

Communication Hiding Through Pipelining in Krylov Methods

Wim Vanroose

Department of Mathematics and Computer Science, U. Antwerpen, Belgium

Pieter Ghysels, Siegfried Cools, Jeffrey Cornelis

Funding: IWT flanders and Intel, Exa2ct Horizon 2020 (EU funding ref.no. 610741), FWO flanders,

U. Antwerpen Research Fund.

SIAM PP 18, Tokyo

Krylov subspace methods

We search for a solution x of $Ax = b$, $Ax = \lambda x$ or $A^T Ax = A^T b$ in the Krylov subspace

$$x \in x^{(0)} + \text{span} \left\{ r^{(0)}, Ar^{(0)}, A^2 r^{(0)}, \dots, A^{k-1} r^{(0)} \right\}, \quad (1)$$

where $r^{(0)} = b - Ax^{(0)}$.

- ▶ Conjugate Gradients,
- ▶ Lanczos,
- ▶ GMRES, Minres,
- ▶ BiCG, CGS, BiCGstab
- ▶ CGLS, bidiagonalization,
- ▶ ...
- ▶ Algebraic and Geometric Multigrid,
- ▶ Incomplete factorization,
- ▶ Domain Decomposition, FETI, BDDC,
- ▶ Physics based preconditioners,
- ▶ ...

GMRES, classical Gram-Schmidt

```
1:  $r_0 \leftarrow b - Ax_0$ 
2:  $v_0 \leftarrow r_0 / \|r_0\|_2$ 
3: for  $i = 0, \dots, m - 1$  do
4:    $w \leftarrow Av_i$ 
5:   for  $j = 0, \dots, i$  do
6:      $h_{j,i} \leftarrow \langle w, v_j \rangle$ 
7:   end for
8:    $\tilde{v}_{i+1} \leftarrow w - \sum_{j=1}^i h_{j,i} v_j$ 
9:    $h_{i+1,i} \leftarrow \|\tilde{v}_{i+1}\|_2$ 
10:   $v_{i+1} \leftarrow \tilde{v}_{i+1} / h_{i+1,i}$ 
11:  { apply Givens rotations to  $h_{:,i}$  }
12: end for
13:  $y_m \leftarrow$   

    $\operatorname{argmin} \|H_{m+1,m} y_m - \|r_0\|_2 e_1\|_2$ 
14:  $x \leftarrow x_0 + V_m y_m$ 
```

Sparse Matrix-Vector product

- ▶ Only communication with neighbors
- ▶ Good scaling

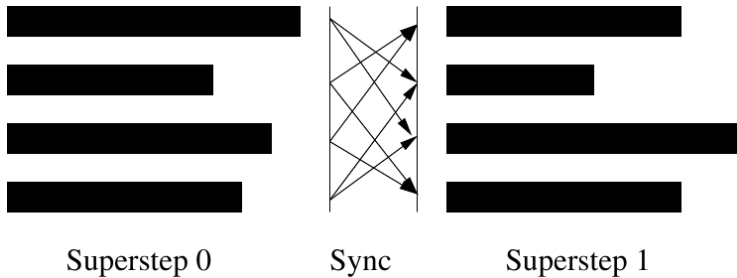
Dot-product

- ▶ Global communication
- ▶ Scales as $\log(P)$

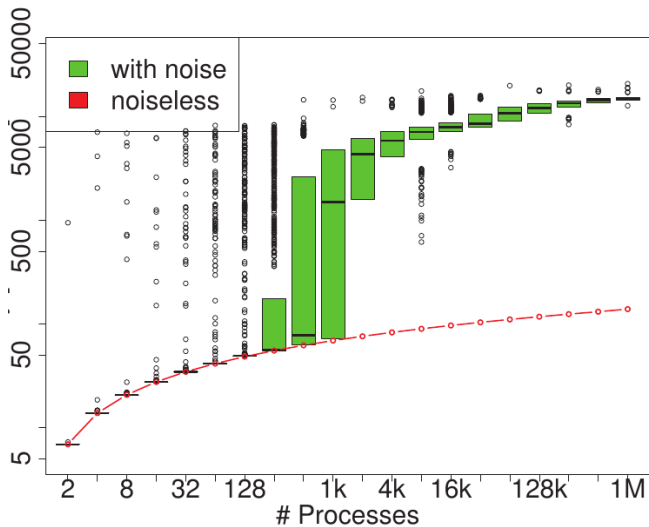
Scalar vector multiplication, vector-vector addition

- ▶ No communication

Bulk Synchronous Parallelism



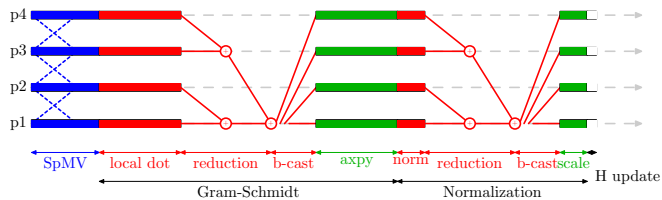
latency of global reductions



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

GMRES vs Pipelined GMRES iteration on 4 nodes

Classical GMRES



Communication Hiding and Communication Avoiding

Pipelined Conjugate Gradients

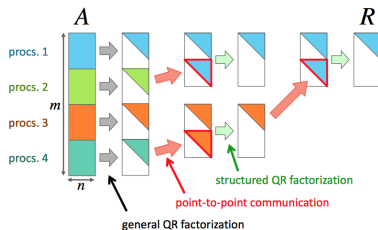
Propagation of Rounding Errors

Deeper Pipelines




Exploiting the properties of Symmetric Matrices

Implementation

Communication Hiding vs Communication Avoiding



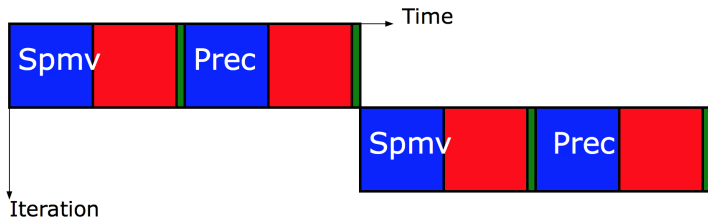
(figure taken from Fukaya et al, 2014)

