for
end procedure
```

Ghysels & Vanroose (2014)

Pipelined CG = re-engineered version of CG for improved parallel performance

- Re-ordering of operations requires new auxiliary variables: $s_i = Ap_i$, $w_i = Ar_i$, $z_i = As_i$
- Derive recurrence relations to avoid computing additional SpMV's, e.g.:

$$s_i := Ap_i$$

$$p_i = r_i + \beta_i p_{i-1}$$

$$\Downarrow$$

$$s_i = w_i + \beta_i s_{i-1}$$

with $w_i := Ar_i$

Pipelined CG

Algorithm Pipelined CG

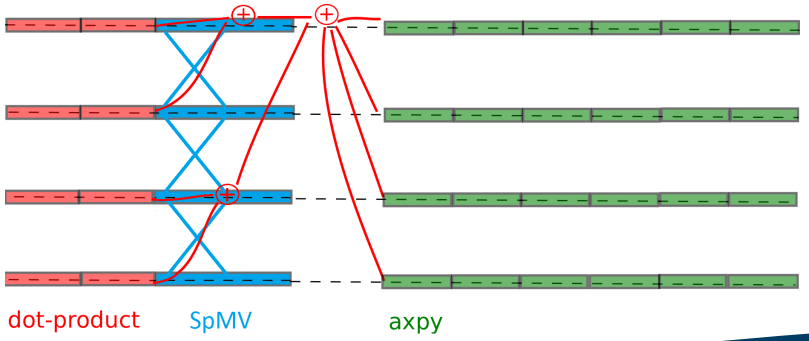
```
procedure PIPE-CG(A, b, x_0)
    r_0 := b − A x_0; w_0 := A r_0
    for i = 0, ... do
        γ_i := (r_i, r_i)
        δ := (w_i, r_i)
        q_i := A w_i
        if i > 0 then
            β_i := γ_i/γ_{i−1}; α_i := (δ/γ_i − β_i/α_{i−1})^{−1}
        else
            β_i := 0; α := γ_i/δ
        end if
        z_i := q_i + β_i z_{i−1}
        s_i := w_i + β_i s_{i−1}
        p_i := r_i + β_i p_{i−1}
        x_{i+1} := x_i + α_i p_i
        r_{i+1} := r_i − α_i s_i
        w_{i+1} := w_i − α_i z_i
    end for
end procedure
```

dot-pr
SpMV
axpy

i. Communication avoiding: dot-products grouped in one global reduction phase per iteration

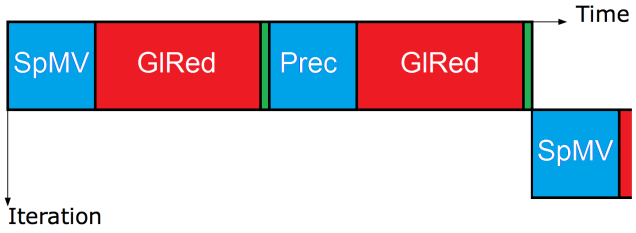
ii. Communication hiding: overlap global synchronization with SpMV (+ Prec) computation

iii. No free lunch: Additional recurrence relations, i.e. axpy's, for auxiliary var's $s_i = Ap_i$, $w_i = Ar_i$, $z_i = As_i$

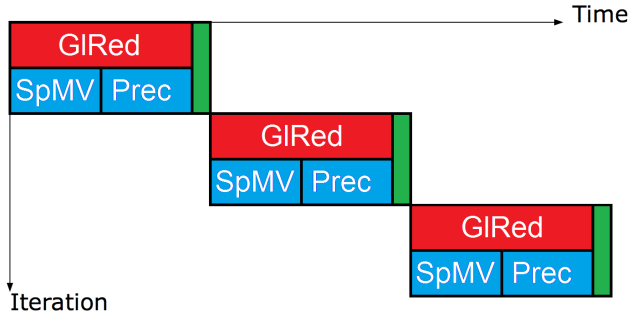


Pipelined KSM with deep pipeline

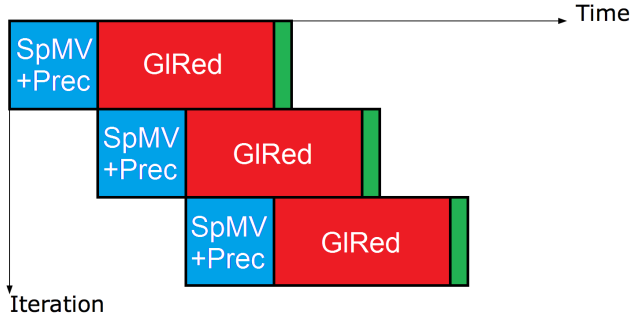
Classic KSM:



Pipelined KSM:

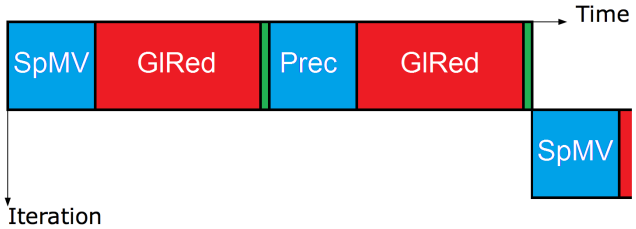


Deep pipelined KSM:

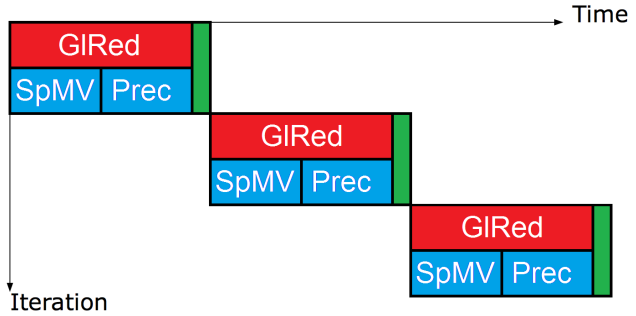


Pipelined KSM with deep pipeline

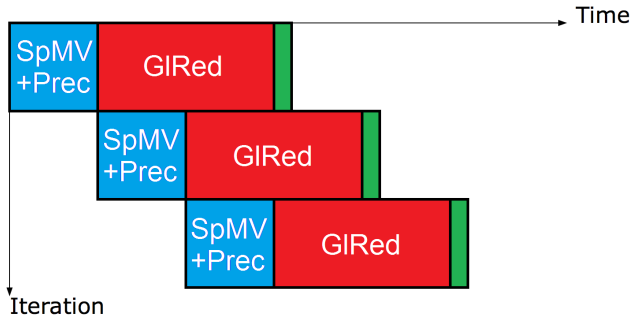
Classic KSM:



Pipelined KSM:



Deep pipelined KSM:



Consider the Arnoldi relation

$$AV_i = V_{i+1} H_{i+1,i}$$

with V_i the Krylov subspace basis and $H_{i+1,i}$ upper Hessenberg. Introduce the auxiliary vectors $Z_{i+1} = [z_0, z_1, \ldots, z_i]$ as

$$z_j := \begin{cases} v_0 & j = 0, \\ P_j(A) v_0 & 0 < j \leq l, \\ P_l(A) v_{j-l} & j > l, \end{cases}$$

with polynomials $P_l(t)$ of fixed order l, where l is the pipeline length

$$P_l(t) := \prod_{j=0}^{l-1} (t - \sigma_j).$$

Pipelined KSM with deep pipeline

Writing the Arnoldi relation $AV_i = V_{i+1}H_{i+1,i}$ for basis vector v_{j-l} and applying $P_l(A)$ to both sides, we obtain recurrence relations for z_j:

$$v_{j-l} = \frac{Av_{j-l-1} - \sum_{k=0}^{j-l-1} h_{k,j-l-1}v_k}{h_{j-l,j-l-1}} \quad \Rightarrow \quad z_j = \frac{Az_{j-1} - \sum_{k=0}^{j-l-1} h_{k,j-l-1}z_{k+l}}{h_{j-l,j-l-1}}.$$

This implies an Arnoldi-like relation for the auxiliary variables

$$AZ_i = Z_{i+1}B_{i+1,i}$$

with

$$B_{i+1,i} := \begin{pmatrix} \sigma_0 & & & & & & \\ 1 & \ddots & & & & & \\ & \ddots & \sigma_l & & & & \\ & & 1 & h_{0,0} & \cdots & h_{0,i-l} \\ & & & h_{1,0} & & h_{1,i-l} \\ & & & & \ddots & \vdots \\ & & & & & h_{i+1-1,i-l} \end{pmatrix}.$$

Pipelined KSM with deep pipeline

Basis transformation: *Z_i and V_i both span the i-th Krylov subspace, thus*

$$Z_i = V_i G_i,$$

where G_i is an upper triangular matrix.

Recursive calculation of entries G_i:

$$\begin{aligned} g_{j,k-1} &= (z_{k-1}, v_j) = \left(z_{k-1}, \frac{z_j - \sum_{m=0}^{j-1} g_{m,j} v_m}{g_{j,j}} \right) \\ &= \frac{(z_{k-1}, z_j) - \sum_{m=0}^{j-1} g_{m,j} g_{m,k-1}}{g_{j,j}} \qquad (\forall j = 0, 1, \ldots, k-1) \end{aligned}$$

Diagonal element:

$$g_{j,j} = \frac{(z_j, z_j) - \sum_{m=0}^{j-1} g_{m,j} g_{m,j}}{g_{j,j}} \quad \Rightarrow \quad g_{j,j} = \sqrt{(z_j, z_j) - \sum_{m=0}^{j-1} g_{m,j}^2}.$$

Note that this can lead to breakdowns.