-  M. Mohiyuddin, M. Hoemmen, J. Demmel and K. Yelick, *Minimizing communication in sparse matrix solvers*, p1–12, 2009.
-  M. Hoemmen, *Communication-avoiding Krylov subspace methods*, PhD Thesis, 2010.
-  J. Demmel, L. Grigori, M. Hoemmen and J. Langou *Communication-optimal parallel and sequential QR and LU factorizations* SISC, 34, pA206–A239, 2012.

Conjugate Gradients?

Preconditioned Conjugate Gradient

- 1: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $p_0 := u_0$
- 2: **for** $i = 0, \dots, m - 1$ **do**
- 3: $s := Ap_i$
- 4: $\alpha := \langle r_i, u_i \rangle / \langle s, p_i \rangle$
- 5: $x_{i+1} := x_i + \alpha p_i$
- 6: $r_{i+1} := r_i - \alpha s$
- 7: $u_{i+1} := M^{-1}r_{i+1}$
- 8: $\beta := \langle r_{i+1}, u_{i+1} \rangle / \langle r_i, u_i \rangle$
- 9: $p_{i+1} := u_{i+1} + \beta p_i$
- 10: **end for**



Auxiliary Recurrences

We already have introduced the notation

$$s_i := Ap_i \quad u_i := M^{-1}r_i \quad (2)$$

The recurrence of the search directions

$$p_i = u_i + \beta p_{i-1} \quad (3)$$

$$\times A \quad Ap_i = Au_i + \beta Ap_{i-1} \quad (4)$$

$$s_i = w_i + \beta s_{i-1} \quad (5)$$

where

$$w_i := Au_i \quad (6)$$

Only one global reduction each iteration.

- 1: $r_0 := b - Ax_0; u_0 := M^{-1}r_0; w_0 := Au_0$
- 2: $\alpha_0 := \langle r_0, u_0 \rangle / \langle w_0, u_0 \rangle; \beta := 0; \gamma_0 := \langle r_0, u_0 \rangle$
- 3: **for** $i = 0, \dots, m - 1$ **do**
- 4: $p_i := u_i + \beta_i p_{i-1}$
- 5: $s_i := w_i + \beta_i s_{i-1}$
- 6: $x_{i+1} := x_i + \alpha p_i$
- 7: $r_{i+1} := r_i - \alpha s_i$
- 8: $u_{i+1} := M^{-1}r_{i+1}$
- 9: $w_{i+1} := Au_{i+1}$
- 10: $\gamma_{i+1} := \langle r_{i+1}, u_{i+1} \rangle$
- 11: $\delta := \langle w_{i+1}, u_{i+1} \rangle$
- 12: $\beta_{i+1} := \gamma_{i+1} / \gamma_i$
- 13: $\alpha_{i+1} := \gamma_{i+1} / (\delta - \beta_{i+1} \gamma_{i+1} / \alpha_i)$
- 14: **end for**

Global reduction overlaps with sparse matrix-vector (SpMV) product.

```

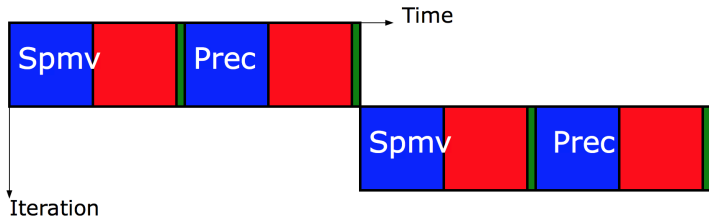
1:  $r_0 := b - Ax_0$ ;  $w_0 := Au_0$ 
2: for  $i = 0, \dots, m - 1$  do
3:    $\gamma_i := \langle r_i, r_i \rangle$ 
4:    $\delta := \langle w_i, r_i \rangle$ 
5:    $q_i := Aw_i$ 
6:   if  $i > 0$  then
7:      $\beta_i := \gamma_i / \gamma_{i-1}$ ;  $\alpha_i := \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$ 
8:   else
9:      $\beta_i := 0$ ;  $\alpha_i := \gamma_i / \delta$ 
10:  end if
11:   $z_i := q_i + \beta_i z_{i-1}$ 
12:   $s_i := w_i + \beta_i s_{i-1}$ 
13:   $p_i := r_i + \beta_i p_{i-1}$ 
14:   $x_{i+1} := x_i + \alpha_i p_i$ 
15:   $r_{i+1} := r_i - \alpha_i s_i$ 
16:   $w_{i+1} := w_i - \alpha_i z_i$ 
17: end for

```

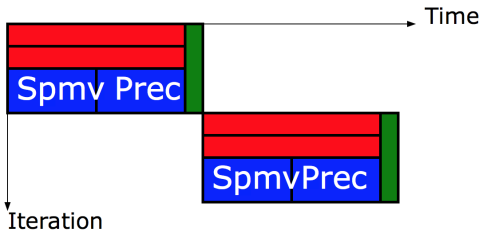
Preconditioned pipelined CG

```
1:  $r_0 := b - Ax_0$ ;  $u_0 := M^{-1}r_0$ ;  $w_0 := Au_0$ 
2: for  $i = 0, \dots$  do
3:    $\gamma_i := \langle r_i, u_i \rangle$ 
4:    $\delta := \langle w_i, u_i \rangle$ 
5:    $m_i := M^{-1}w_i$ 
6:    $n_i := Am_i$ 
7:   if  $i > 0$  then
8:      $\beta_i := \gamma_i/\gamma_{i-1}$ ;  $\alpha_i := \gamma_i/(\delta - \beta_i\gamma_i/\alpha_{i-1})$ 
9:   else
10:     $\beta_i := 0$ ;  $\alpha_i := \gamma_i/\delta$ 
11:   end if
12:    $z_i := n_i + \beta_i z_{i-1}$ 
13:    $q_i := m_i + \beta_i q_{i-1}$ 
14:    $s_i := w_i + \beta_i s_{i-1}$ 
15:    $p_i := u_i + \beta_i p_{i-1}$ 
16:    $x_{i+1} := x_i + \alpha_i p_i$ 
17:    $r_{i+1} := r_i - \alpha_i s_i$ 
18:    $u_{i+1} := u_i - \alpha_i q_i$ 
19:    $w_{i+1} := w_i - \alpha_i z_i$ 
20: end for
```

Conjugate Gradients



Pipelined Conjugate Gradients



Preconditioned pipelined CR

```
1:  $r_0 := b - Ax_0$ ;  $u_0 := M^{-1}r_0$ ;  $w_0 := Au_0$ 
2: for  $i = 0, \dots$  do
3:    $m_i := M^{-1}w_i$ 
4:    $\gamma_i := \langle w_i, u_i \rangle$ 
5:    $\delta := \langle m_i, w_i \rangle$ 
6:    $n_i := Am_i$ 
7:   if  $i > 0$  then
8:      $\beta_i := \gamma_i / \gamma_{i-1}$ ;  $\alpha_i := \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$ 
9:   else
10:     $\beta_i := 0$ ;  $\alpha_i := \gamma_i / \delta$ 
11:   end if
12:    $z_i := n_i + \beta_i z_{i-1}$ 
13:    $q_i := m_i + \beta_i q_{i-1}$ 
14:    $p_i := u_i + \beta_i p_{i-1}$ 
15:    $x_{i+1} := x_i + \alpha_i p_i$ 
16:    $u_{i+1} := u_i - \alpha_i q_i$ 
17:    $w_{i+1} := w_i - \alpha_i z_i$ 
18: end for
```

Cost Model

- ▶ G := time for a global reduction
- ▶ SpMV := time for a sparse-matrix vector product
- ▶ PC := time for preconditioner application
- ▶ local work such as AXPY is neglected

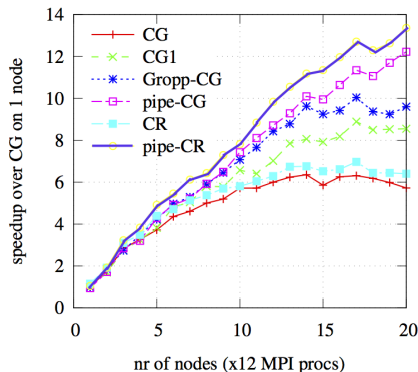
	flops	time (excl, AXPYs, DOTs)	#glob syncs	memory
CG	10	$2G + \text{SpMV} + \text{PC}$	2	4
Chro/Gea	12	$G + \text{SpMV} + \text{PC}$	1	5
CR	12	$2G + \text{SpMV} + \text{PC}$	2	5
pipe-CG	20	$\max(G, \text{SpMV} + \text{PC})$	1	9
pipe-CR	16	$\max(G, \text{SpMV}) + \text{PC}$	1	7
Gropp-CG	14	$\max(G, \text{SpMV}) + \max(G, \text{PC})$	2	6



P. Ghysels and W. Vanroose, *Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm*, *Parallel Computing*, **40**, (2014), Pages 224–238

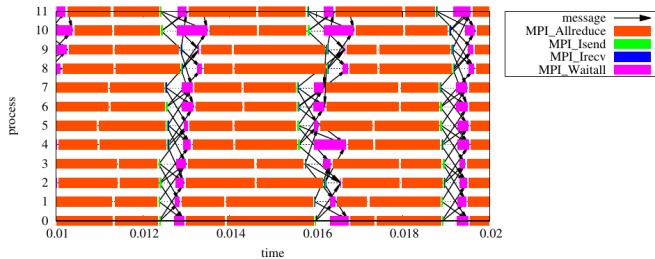
Better Strong Scaling

- ▶ Hydrostatic ice sheet flow, $100 \times 100 \times 50$ Q1 finite elements
- ▶ line search Newton method ($\text{rtol} = 10^{-8}$, $\text{atol} = 10^{-15}$)
- ▶ CG with block Jacobi ICC(0) precondition ($\text{rtol} = 10^{-5}$, $\text{atol} = 10^{-50}$)

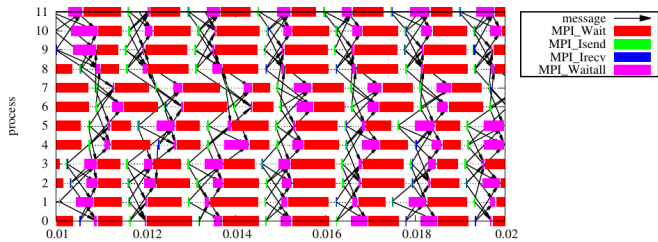


Measured speedup over standard CG for different variations of pipelined CG.