Pipelined KSM with deep pipeline

Recursive calculation of the Hessenberg matrix $H_{i+1,i}$:

$$
\begin{aligned}
& H_{k+1,k} = V_{k+1}^T A V_k \\
= \quad & V_{k+1}^T \underbrace{A Z_k}\, G_k^{-1} \quad \text{using} \quad Z_k = V_k G_k \\
= \quad & \underbrace{V_{k+1}^T Z_{k+1}}\, B_{k+1,k} G_k^{-1} \quad \text{using} \quad A Z_k = Z_{k+1} B_{k+1,k} \\
= \quad & G_{k+1} B_{k+1,k} G_k^{-1} \quad \text{using} \quad G_{k+1} = V_{k+1}^T Z_{k+1} \\
= \quad & \begin{pmatrix} G_k & g_{:,k+1} \\ 0 & g_{k+1,k+1} \end{pmatrix} \begin{pmatrix} B_{k,k-1} & b_{:,k} \\ 0 & b_{k+1,k} \end{pmatrix} \begin{pmatrix} G_k & g_{:,k+1} \\ 0 & g_{k+1,k+1} \end{pmatrix}^{-1} \\
= \quad & \begin{pmatrix} G_k B_{k,k-1} & G_k b_{:,k} + g_{:,k+1} b_{k+1,k} \\ 0 & g_{k+1,k+1} b_{k+1,k} \end{pmatrix} \begin{pmatrix} G_{k-1}^{-1} & -G_{k-1}^{-1} g_{:,k} g_{k,k}^{-1} \\ 0 & g_{k,k}^{-1} \end{pmatrix} \\
= \quad & \begin{pmatrix} G_k B_{k,k-1} G_{k-1}^{-1} & \left(-G_k B_{k,k-1} G_{k-1}^{-1} g_{:,k} + G_k b_{:,k} + g_{:,k+1} b_{k+1,k}\right) g_{k,k}^{-1} \\ 0 & g_{k+1,k+1} b_{k+1,k} g_{k,k}^{-1} \end{pmatrix} \\
= \quad & \begin{pmatrix} H_{k,k-1} & \left(G_k b_{:,k} + g_{:,k+1} b_{k+1,k} - H_{k,k-1} g_{:,k}\right) g_{k,k}^{-1} \\ 0 & g_{k+1,k+1} b_{k+1,k} g_{k,k}^{-1} \end{pmatrix} \qquad H_{k,k-1} G_{k-1} = G_k B_{k,k-1}
\end{aligned}
$$

Pipelined GMRES with deep pipeline

Algorithm p(l)-GMRES

$r_0 \leftarrow b - Ax_0\,; \quad v_0 \leftarrow r_0/||r_0||\,; \quad z_0 \leftarrow v_0$
for $i = 0, \ldots, m+l$ **do**
  $w \leftarrow \begin{cases} (A - \sigma_i I) z_i, & i < l \\ A z_i, & i \geq l \end{cases}$ ← SpMV Az_i
  $a \leftarrow i - l$
  if $a \geq 0$ **then**
    $g_{j,a+1} \leftarrow (g_{j,a+1} - \sum_{k=0}^{j-1} g_{k,j} g_{k,a+1})/g_{j,j}, \; j = a-l+2, \ldots, a$ ← scalar update G_i
    $g_{a+1,a+1} \leftarrow \sqrt{g_{a+1,a+1} - \sum_{k=0}^{a} g_{k,a+1}^2}$
    # Check for breakdown and restart or re-orthogonalize if necessary
    if $a < l$ **then**
      $h_{j,a} \leftarrow (g_{j,a+1} + \sigma_a g_{j,a} - \sum_{k=0}^{a-1} h_{j,k} g_{k,a})/g_{a,a}, \; j = 0, \ldots, a$ ← scalar update $H_{i+1,i}$
      $h_{a+1,a} \leftarrow g_{a+1,a+1}/g_{a,a}$
    else
      $h_{j,a} \leftarrow (\sum_{k=0}^{a+1-l} g_{j,k+l} h_{k,a-l} - \sum_{k=j-1}^{a-1} h_{j,k} g_{k,a})/g_{a,a}, \; j = 0, \ldots, a$
      $h_{a+1,a} \leftarrow g_{a+1,a+1} h_{a+1-l,a-l}/g_{a,a}$
    end if
    $v_a \leftarrow (z_a - \sum_{j=0}^{a-1} g_{j,a} v_j)/g_{a,a}$ ← axpy's v_{i-l}, z_{i+1}
    $z_{i+1} \leftarrow (w - \sum_{j=0}^{a-1} h_{j,a-1} z_{j+l})/h_{a,a-1}$
  end if
  $g_{j,i+1} \leftarrow \begin{cases} \langle z_{i+1}, v_j \rangle, & j = 0, \ldots, a \\ \langle z_{i+1}, z_j \rangle, & j = a+1, \ldots, i+1 \end{cases}$ ← dot-prs (z_{i+1}, v_j), (z_{i+1}, z_j)
end for
$y_m \leftarrow \text{argmin}||(H_{m+1,m} y_m - ||r_0|| e_1)||_2$
$x \leftarrow x_0 + V_m y_m$

Ghysels & Vanroose (2013)

Pipelined GMRES with deep pipeline

Algorithm p(l)-GMRES

$r_0 \leftarrow b - Ax_0\,; \quad v_0 \leftarrow r_0/||r_0||\,; \quad z_0 \leftarrow v_0$

for $i = 0, \ldots, m+l$ **do**

$$w \leftarrow \begin{cases} (A - \sigma_i I) z_i, & i < l \\ A z_i, & i \geq l \end{cases}$$

$a \leftarrow i - l$

if $a \geq 0$ **then**

$g_{j,a+1} \leftarrow (g_{j,a+1} - \sum_{k=0}^{j-1} g_{k,j} g_{k,a+1})/g_{j,j}, \; j = a-l+2, \ldots, a$

$g_{a+1,a+1} \leftarrow \sqrt{g_{a+1,a+1} - \sum_{k=0}^{a} g_{k,a+1}^2}$

Check for breakdown and restart or re-orthogonalize if necessary

if $a < l$ **then**

$h_{j,a} \leftarrow (g_{j,a+1} + \sigma_a g_{j,a} - \sum_{k=0}^{a-1} h_{k,a} g_{k,j})/g_{j,j}, \; j = 0, \ldots, a$

$h_{a+1,a} \leftarrow g_{a+1,a+1}$

else

$h_{j,a} \leftarrow (\sum_{k=0}^{a+1-l} g_{k,a+1-l} g_{k,j} - \sum_{k=0}^{a-1} h_{k,a} g_{k,j})/g_{a,a}, \; j = 0, \ldots, a$

$h_{a+1,a} \leftarrow g_{a+1,a+1} - \sum_{k=0}^{a} g_{k,a+1} h_{k,a}$

end if

$v_a \leftarrow (z_a - \sum_{j=0}^{a-1} g_{j,a} v_j)/g_{a,a}$

$z_{a+1} \leftarrow (w - \sum_{j=0}^{a-1} h_{j,a-1} z_{j+l})/h_{a,a-1}$

end if

$$g_{j,i+1} \leftarrow \begin{cases} \langle z_{i+1}, v_j \rangle, & j = 0, \ldots, a \\ \langle z_{i+1}, z_j \rangle, & j = a+1, \ldots, i+1 \end{cases}$$

end for

$y_m \leftarrow \text{argmin}||(H_{m+1,m} y_m - ||r_0|| e_1)||_2$

$x \leftarrow x_0 + V_m y_m$

Results of dot-products needed l iterations later to update $G_{:,i-l+1}$

← SpMV Az_i

← scalar update G_i

← scalar update $H_{i+1,i}$

← axpy's v_{i-l}, z_{i+1}

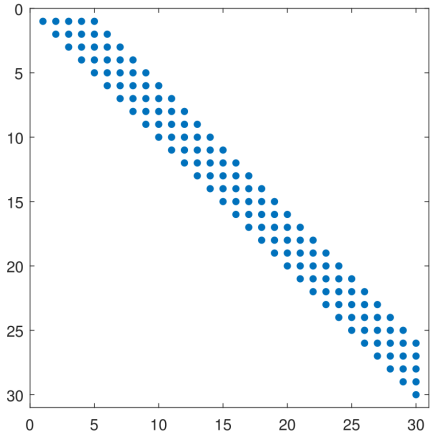
← dot-prs (z_{i+1}, v_j), (z_{i+1}, z_j)

Ghysels & Vanroose (2013)

Pipelined CG with deep pipeline

In case of an SPD matrix A, the following observations simplify the algorithm:

- $H_{i+1,i}$ is tridiagonal
- G_i has a $(2l+1)$-nonzero diagonal band structure.