Conjugate Gradients for 2D 5-point stencil



Pipelined Conjugate Gradients

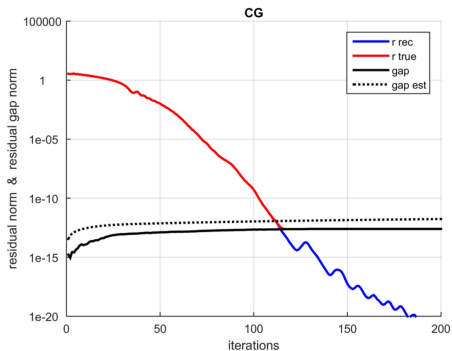


Pipelined methods in PETSc (from 3.4.2)

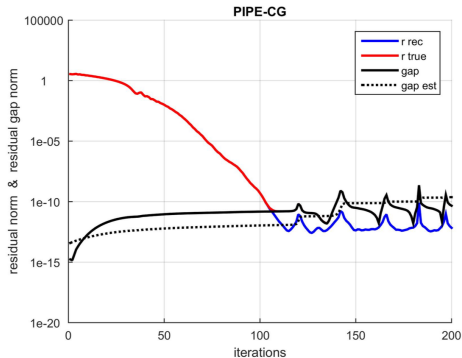
- ▶ Krylov methods: KSPPIPECR, KSPPIPECG, KSPGROPPCG, KSPPGMRES
- ▶ Uses MPI-3 non-blocking collectives
- ▶ export MPICH_ASYNC_PROGRESS=1

```
1: ...
2: KSP_PCAppl(ksp,W,M); /* m ← Bw */
3: if (i > 0 && ksp->normtype == KSP_NORM_PRECONDITIONED)
4:     VecNormBegin(U,NORM_2,&dp);
5: VecDotBegin(W,U,&gamma);
6: VecDotBegin(M,W,&delta);
7: PetscCommSplitReductionBegin(PetscObjectComm((PetscObject)U));
8: KSP_MatMult(ksp,Amat,M,N); /* n ← Am */
9: if (i > 0 && ksp->normtype == KSP_NORM_PRECONDITIONED)
10:    VecNormEnd(U,NORM_2,&dp);
11: VecDotEnd(W,U,&gamma);
12: VecDotEnd(M,W,&delta);
13: ...
```

True Residual vs recused Residual



CG



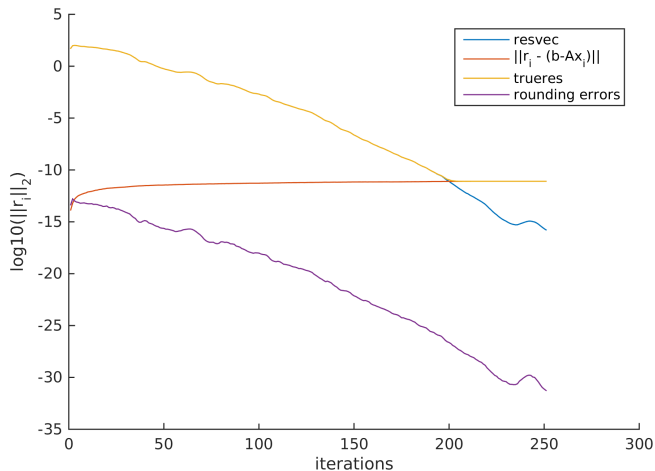
pipelined Chron/Gear CG

$$\begin{cases} \bar{x}_{i+1} = \bar{x}_i + \alpha_{i+1}\bar{p}_i + \delta_i^x \\ \bar{r}_{i+1} = \bar{r}_i - \alpha_{i+1}\bar{s}_i + \delta_i^r \end{cases} \quad (7)$$

where $s_i := Ap_i$ and $\bar{x}_i, \bar{r}_i, \bar{s}_i$ denote the computed iterates and δ_i^x and δ_i^r are the local rounding errors.

$$\begin{aligned} \Delta_{i+1}^r &= b - A\bar{x}_{i+1} - \bar{r}_{i+1} \\ &= b - A(\bar{x}_i + \alpha_{i+1}\bar{p}_i + \delta_i^x) - \bar{r}_i + \alpha_{i+1}\bar{s}_i - \delta_i^r \\ &= \underbrace{b - A\bar{x}_i - \bar{r}_i}_{\Delta_i^r} + \underbrace{A\delta_i^x + \delta_i^r}_{\text{local rounding error}} \end{aligned}$$

Rounding Errors



Global reduction overlaps with matrix vector product.

```

1:  $r_0 := b - Ax_0$ ;  $w_0 := Au_0$ 
2: for  $i = 0, \dots, m - 1$  do
3:    $\gamma_i := \langle r_i, r_i \rangle$ 
4:    $\delta := \langle w_i, r_i \rangle$ 
5:    $q_i := Aw_i$ 
6:   if  $i > 0$  then
7:      $\beta_i := \gamma_i / \gamma_{i-1}$ ;  $\alpha_i := \gamma_i / (\delta - \beta_i \gamma_i / \alpha_{i-1})$ 
8:   else
9:      $\beta_i := 0$ ;  $\alpha_i := \gamma_i / \delta$ 
10:  end if
11:   $z_i := q_i + \beta_i z_{i-1}$ 
12:   $s_i := w_i + \beta_i s_{i-1}$ 
13:   $p_i := r_i + \beta_i p_{i-1}$ 
14:   $x_{i+1} := x_i + \alpha_i p_i$ 
15:   $r_{i+1} := r_i - \alpha_i s_i$ 
16:   $w_{i+1} := w_i - \alpha_i z_i$ 
17: end for

```

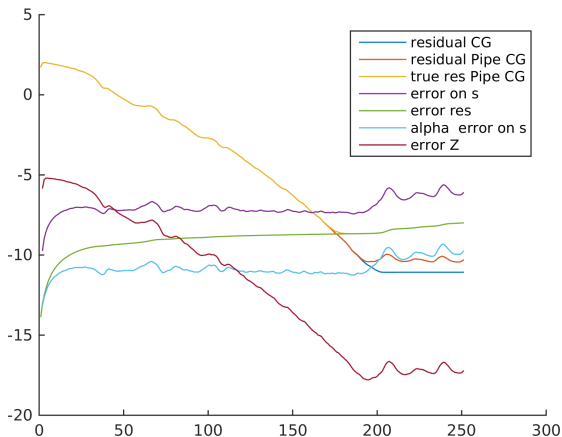
$$\begin{cases} s_i & := Ap_i, \\ w_i & := Ar_i, \\ z_i & := As_i = A^2 p_i. \end{cases} \quad (8)$$

We use the following definition for the errors

$$\begin{cases} \Delta_i^r := \bar{r}_i - (b - A\bar{x}_i) \\ \Delta_i^s := \bar{s}_i - A\bar{p}_i \\ \Delta_i^w := \bar{w}_i - A\bar{r}_i \\ \Delta_i^z := \bar{z}_i - A\bar{s}_i \end{cases} \quad (9)$$

Rounding errors

$$\begin{pmatrix} \Delta_i^r \\ \Delta_i^s \\ \Delta_i^w \\ \Delta_i^z \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i & 0 & 0 \\ 0 & \beta_i & 1 & 0 \\ 0 & 0 & 1 & \alpha_i \\ 0 & 0 & 0 & \beta_i \end{pmatrix} \begin{pmatrix} \Delta_{i-1}^r \\ \Delta_{i-1}^s \\ \Delta_{i-1}^w \\ \Delta_{i-1}^z \end{pmatrix} + \begin{pmatrix} A\delta^x + \delta^r \\ A\delta^r + \delta^s \\ A\delta^r + \delta^w \\ A\delta^s + \delta^z \end{pmatrix} \quad (10)$$



Residual replacement

Replace r_i , s_i , w_i and z_i with their true values.

$$\begin{cases} r_i & := b - Ax_i \\ s_i & := Ap_i, \\ w_i & := Ar_i, \\ z_i & := As_i = A^2 p_i. \end{cases} \quad (11)$$



van der Vorst and Ye, *Residual Replacement strategies for Krylov Subspace iterative methods for Convergence of the True Residuals*, SISC 2000

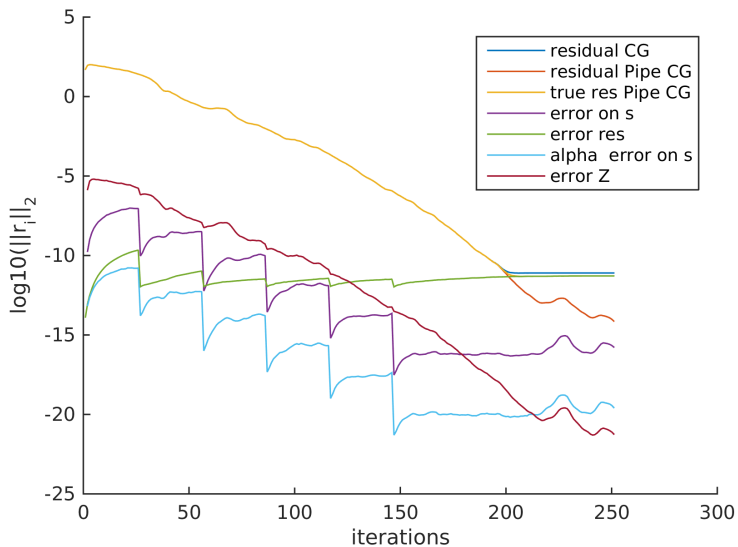


Tong and Ye, *Analysis of the finite precision bi-conjugate gradient algorithm for nonsymmetric linear systems*, Mathematics and computations, 1999.

$$AZ = ZT + f \quad (12)$$

$$f_i = \frac{A\tau_i}{\|r_i\|} + \frac{1}{\alpha_i} \frac{\eta_{i+1}}{\|r_i\|} - \frac{\beta_i}{\alpha_{i-1}} \frac{\eta_i}{\|r_i\|} \quad (13)$$

Residual Replacement for Pipelined Chron/Gear CG



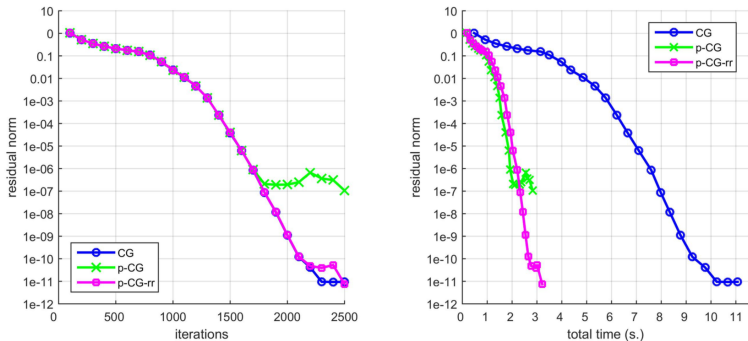
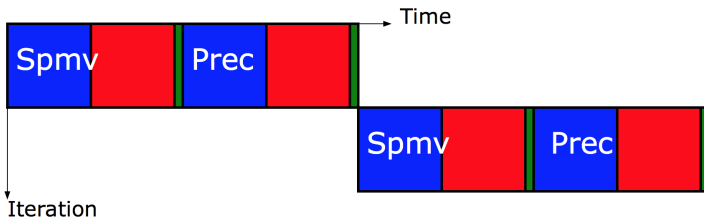


Fig. 4.6: Accuracy experiment on 20 nodes (240 cores) for a 2D Poisson problem with 1.000.000 unknowns. Left: Explicitly computed residual as a function of iterations. Right: Residual as a function of total time spent by the algorithm. Maximal number of iterations is 2500 for all methods; p-CG-rr performed (maximum) 39 replacements.