  E.g. $l = 2$:



Cornelis et al. (2018)

- The solution x_i can be constructed using a recursively computed search direction p_i, instead of using the entire Krylov basis V_i:

$$p_j = (v_j - \delta_{j-1} p_{j-1})/\eta_j, \qquad x_j = x_{j-1} + \xi_{j-1} p_{j-1}.$$

Liesen & Strakos, *Krylov Subspace Methods* (2012)

Pipelined CG with deep pipeline

Algorithm Deep pipelined Conjugate Gradients (p(ℓ)-CG)

$r_0 := b - Ax_0;$
$v_0 := r_0/\|r_0\|_2;$
$z_0 := v_0;\ g_{0,0} := 1;$
for $i = 0, \ldots, m + l$ **do**
  $z_{i+1} := \begin{cases} (A - \sigma_i I)z_i, & i < l \\ Az_i, & i \geq l \end{cases}$
  $a := i - l$
  if $a \geq 0$ **then**
    $g_{j,a+1} := (g_{j,a+1} - \sum_{k=a+1-2l}^{j-1} g_{k,j} g_{k,a+1})/g_{j,j}; \qquad j = a - l + 2, \ldots, a$
    $g_{a+1,a+1} := \sqrt{g_{a+1,a+1} - \sum_{k=a+1-2l}^{a} g_{k,a+1}^2};$
    # Check for breakdown and restart if required
    if $a < l$ **then**
      $\gamma_a := (g_{a,a+1} + \sigma_a g_{a,a} - \delta_{a-1} g_{a-1,a})/g_{a,a};$
      $\delta_a := g_{a+1,a+1}/g_{a,a};$
    else
      $\gamma_a := (g_{a,a}\gamma_{a-l} + g_{a,a+1}\delta_{a-l} - \delta_{a-1} g_{a-1,a})/g_{a,a};$
      $\delta_a := (g_{a+1,a+1}\delta_{a-l})/g_{a,a};$
    end if
    $v_{a+1} := (z_{a+1} - \sum_{j=a-2l+1}^{a} g_{j,a+1} v_j)/g_{a+1,a+1};$
    $z_{i+1} := (z_{i+1} - \gamma_a z_i - \delta_{a-1} z_{i-1})/\delta_a;$
  end if
  if $a < 0$ **then**
    $g_{j,i+1} := (z_{i+1}, z_j); \qquad j = 0, \ldots, i + 1$
  else
    $g_{j,i+1} := \begin{cases} (z_{i+1}, v_j); & j = \max(0, i - 2l + 1), \ldots, a + 1 \\ (z_{i+1}, z_j); & j = a + 2, \ldots, i + 1 \end{cases}$
  end if

← SpMV Az_i

← scalar update G_i

← scalar update $H_{i+1,i}$

← axpy's v_{i-l+1}, z_{i+1}

← dot-prs (z_{i+1}, v_j), (z_{i+1}, z_j)

Cornelis et al. (2018)

Pipelined CG with deep pipeline

Algorithm Deep pipelined Conjugate Gradients (p(ℓ)-CG)

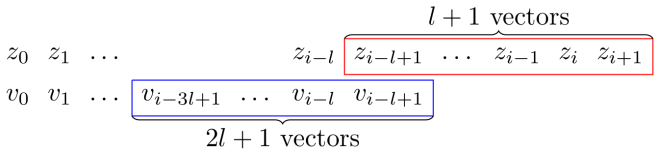
$r_0 := b - Ax_0;$
$v_0 := r_0/\|r_0\|_2;$
$z_0 := v_0;\ g_{0,0} := 1;$
for $i = 0, \ldots, m+l$ **do**
  $z_{i+1} := \begin{cases} (A-\sigma_i I)z_i, & i < l \\ Az_i, & i \geq l \end{cases}$
  $a := i - l$
  if $a \geq 0$ **then**
    $g_{j,a+1} := (g_{j,a+1} - \sum_{k=a+1-2l}^{j-1} g_{k,j}g_{k,a+1})/g_{j,j}; \qquad j = a-l+$
    $g_{a+1,a+1} := \sqrt{g_{a+1,a+1} - \sum_{k=a+1-2l}^{a} g_{k,a+1}^2};$
    # Check for breakdown and restart if required
    if $a < l$ **then**
      $\gamma_a := (g_{a,a+1} + \sigma_a g_{a,a} - \delta_{a-1}g_{a-1,a})/g_{a,a};$
      $\delta_a := g_{a+1,a+1}/g_{a,a};$
    else
      $\gamma_a := (g_{a,a}\gamma_{a-l} + g_{a,a+1}\delta_{a-l} - \delta_{a-1}g_{a-1,a})/g_{a,a};$
      $\delta_a := (g_{a+1,a+1}\delta_{a-l})/g_{a,a};$
    end if
    $v_{a+1} := (z_{a+1} - \sum_{j=a-2l+1}^{a} g_{j,a+1}v_j)/g_{a+1,a+1};$
    $z_{i+1} := (z_{i+1} - \gamma_a z_i - \delta_{a-1}z_{i-1})/\delta_a;$
  end if
  if $a < 0$ **then**
    $g_{j,i+1} := (z_{i+1}, z_j); \qquad j = 0, \ldots, i+1$
  else
    $g_{j,i+1} := \begin{cases} (z_{i+1}, v_j); & j = \max(0, i-2l+1), \ldots, a+1 \\ (z_{i+1}, z_j); & j = a+2, \ldots, i+1 \end{cases}$
  end if

SpMV Az_i

(z_{i+1}, z_j)

Shorter recurrences compared to p(l)-GMRES

- computation: $2l+2$ axpy's
- storage: $3l+2$ basis vectors



Fewer dot-products compared to p(l)-GMRES

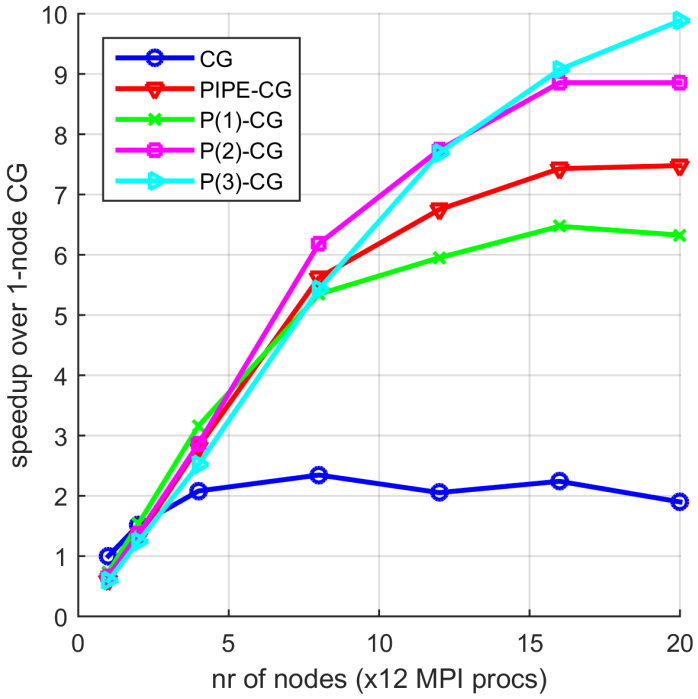
- $2l+1$ band structure of G_i

Cornelis et al. (2018)

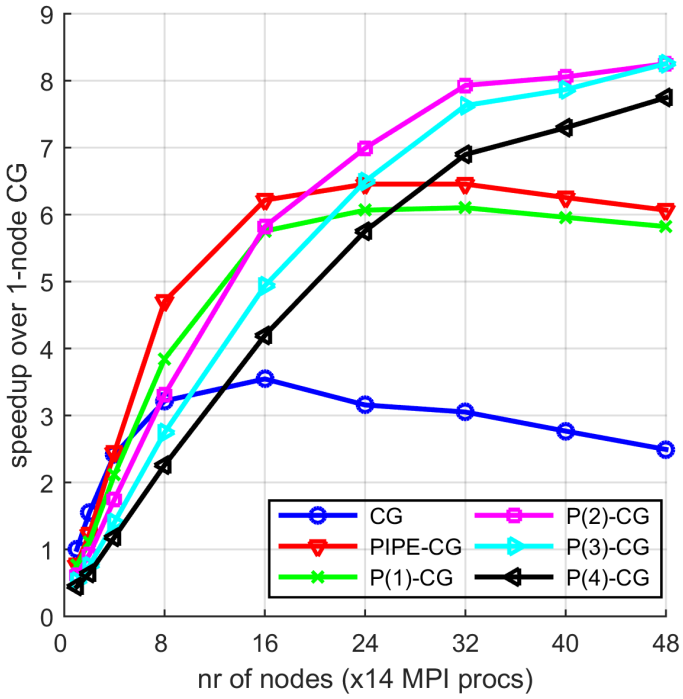
Parallel performance of pipelined CG

Strong scaling experiments - PETSc 3.6.3/3.7.6 library - MPICH 3.1.3/3.3a2

Two 6-core Intel Xeon X5660 Nehalem 2.80 GHz per node - 2D Poisson (5pt) - 1 million unknowns



Two 14-core Intel E5-2680v4 Broadwell 2.40 GHz per node - 2D Poisson (5pt) - 3 million unknowns



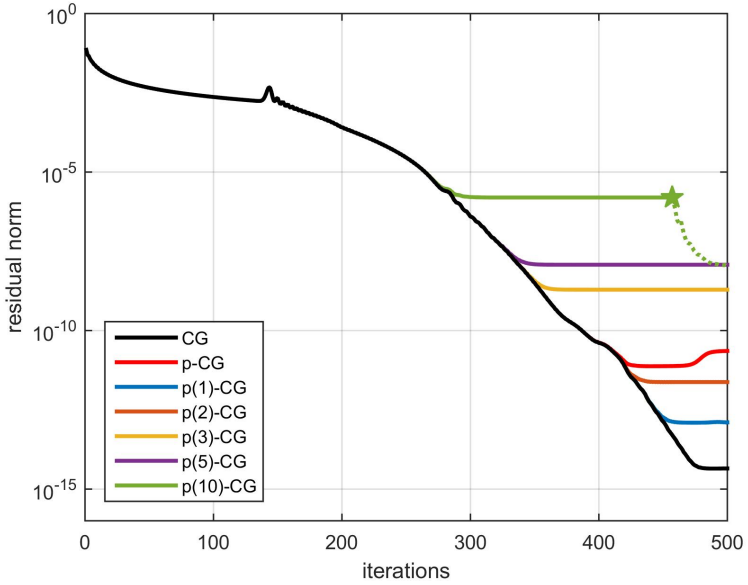
Overview of pipelined Krylov methods

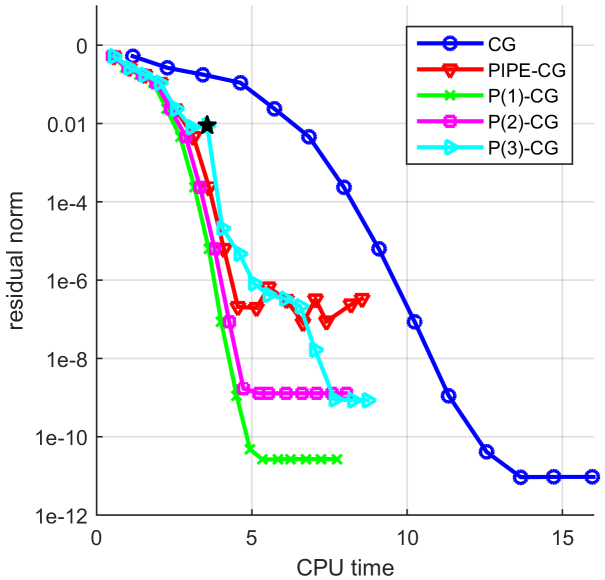
- Pipelined GMRES — Ghysels et al. (2013)
  - depth-one pipeline: p-GMRES
  - deep pipelines: p(ℓ)-GMRES: compute ℓ new Krylov subspace basis vectors (SpMV's) during global communication and orthogonalize after ℓ iterations.