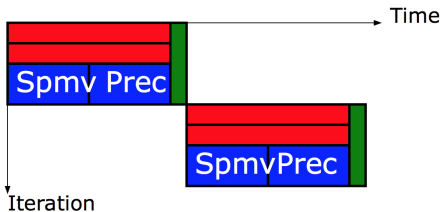


Siegfried Cools, Emrullah Fatih Yetkin, Emmanuel Agullo, Luc Giraud, Wim Vanroose, *Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined Conjugate Gradient method*, arxiv: 1601.07068. To appear in SIAM SIMAX.

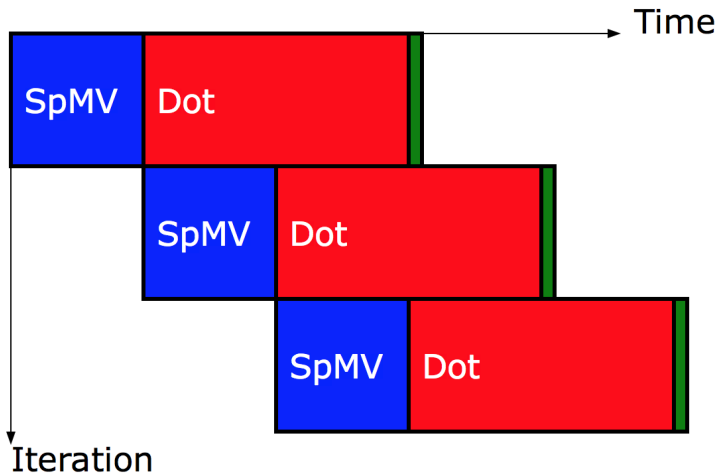
Conjugate Gradients



Pipelined Conjugate Gradients



Extending the pipeline depth



We introduce the auxiliary vectors $Z_{i+1} = [z_0, z_1, \dots, z_i, z_{i+1}]$ defined as

$$z_j := \begin{cases} v_0 & j = 0, \\ P_j(A)v_0 & 0 < j \leq l, \\ P_l(A)v_{j-l} & j > l, \end{cases} \quad (14)$$

where the **polynomials** $P_l(t)$ is fixed to order l , where l is the **depth of the pipeline**

$$P_l(t) := \prod_{j=0}^{l-1} (t - \sigma_j), \quad (15)$$

and for $P_j(t) = (t - \sigma_j)P_{j-1}(t)$ with $P_0(t) := 1$.

Example with $l = 3$

Let $V_4 = [v_0, v_1, v_2, v_3]$ be a Krylov subspace. We then have the auxiliary variable $Z_7 = [z_0, \dots, z_7]$.

$$\begin{aligned}z_0 &= v_0, \\z_1 &= (A - \sigma_0 I)v_0, \\z_2 &= (A - \sigma_1 I)(A - \sigma_0 I)v_0, \\z_3 &= (A - \sigma_2 I)(A - \sigma_1 I)(A - \sigma_0 I)v_0, \\z_4 &= (A - \sigma_2 I)(A - \sigma_1 I)(A - \sigma_0 I)v_1, \\z_5 &= (A - \sigma_2 I)(A - \sigma_1 I)(A - \sigma_0 I)v_2, \\z_6 &= (A - \sigma_2 I)(A - \sigma_1 I)(A - \sigma_0 I)v_3.\end{aligned}\tag{16}$$

We start from the Arnoldi relation

$$AV_k = V_{k+1}H_{k+1,k}, \quad (17)$$

where V_k is the basis of the Krylov subspace and $H_{k+1,k}$ is the upper Hessenberg matrix.

Written for basis vector v_{j-l} , this becomes

$$v_{j-l} = \frac{Av_{j-l-1} - \sum_{k=0}^{j-l-1} h_{k,j-l-1}v_k}{h_{j-l,j-l-1}}. \quad (18)$$

When we apply the fixed polynomial

$$P_l(A)v_{j-l} = \frac{P_l(A)Av_{j-l-1} - \sum_{k=0}^{j-l-1} h_{k,j-l-1}P_l(A)v_k}{h_{j-l,j-l-1}}. \quad (19)$$

This is now, using the auxiliary variables,

$$z_j = \frac{Az_{j-1} - \sum_{k=0}^{j-l-1} h_{k,j-l-1}z_{k+l}}{h_{j-l,j-l-1}}. \quad (20)$$

The auxiliary variables also lie in the Krylov subspace.

$$[z_0, z_1, \dots, z_{k-1}] = [v_0, v_1, \dots, v_{k-1}] \begin{pmatrix} * & * & * & \dots & * & g_{0,k-1} \\ & * & * & \dots & * & g_{1,k-1} \\ & & * & \dots & * & g_{2,k-1} \\ & & & \ddots & \vdots & \\ & & & & * & \\ & & & & & g_{k-1,k-1} \end{pmatrix}$$

so

$$Z_k = V_k G_k, \quad (23)$$

where G_k is a upper triangular matrix.