$$V_{i-\ell+1} = [v_0, v_1, \ldots, v_{i-\ell}]$$
$$Z_{i+1} = [z_0, z_1, \ldots, z_{i-\ell}, \underbrace{z_{i-\ell+1}, \ldots, z_i}_{\ell}]$$

- Pipelined CG
  - depth-one pipeline: p-CG — Ghysels et al. (2014)
  - deep pipelines: p(ℓ)-CG — Cornelis et al. (2018)
- Pipelined BiCGStab — C. & Vanroose (2017)
- Pipelined Block-CG — (work in progress)
- Pipelined BiCG/CGLS/LSQR — (work in progress)
- Preconditioned pipelined variants
- Augmented/deflated/hybrid s-step pipelined methods — Yamazaki et al. (2017)

Numerical stability in finite precision

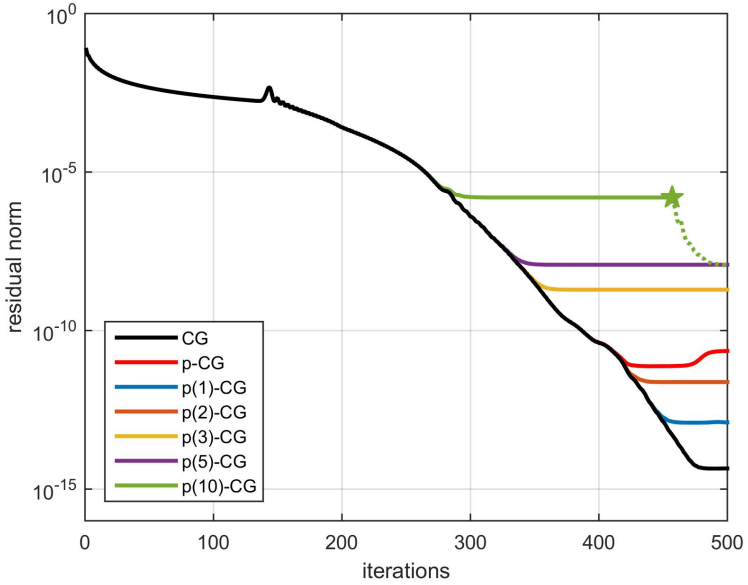


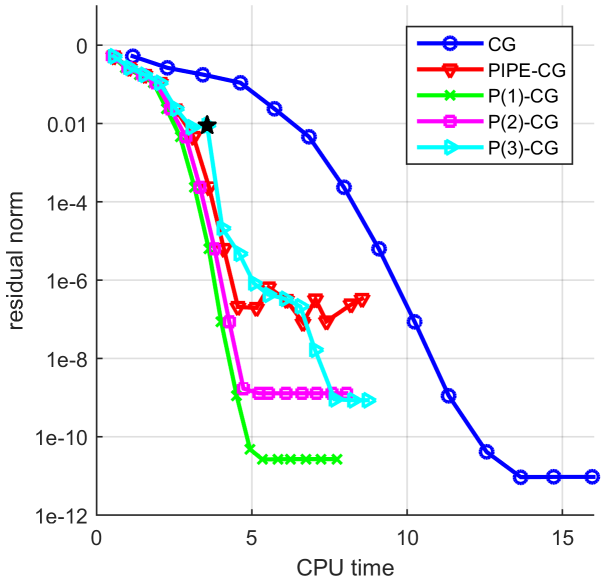


Pipelined KSM produce identical iterates to classic KSM in exact arithmetic, but finite precision computations introduce roundoff errors. This has two effects:

1. Delay of convergence, caused by loss of basis orthogonality
2. Loss of attainable accuracy, caused by local rounding errors in recurrences

Numerical stability in finite precision





Pipelined KSM produce identical iterates to classic KSM in exact arithmetic, but finite precision computations introduce roundoff errors. This has two effects:

1. Delay of convergence, caused by loss of basis orthogonality
2. Loss of attainable accuracy, caused by local rounding errors in recurrences

Severity of loss of attainable accuracy depends strongly on

- pipeline length l
- system size N
- choice of polynomial $P_l(A)$

Rounding error behavior in CG

Rounding errors due to recurrence relations for residual and auxiliary variables:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}.$$

Rounding error behavior in CG

Rounding errors due to recurrence relations for residual and auxiliary variables:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}.$$

Residual deviates from the true residual $b - A\bar{x}_{i+1}$ in finite precision:

$$\begin{aligned} f_{i+1} &= (b - A\bar{x}_{i+1}) - \bar{r}_{i+1} \\ &= b - A(\bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}) - (\bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}) \\ &= f_i - A\xi^x_{i+1} - \xi^r_{i+1} \\ &= f_0 - \sum_{k=0}^{i} (A\xi^x_{k+1} + \xi^r_{k+1}) \,. \end{aligned}$$

Sleijpen & van der Vorst (1995)
Greenbaum (1997)

Rounding error behavior in CG

Rounding errors due to recurrence relations for residual and auxiliary variables:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}.$$

Residual deviates from the true residual $b - A\bar{x}_{i+1}$ in finite precision:

$$\begin{aligned} f_{i+1} &= (b - A\bar{x}_{i+1}) - \bar{r}_{i+1} \\ &= b - A(\bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}) - (\bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}) \\ &= f_i - A\xi^x_{i+1} - \xi^r_{i+1} \\ &= f_0 - \sum_{k=0}^{i} (A\xi^x_{k+1} + \xi^r_{k+1}) \,. \end{aligned}$$

Sleijpen & van der Vorst (1995)
Greenbaum (1997)

Matrix notation: $\mathcal{F}_{i+1} = [f_0, \ldots, f_i]$, $\Theta^x_i = [0, \xi^x_1, \ldots, \xi^x_{i-1}]$, $\Theta^r_i = [f_0, \xi^r_1, \ldots, \xi^r_{i-1}]$

$$\mathcal{F}_{i+1} = -(A\Theta^x_{i+1} + \Theta^r_{i+1})\, U_{i+1},$$

with U_{i+1} an upper triangular matrix with all entries one.

Rounding error behavior in CG

Rounding errors due to recurrence relations for residual and auxiliary variables:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}.$$

Residual deviates from the true residual $b - A\bar{x}_{i+1}$ in finite precision:

$$\begin{aligned} f_{i+1} &= (b - A\bar{x}_{i+1}) - \bar{r}_{i+1} \\ &= b - A(\bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}) - (\bar{r}_i - \bar{\alpha}_i A \bar{p}_i + \xi^r_{i+1}) \\ &= f_i - A\xi^x_{i+1} - \xi^r_{i+1} \\ &= f_0 - \sum_{k=0}^{i} (A\xi^x_{k+1} + \xi^r_{k+1}) \,. \end{aligned}$$

Sleijpen & van der Vorst (1995)
Greenbaum (1997)

Matrix notation: $\mathcal{F}_{i+1} = [f_0, \ldots, f_i]$, $\Theta^x_i = [0, \xi^x_1, \ldots, \xi^x_{i-1}]$, $\Theta^r_i = [f_0, \xi^r_1, \ldots, \xi^r_{i-1}]$

$$\mathcal{F}_{i+1} = -(A\Theta^x_{i+1} + \Theta^r_{i+1})\, U_{i+1},$$

with U_{i+1} an upper triangular matrix with all entries one.