Recursion for G

$$\begin{aligned}
 \forall j = 0, 1, \dots, k-2 \quad g_{j,k-1} &= (z_{k-1}, v_j) & (24) \\
 &= \left(z_{k-1}, \frac{z_j - \sum_{m=0}^{j-1} g_{m,j} v_m}{g_{j,j}} \right) \\
 &= \frac{(z_{k-1}, z_j) - \sum_{m=0}^{j-1} g_{m,j} g_{m,k-1}}{g_{j,j}}
 \end{aligned}$$

Two cases:

- ▶ If v_j vector is already constructed $\rightarrow (z_{k-1}, v_j)$
- ▶ If v_j not, use (z_{k-1}, z_j) and recurrence.

$$\left(\begin{array}{ccc}
 x & x & x & (z_3, v_0) \\
 & x & x & (z_3, v_1) \\
 & & & \downarrow \\
 & & x & (z_3, z_2) \\
 & & & \downarrow \\
 & & & (z_3, z_3)
 \end{array} \right) \quad (25)$$

for the diagonal elements we use

$$g_{j,j} = \frac{(z_j, z_j) - \sum_{m=0}^{j-1} g_{m,j} g_{m,j}}{g_{j,j}} \quad (26)$$

resulting in

$$g_{j,j} = \sqrt{(z_j, z_j) - \sum_{m=0}^{j-1} g_{m,j}^2} \quad (27)$$

Note that this can lead to breakdowns.

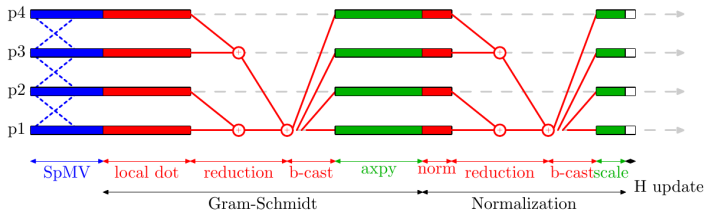
Recursive calculation of the Hessenberg Matrix

$$\begin{aligned}
 & H_{k+1,k} = V_{k+1}^T A V_k \tag{28} \\
 = & V_{k+1}^T \underbrace{A Z_k}_{AZ_k} G_k^{-1} \text{ using } Z = VG \\
 = & \underbrace{V_{k+1}^T Z_{k+1}}_{Z_{k+1}} B_{k+1,k} G_k^{-1} \text{ using } AZ = ZB \\
 = & G_{k+1} B_{k+1,k} G_k^{-1} \text{ using } G_{k+1} = V_{k+1}^T Z_{k+1} \\
 = & \begin{pmatrix} G_k & g_{:,k+1} \\ 0 & g_{k+1,k+1} \end{pmatrix} \begin{pmatrix} B_{k,k-1} & b_{:,k} \\ 0 & b_{k+1,k} \end{pmatrix} \begin{pmatrix} G_k & g_{:,k+1} \\ 0 & g_{k+1,k+1} \end{pmatrix}^{-1} \\
 = & \begin{pmatrix} G_k B_{k,k-1} & G_k b_{:,k} + g_{:,k+1} b_{k+1,k} \\ 0 & g_{k+1,k+1} b_{k+1,k} \end{pmatrix} \begin{pmatrix} G_{k-1}^{-1} & -G_{k-1}^{-1} g_{:,k} g_{k,k}^{-1} \\ 0 & g_{k,k}^{-1} \end{pmatrix} \\
 = & \begin{pmatrix} G_k B_{k,k-1} G_{k-1}^{-1} & (-G_k B_{k,k-1} G_{k-1}^{-1} g_{:,k} + G_k b_{:,k} + g_{:,k+1} b_{k+1,k}) g_{k,k}^{-1} \\ 0 & g_{k+1,k+1} b_{k+1,k} g_{k,k}^{-1} \end{pmatrix} \\
 = & \begin{pmatrix} H_{k,k-1} & (G_k b_{:,k} + g_{:,k+1} b_{k+1,k} - H_{k,k-1} g_{:,k}) g_{k,k}^{-1} \\ 0 & g_{k+1,k+1} b_{k+1,k} g_{k,k}^{-1} \end{pmatrix} \tag{29}
 \end{aligned}$$