Only accumulation of local rounding errors in classical CG, no amplification.

Gutknecht & Strakos (2000)
van der Vorst & Ye (2000)

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Residual gap is coupled to the gaps on the other auxiliary variables:

$$f_j = (b - A\bar{x}_j) - \bar{r}_j = f_0 - \sum_{k=0}^{j-1} \bar{\alpha}_k g_k - \sum_{k=0}^{j-1} \left(A\xi^x_{k+1} + \xi^r_{k+1} \right),$$

$$g_j = A\bar{p}_j - \bar{s}_j = \left(\prod_{k=1}^{j} \bar{\beta}_k \right) g_0 + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) \left(A\xi^p_k - \xi^s_k \right) + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) h_k,$$

$$h_j = A\bar{r}_j - \bar{w}_j = h_0 - \sum_{k=0}^{j-1} \bar{\alpha}_k e_k + \sum_{k=0}^{j-1} \left(A\xi^r_{k+1} - \xi^w_{k+1} \right),$$

$$e_j = A\bar{s}_j - \bar{z}_j = \left(\prod_{k=1}^{j} \bar{\beta}_k \right) e_0 + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) \left(A\xi^s_k - \xi^z_k \right).$$

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Residual gap is coupled to the gaps on the other auxiliary variables:

$$f_j = (b - A\bar{x}_j) - \bar{r}_j = f_0 - \sum_{k=0}^{j-1} \bar{\alpha}_k g_k - \sum_{k=0}^{j-1} \left(A\xi^x_{k+1} + \xi^r_{k+1} \right),$$

$$g_j = A\bar{p}_j - \bar{s}_j = \left(\prod_{k=1}^{j} \bar{\beta}_k \right) g_0 + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) \left(A\xi^p_k - \xi^s_k \right) + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) h_k,$$

$$h_j = A\bar{r}_j - \bar{w}_j = h_0 - \sum_{k=0}^{j-1} \bar{\alpha}_k e_k + \sum_{k=0}^{j-1} \left(A\xi^r_{k+1} - \xi^w_{k+1} \right),$$

$$e_j = A\bar{s}_j - \bar{z}_j = \left(\prod_{k=1}^{j} \bar{\beta}_k \right) e_0 + \sum_{k=1}^{j} \left(\prod_{l=k+1}^{j} \bar{\beta}_l \right) \left(A\xi^s_k - \xi^z_k \right).$$

Amplification of local rounding errors possible, depending on $\bar{\alpha}_i$'s and $\bar{\beta}_i$'s.

Carson et al. (2017)
C. & Vanroose (2017)

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Matrix notation:

$$\mathcal{F}_{j+1} = (A\Theta^x_{j+1} + \Theta^r_{j+1})\, U_{j+1} - \mathcal{G}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{G}_{j+1} = (A\Theta^p_{j+1} + \Theta^s_{j+1})\, \mathcal{B}^{-1}_{j+1} + \mathcal{H}_{j+1}\widetilde{\mathcal{B}^{-1}_{j+1}}$$

$$\mathcal{H}_{j+1} = (A\Theta^u_{j+1} + \Theta^w_{j+1})\, U_{j+1} - \mathcal{E}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{E}_{j+1} = (A\Theta^q_{j+1} + \Theta^z_{j+1})\, \mathcal{B}^{-1}_{j+1}$$

Amplification of local rounding errors possible, depending on $\bar{\alpha}_i$'s and $\bar{\beta}_i$'s.

Carson et al. (2017)
C. & Vanroose (2017)

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Matrix notation:

$$\mathcal{F}_{j+1} = (A\Theta^x_{j+1} + \Theta^r_{j+1})\, U_{j+1} - \mathcal{G}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{G}_{j+1} = (A\Theta^p_{j+1} + \Theta^s_{j+1})\, \mathcal{B}^{-1}_{j+1} + \mathcal{H}_{j+1}\widetilde{\mathcal{B}^{-1}_{j+1}}$$

$$\mathcal{H}_{j+1} = (A\Theta^u_{j+1} + \Theta^w_{j+1})\, U_{j+1} - \mathcal{E}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{E}_{j+1} = (A\Theta^q_{j+1} + \Theta^z_{j+1})\, \mathcal{B}^{-1}_{j+1}$$

$$\begin{pmatrix} 0 & \bar{\alpha}_0 & \bar{\alpha}_0 & \cdots & \bar{\alpha}_0 \\ & 0 & \bar{\alpha}_1 & \cdots & \bar{\alpha}_1 \\ & & \ddots & & \vdots \\ & & & 0 & \bar{\alpha}_{j-1} \\ & & & & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & \bar{\beta}_1 & \bar{\beta}_1\bar{\beta}_2 & \cdots & \bar{\beta}_1\bar{\beta}_2\cdots\bar{\beta}_j \\ & 1 & \bar{\beta}_2 & & \bar{\beta}_2\cdots\bar{\beta}_j \\ & & \ddots & & \vdots \\ & & & 1 & \bar{\beta}_j \\ & & & & 1 \end{pmatrix}$$

Amplification of local rounding errors possible, depending on $\bar{\alpha}_i$'s and $\bar{\beta}_i$'s.

Carson et al. (2017)
C. & Vanroose (2017)

Rounding error behavior in pipelined CG

Additional recurrence relations in pipelined CG all introduce local rounding errors:

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\alpha}_i \bar{p}_i + \xi^x_{i+1}, \qquad \bar{s}_i = \bar{w}_i + \bar{\beta}_i \bar{s}_{i-1} + \xi^s_i,$$

$$\bar{r}_{i+1} = \bar{r}_i - \bar{\alpha}_i \bar{s}_i + \xi^r_{i+1}, \qquad \bar{w}_{i+1} = \bar{w}_i - \bar{\alpha}_i \bar{z}_i + \xi^w_{i+1},$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i \bar{p}_{i-1} + \xi^p_i, \qquad \bar{z}_i = A\bar{w}_i + \bar{\beta}_i \bar{z}_{i-1} + \xi^z_i,$$

Matrix notation:

$$\mathcal{F}_{j+1} = (A\Theta^x_{j+1} + \Theta^r_{j+1})\, U_{j+1} - \mathcal{G}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{G}_{j+1} = (A\Theta^p_{j+1} + \Theta^s_{j+1})\, \mathcal{B}^{-1}_{j+1} + \mathcal{H}_{j+1}\widetilde{\mathcal{B}^{-1}_{j+1}}$$

$$\mathcal{H}_{j+1} = (A\Theta^u_{j+1} + \Theta^w_{j+1})\, U_{j+1} - \mathcal{E}_{j+1}\mathcal{A}_{j+1}$$

$$\mathcal{E}_{j+1} = (A\Theta^q_{j+1} + \Theta^z_{j+1})\, \mathcal{B}^{-1}_{j+1}$$

$$\begin{pmatrix} 0 & \bar{\alpha}_0 & \bar{\alpha}_0 & \cdots & \bar{\alpha}_0 \\ & 0 & \bar{\alpha}_1 & \cdots & \bar{\alpha}_1 \\ & & \ddots & & \vdots \\ & & & 0 & \bar{\alpha}_{j-1} \\ & & & & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & \bar{\beta}_1 & \bar{\beta}_1\bar{\beta}_2 & \cdots & \bar{\beta}_1\bar{\beta}_2\cdots\bar{\beta}_j \\ & 1 & \bar{\beta}_2 & & \bar{\beta}_2\cdots\bar{\beta}_j \\ & & \ddots & & \vdots \\ & & & 1 & \bar{\beta}_j \\ & & & & 1 \end{pmatrix}$$

Note:

$\beta_i\, \beta_{i+1} \cdots \beta_j = \dfrac{\|r_j\|^2}{\|r_{i-1}\|^2} \Rightarrow$ arbitrarily large in CG!

Amplification of local rounding errors possible, depending on $\bar{\alpha}_i$'s and $\bar{\beta}_i$'s.

Carson et al. (2017)
C. & Vanroose (2017)

Rounding error behavior in p(l)-CG

True basis vector $\bar{\mathbf{v}}_{j+1}$ satisfies:

$$A\bar{V}_j = \bar{V}_{j+1}\bar{H}_{j+1,j} + (\bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1})\bar{\Delta}_{j+1,j}$$

Rounding error behavior in p(l)-CG

True basis vector $\bar{\mathbf{v}}_{j+1}$ satisfies:

$$A\bar{V}_j = \bar{V}_{j+1}\bar{H}_{j+1,j} + (\bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1})\bar{\Delta}_{j+1,j}$$

Computed basis vector $\bar{v}_{j+1}$ satisfies:

$$\bar{Z}_j = \bar{V}_j\bar{G}_j + \Theta_j^v$$

Rounding error behavior in p(l)-CG

True basis vector $\bar{\mathbf{v}}_{j+1}$ satisfies:

$$A\bar{V}_j = \bar{V}_{j+1}\bar{H}_{j+1,j} + (\bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1})\bar{\Delta}_{j+1,j}$$

Computed basis vector $\bar{v}_{j+1}$ satisfies:

$$\bar{Z}_j = \bar{V}_j\bar{G}_j + \Theta_j^v$$

Computed basis vector $\bar{z}_{j+1}$ satisfies:

$$A\bar{Z}_j = \bar{Z}_{j+1}\bar{B}_{j+1,j} + \Theta_j^z$$

Rounding error behavior in p(l)-CG

True basis vector $\bar{\mathbf{v}}_{j+1}$ satisfies:

$$A\bar{V}_j = \bar{V}_{j+1}\bar{H}_{j+1,j} + (\bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1})\bar{\Delta}_{j+1,j}$$

Computed basis vector $\bar{v}_{j+1}$ satisfies:

$$\bar{Z}_j = \bar{V}_j\bar{G}_j + \Theta_j^v$$

Computed basis vector $\bar{z}_{j+1}$ satisfies:

$$A\bar{Z}_j = \bar{Z}_{j+1}\bar{B}_{j+1,j} + \Theta_j^z$$

Thus the basis vector gap $\mathcal{F}_{j+1} = \bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1}$ can be calculated as

$$\mathcal{F}_{j+1} = (\Theta_j^{\bar{z}}\bar{G}_j^{-1} - A\Theta_j^{\bar{v}}\bar{G}_j^{-1} + \Theta_{j+1}^{\bar{v}}\bar{B}_{j+1,j}\bar{G}_j^{-1})\bar{\Delta}_{j+1,j}^{-1}$$

Rounding error behavior in p(l)-CG

True basis vector $\bar{\mathbf{v}}_{j+1}$ satisfies:

$$A\bar{V}_j = \bar{V}_{j+1}\bar{H}_{j+1,j} + (\bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1})\bar{\Delta}_{j+1,j}$$

Computed basis vector $\bar{v}_{j+1}$ satisfies:

$$\bar{Z}_j = \bar{V}_j\bar{G}_j + \Theta_j^v$$

Computed basis vector $\bar{z}_{j+1}$ satisfies:

$$A\bar{Z}_j = \bar{Z}_{j+1}\bar{B}_{j+1,j} + \Theta_j^z$$

Thus the basis vector gap $\mathcal{F}_{j+1} = \bar{\boldsymbol{V}}_{j+1} - \bar{V}_{j+1}$ can be calculated as

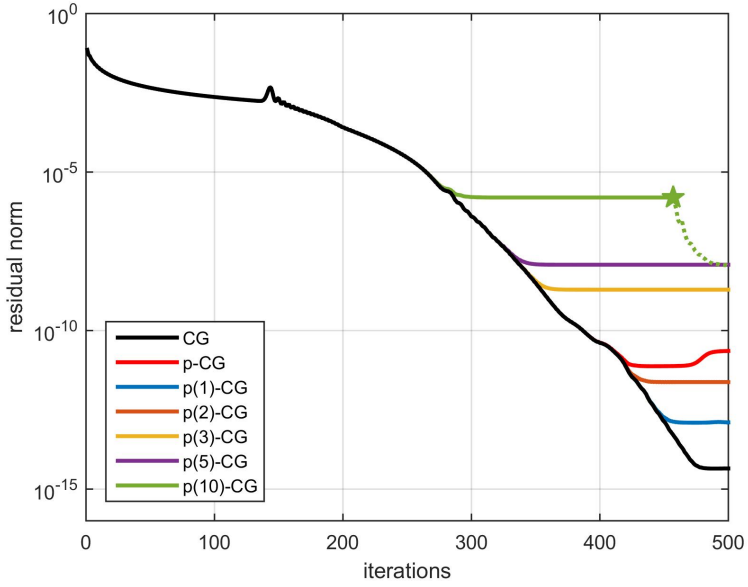
$$\mathcal{F}_{j+1} = (\Theta_j^{\bar{v}}\bar{G}_j^{-1} - A\Theta_j^{\bar{v}}\bar{G}_j^{-1} + \Theta_{j+1}^{\bar{v}}\bar{B}_{j+1,j}\bar{G}_j^{-1})\bar{\Delta}_{j+1,j}^{-1}$$

Amplification of local rounding errors possible, depending on $\bar{G}_j^{-1}$.

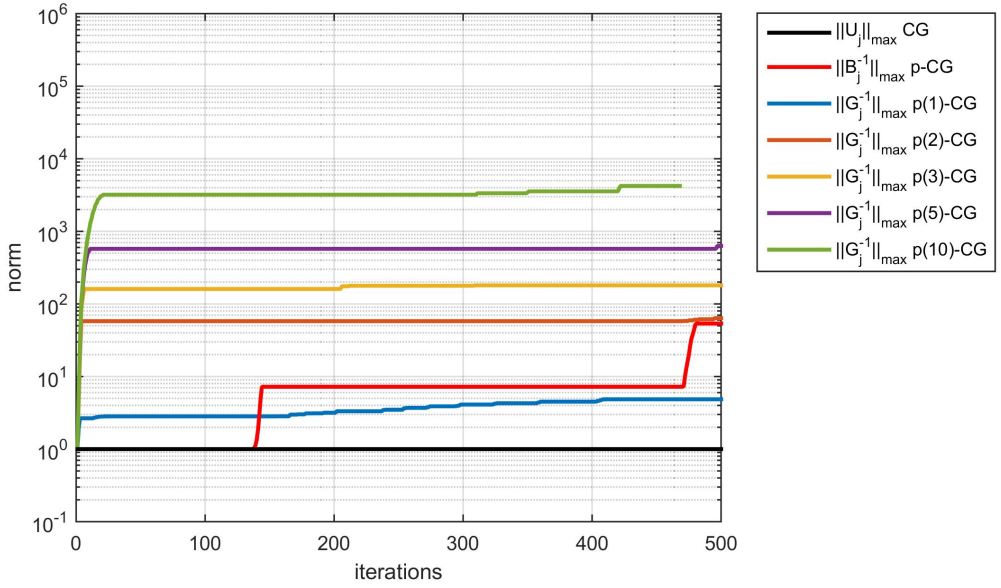
Carson & Demmel (2014)
C. (2018)

Rounding error behavior in p(l)-CG

residual history $\|b - A\bar{x}_i\|_2$



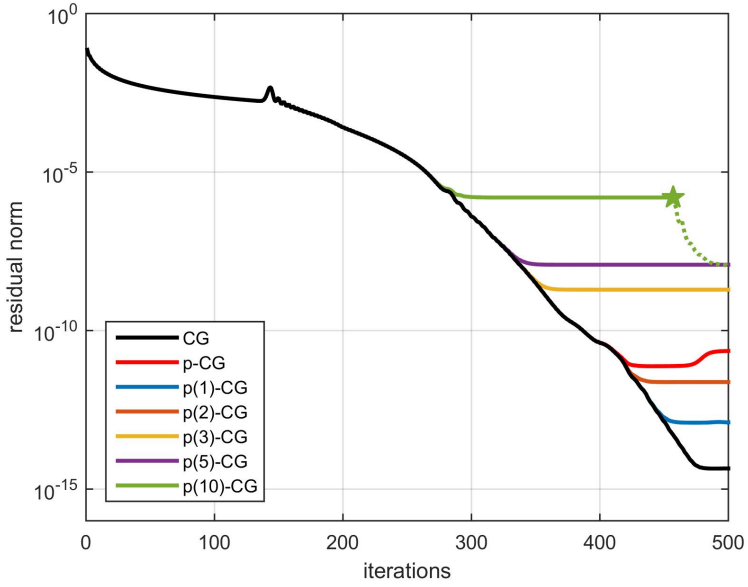
$\|G_i^{-1}\|_{\max}$ for different l



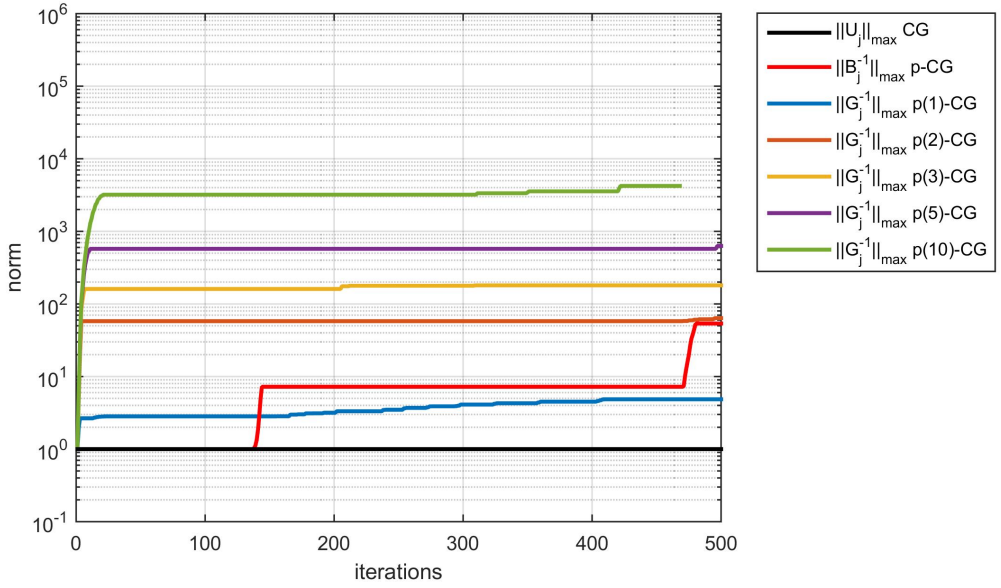
Maximum norm $\|G_i^{-1}\|_{\max}$ provides a measure for the impact of local rounding error propagation on maximal attainable accuracy in p(l)-CG

Rounding error behavior in p(l)-CG

residual history $\|b - A\bar{x}_i\|_2$



$\|G_i^{-1}\|_{\max}$ for different l



Maximum norm $\|G_i^{-1}\|_{\max}$ provides a measure for the impact of local rounding error propagation on maximal attainable accuracy in p(l)-CG

Rounding error analysis explains loss of attainable accuracy with respect to

- pipeline length l
- system size N
- choice of polynomial $P_l(A)$

Residual replacement in pipelined CG

- Idea: improve accuracy by replacing $\bar{r}_i$, $\bar{s}_i$, $\bar{w}_i$ and $\bar{z}_i$ by their true values in selected iterations:

$$\bar{r}_{i+1} = \mathrm{fl}(b - A\bar{x}_{i+1}), \quad \bar{w}_{i+1} = \mathrm{fl}(A\bar{r}_{i+1}), \quad \bar{s}_i = \mathrm{fl}(A\bar{p}_i), \quad \bar{z}_i = \mathrm{fl}(A\bar{s}_i).$$

van der Vorst & Ye (2000)

Residual replacement in pipelined CG

- Idea: improve accuracy by replacing $\bar{r}_i$, $\bar{s}_i$, $\bar{w}_i$ and $\bar{z}_i$ by their true values in selected iterations:

$$\bar{r}_{i+1} = \mathrm{fl}(b - A\bar{x}_{i+1}), \quad \bar{w}_{i+1} = \mathrm{fl}(A\bar{r}_{i+1}), \quad \bar{s}_i = \mathrm{fl}(A\bar{p}_i), \quad \bar{z}_i = \mathrm{fl}(A\bar{s}_i).$$

van der Vorst & Ye (2000)

- Choose when to replace based on estimate of $\|f_i\| = \|(b - A\bar{x}_i) - \bar{r}_i\|$; replacement criterion:

$$\|f_{i-1}\| \leq \tau\|\bar{r}_{i-1}\| \quad \text{and} \quad \|f_i\| > \tau\|\bar{r}_i\| \qquad \text{with } \tau = \sqrt{\epsilon}.$$

  - Replace sufficiently often such that $\|f_i\|$ remains small
  - Don't replace too often to limit additional computation cost of SpMV's
  - Don't replace when $\|\bar{r}_i\|$ is too small, which may cause delay of convergence

Sleijpen & van der Vorst (1996)

Strakos & Tichy (2002)

Residual replacement in pipelined CG

- Idea: improve accuracy by replacing $\bar{r}_i$, $\bar{s}_i$, $\bar{w}_i$ and $\bar{z}_i$ by their true values in selected iterations:

$$\bar{r}_{i+1} = \mathrm{fl}(b - A\bar{x}_{i+1}), \quad \bar{w}_{i+1} = \mathrm{fl}(A\bar{r}_{i+1}), \quad \bar{s}_i = \mathrm{fl}(A\bar{p}_i), \quad \bar{z}_i = \mathrm{fl}(A\bar{s}_i).$$

  van der Vorst & Ye (2000)

- Choose when to replace based on estimate of $\|f_i\| = \|(b - A\bar{x}_i) - \bar{r}_i\|$; replacement criterion:

$$\|f_{i-1}\| \leq \tau \|\bar{r}_{i-1}\| \quad \text{and} \quad \|f_i\| > \tau \|\bar{r}_i\| \qquad \text{with } \tau = \sqrt{\epsilon}.$$

  - Replace sufficiently often such that $\|f_i\|$ remains small
  - Don't replace too often to limit additional computation cost of SpMV's
  - Don't replace when $\|\bar{r}_i\|$ is too small, which may cause delay of convergence

  Sleijpen & van der Vorst (1996)

  Strakos & Tichy (2002)

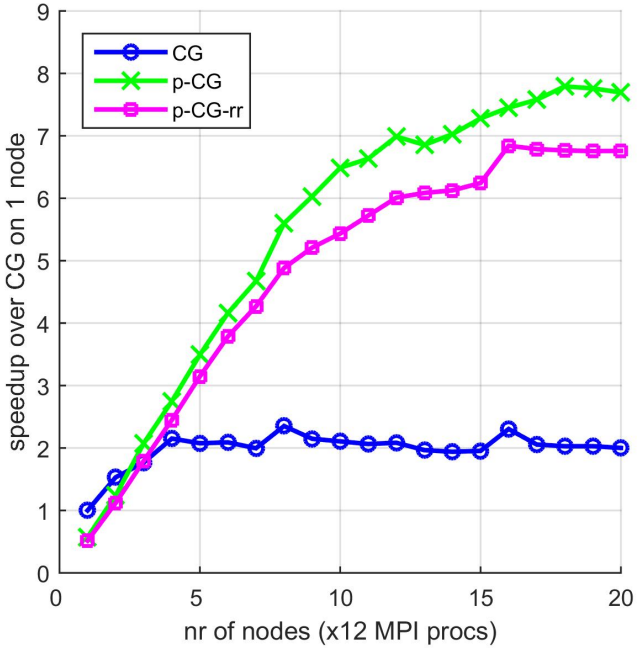
- Estimate of $\|f_i\|$ is computed at runtime (inexpensive). Accuracy is improved to comparable level as classical CG method in many cases.

  C. & Vanroose (2017)

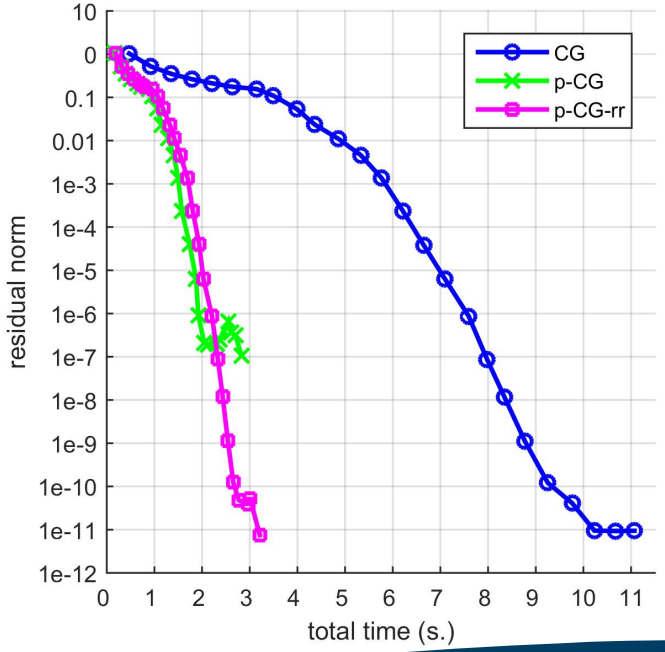
Residual replacement in pipelined CG

- PETSc implementation using MPICH-3.1.3 communication
- Benchmark problem: 2D Laplacian model, 1,000,000 unknowns
- System specs: 20 nodes, two 6-core Intel Xeon X5660 Nehalem 2.8GHz CPUs/node

Speedup over single-node CG (12-240 cores)



Accuracy i.f.o. total time spent (240 cores)

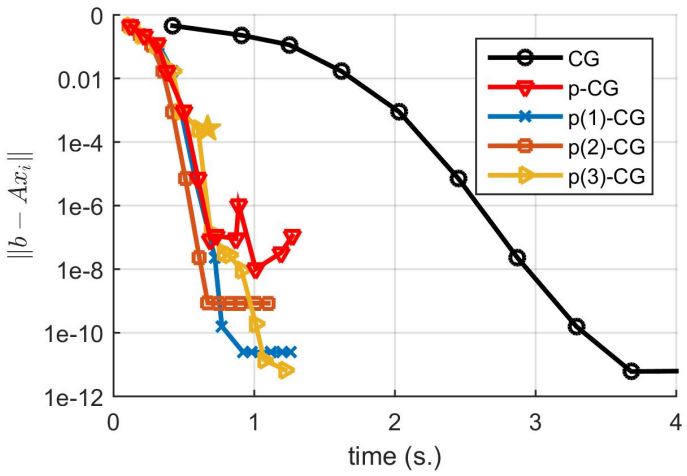


Numerically stable variant of p(l)-CG

Use idea similar to residual replacement to improve stability: replace recurrence for v_{j+1} by the Arnoldi relation:

Original p(l)-CG:

$$v_{j+1} = \left(z_{j+1} - \sum_{k=j-2l+1}^{j} g_{k,j+1} v_k \right) / g_{j+1,j+1}$$



Numerically stable variant of p(l)-CG

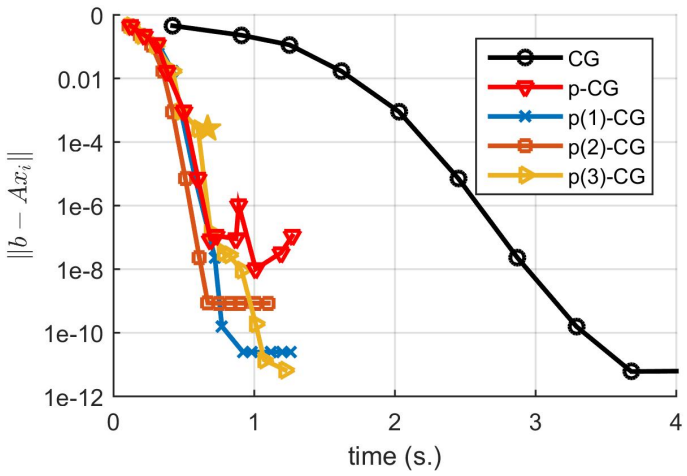
Use idea similar to residual replacement to improve stability: replace recurrence for v_{j+1} by the Arnoldi relation:

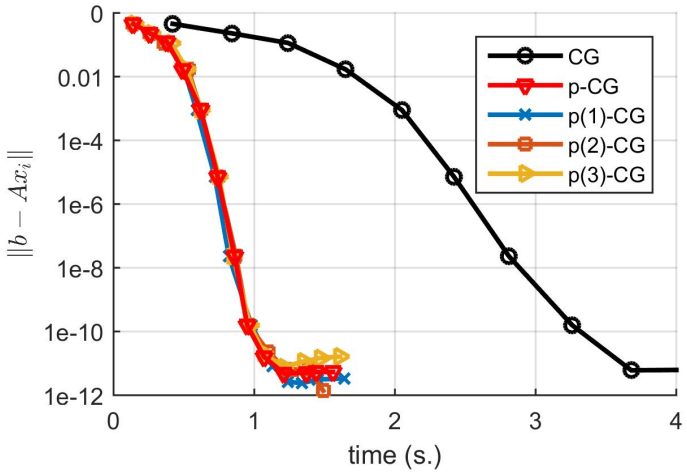
Original p(l)-CG:

$$v_{j+1} = \left(z_{j+1} - \sum_{k=j-2l+1}^{j} g_{k,j+1} v_k \right) / g_{j+1,j+1}$$

Stabilized p(l)-CG:

$$v_{j+1} = (A v_j - \gamma_j v_j - \delta_{j-1} v_{j-1}) / \delta_j$$





Numerically stable variant of p(l)-CG

Use idea similar to residual replacement to improve stability: replace recurrence for v_{j+1} by the Arnoldi relation:

Original p(l)-CG:

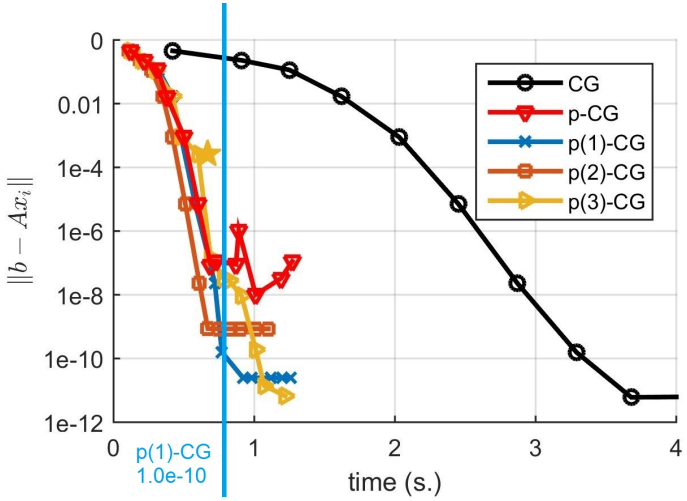
$$v_{j+1} = \left(z_{j+1} - \sum_{k=j-2l+1}^{j} g_{k,j+1} v_k \right) / g_{j+1,j+1}$$

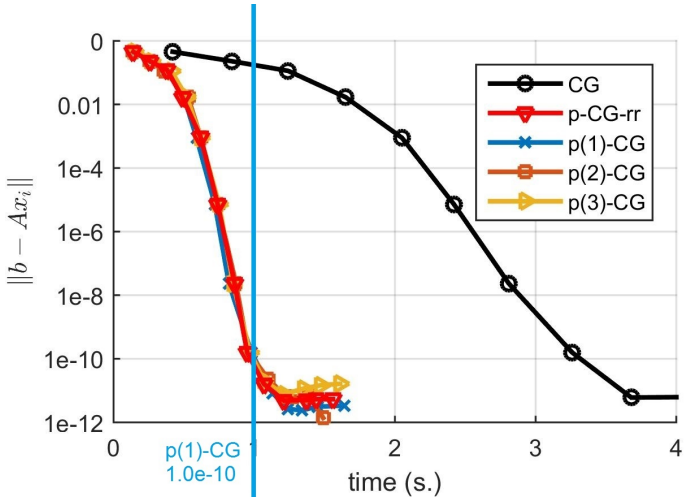
Stabilized p(l)-CG:

$$v_{j+1} = (Av_j - \gamma_j v_j - \delta_{j-1} v_{j-1}) / \delta_j$$

Attainable accuracy is improved for all pipeline lengths l, but extra SpMV increases computation time

$\Rightarrow$ trade-off between numerical stability and performance





Wrap-up of this talk

- Pipelined Krylov subspace methods are a promising approach to reduce synchronization cost in linear solvers for large-scale problems
  - By adding *auxiliary variables* and recurrences it is possible to *hide* communication latency behind computational kernels
  - *Deep pipelines* allow to hide global reductions behind multiple SpMV's
  - *Asynchronous implementation*: dot-products can take multiple iterations to complete, in an overlapping manner
  - *Improved scaling* over classic KSMs in strong scaling limit, where global reduction latencies rise and volume of computations per core diminishes

Wrap-up of this talk

- Pipelined Krylov subspace methods are a promising approach to reduce synchronization cost in linear solvers for large-scale problems
  - By adding *auxiliary variables* and recurrences it is possible to *hide* communication latency behind computational kernels
  - *Deep pipelines* allow to hide global reductions behind multiple SpMV's
  - *Asynchronous implementation*: dot-products can take multiple iterations to complete, in an overlapping manner
  - *Improved scaling* over classic KSMs in strong scaling limit, where global reduction latencies rise and volume of computations per core diminishes

- Finite precision behavior of communication reducing and hiding algorithms should be carefully monitored!
  - *Rounding error analysis* allows to explain observed loss of attainable accuracy
  - *Residual replacement* -type techniques can be applied to improve numerical stability, but at a (slight) increase in computational cost

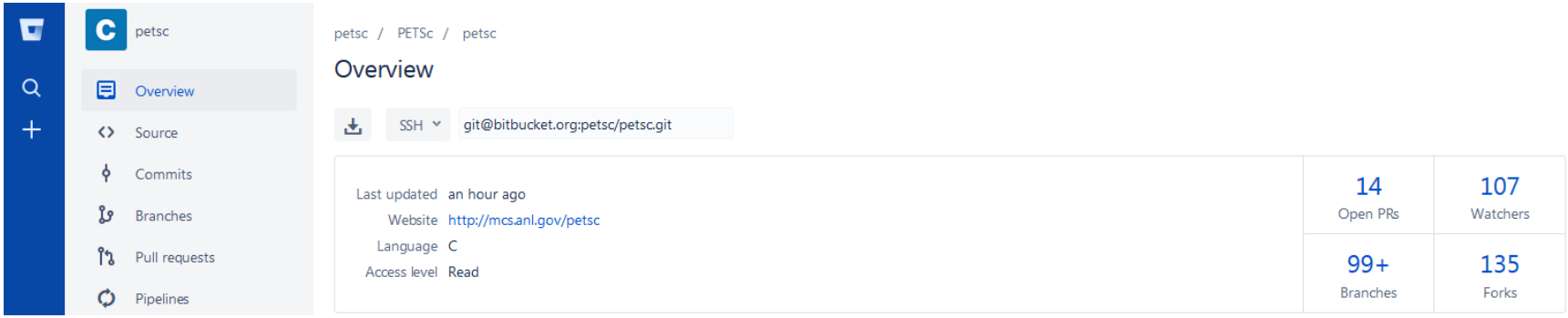
Related work

- Work in progress (not covered in this talk):
  - Impact of *hard faults* & *soft-errors* on pipelined Krylov subspace methods (INRIA Bordeaux)
  - Resilience of pipelined Krylov subspace methods to *system noise* (UChicago/ETHZ/Rice/UIllinois)
  - *Pipelined Blocked* Krylov subspace methods for systems with multiple rhs (UAntwerp/INRIA Bordeaux)
  - *Pipelined Lanczos* for eigenvalue calculation in graph partitioning algorithms (LBNL)

- Many interesting open problems and challenges remain as we push toward exascale-level computing!

Contributions to PETSc

Open source HPC linear algebra toolkit: `https://www.mcs.anl.gov/petsc/`



- KSPPGMRES: pipelined GMRES (thanks to J. Brown)
- KSPPIPECG: pipelined CG
- KSPPIPECGRR: pipelined CG with automated residual replacement
- KSPPIPELCG: pipelined CG with deep pipelines
- KPPPIPECR: pipelined conjugate residuals
- KSPGROPPCG: asynchronous CG variant by W. Gropp and collaborators
- KSPPIPEBCGS: pipelined BiCGStab

We are soliciting for feedback from your applications!

Our main publications

- P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose, *Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines* SIAM J. Sci. Comput., 35(1), pp. C48C71, 2013.
- P. Ghysels and W. Vanroose, *Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm*, Parallel Computing, 40(7), pp. 224-238, 2014.
- S. Cools, E.F. Yetkin, E. Agullo, L. Giraud, W. Vanroose, *Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined Conjugate Gradient method.* SIAM J. on Matrix Anal. Appl., 39(1), pp. 426-450, 2018.
- S. Cools, W. Vanroose, *The communication-hiding pipelined BiCGStab method for the parallel solution of large unsymmetric linear systems.* Parallel Computing, 65, pp. 1-20, Elsevier, 2017.
- J. Cornelis, S. Cools, W. Vanroose, *The communication-hiding Conjugate Gradient method with deep pipelines.* Submitted to SIAM J. Sci. Comput., 2018, Preprint available at arXiv:1801.04728.

Thank you!

siegfried.cools@uantwerp.be

https://www.uantwerpen.be/en/staff/siegfried-cools