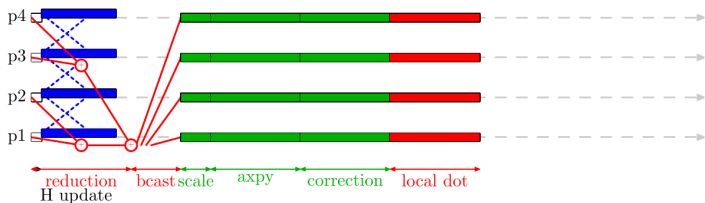
Algorithm 3 p(*l*)-GMRES

1: $r_0 \leftarrow b - Ax_0$; $v_0 \leftarrow r_0 / \|r_0\|$; $z_0 \leftarrow v_0$
2: **for** $i = 0, \dots, m + l$ **do**
3: $w \leftarrow \begin{cases} (A - \sigma_i I)z_i, & i < l \\ Az_i, & i \geq l \end{cases}$
4: $a \leftarrow i - l$
5: **if** $a \geq 0$ **then**
6: $g_{j,a+1} \leftarrow (g_{j,a+1} - \sum_{k=0}^{j-1} g_{k,j} g_{k,a+1}) / g_{j,j}$, $j = a - l + 2, \dots, a$
7: $g_{a+1,a+1} \leftarrow \sqrt{g_{a+1,a+1} - \sum_{k=0}^a g_{k,a+1}^2}$
8: # Check for breakdown and restart or re-orthogonalize if necessary
9: **if** $a < l$ **then**
10: $h_{j,a} \leftarrow (g_{j,a+1} + \sigma_a g_{j,a} - \sum_{k=0}^{a-1} h_{j,k} g_{k,a}) / g_{a,a}$, $j = 0, \dots, a$
11: $h_{a+1,a} \leftarrow g_{a+1,a+1} / g_{a,a}$
12: **else**
13: $h_{j,a} \leftarrow (\sum_{k=0}^{a+1-l} g_{j,k+l} h_{k,a-l} - \sum_{k=j-1}^{a-1} h_{j,k} g_{k,a}) / g_{a,a}$, $j = 0, \dots, a$
14: $h_{a+1,a} \leftarrow g_{a+1,a+1} h_{a+1-l,a-l} / g_{a,a}$
15: **end if**
16: $v_a \leftarrow (z_a - \sum_{j=0}^{a-1} g_{j,a} v_j) / g_{a,a}$
17: $z_{i+1} \leftarrow (w - \sum_{j=0}^{a-1} h_{j,a-1} z_{j+1}) / h_{a,a-1}$
18: **end if**
19: $g_{j,i+1} \leftarrow \begin{cases} \langle z_{i+1}, v_j \rangle, & j = 0, \dots, a \\ \langle z_{i+1}, z_j \rangle, & j = a + 1, \dots, i + 1 \end{cases}$
20: **end for**
21: $y_m \leftarrow \operatorname{argmin} \| (H_{m+1,m} y_m - \|r_0\| e_1) \|_2$
22: $x \leftarrow x_0 + V_m y_m$

Classical GMRES



Pipelined GMRES



Conjugate Gradients with longer pipelines

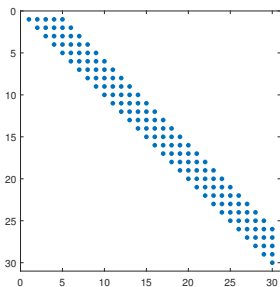
In conjugate gradients is $A = A^T$ and the matrix is positive definite.

- ▶ How does the the GMRES algorithm simplify if $A = A^T$?
- ▶ How can we avoid storing the full basis V and the full set of auxiliary vectors Z ?
- ▶ How do the rounding errors propagate?

Lemma

(Cornelis, Master Thesis 2017) Let A be a symmetric matrix and let $z_i = P_l(A)v_{i-1}$ be auxiliary variables with $l > 0$. The $G_i = V_i^T Z_i$ is now a banded upper triangular matrix with band $2l + 1$, where

$$g_{j,i} = v_j^t z_i = 0 \quad \forall j \notin \{i - 2l, \dots, i\} \quad (30)$$



Updating the solution

In principle we need all basis vectors to construct the solution.

$$x_k = x_0 + V_k y_k = x_0 + V_k T_{k,k}^{-1} \|r_0\| e_1 \quad (31)$$

However, we can introduce the search directions p_k that are also recursively updated. We can then update the current guess with a step in the search direction.

$$p_j = v_{j+1} - \mu_j p_{j-1} \quad (32)$$

$$x_k = x_{k-1} + p_k c_k \quad (33)$$

where c_k and μ_j are determined based on the Hessenberg matrix.



Liesen and Strakos, *Krylov Subspace Methods*, 2012.

Algorithm 15 $p(l)$ -CG

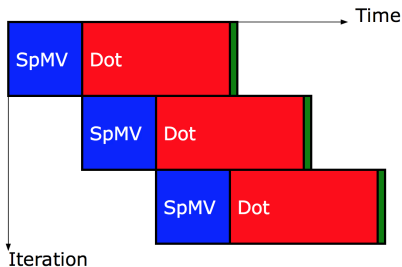
```

1:  $r_0 := b - Ax_0; \beta = \|r_0\|$ ;
2:  $v_0 := \frac{r_0}{\beta}$ ;
3:  $z_0 := v_0$ ;
4:  $g_{0,0} := 1$ ;
5: for  $i = 0, \dots, m + l$  do
6:    $z_{i+1} := \begin{cases} (A - \sigma_i I) z_i & i < l \\ Az_i & i \geq l \end{cases}$ 
7:    $a := i - l$ 
8:   if  $a \geq 0$  then
9:      $g_{j,a+1} := \frac{g_{j,a+1} - \sum_{k=a+1-2l}^{j-1} g_{k,j} g_{k,a+1}}{g_{j,j}}$ ;  $j = a - l + 2, \dots, a$ 
10:     $g_{a+1,a+1} := \sqrt{g_{a+1,a+1} - \sum_{k=a+1-2l}^a g_{k,a+1}^2}$ ;
11:    Controleer op breakdown en herstart indien nodig
12:    if  $a < l$  then
13:       $\gamma_a := \frac{g_{a,a+1} + \sigma_a g_{a,a} - \delta_{a-1} g_{a-1,a}}{g_{a,a}}$ ;
14:       $\delta_a := \frac{g_{a+1,a+1}}{g_{a,a}}$ ;
15:    else
16:       $\gamma_a := \frac{g_{a,a-1} \delta_{a-l-1} + g_{a,a} \gamma_{a-l} + g_{a,a+1} \delta_{a-l} - \delta_{a-1} g_{a-1,a} - \gamma_a g_{a,a}}{g_{a,a}}$ ;
17:       $\delta_a := \frac{g_{a+1,a+1} \delta_{a-l}}{g_{a,a}}$ ;
18:    end if
19:     $v_{a+1} := \frac{z_{a+1} - \sum_{j=a-2l+1}^a g_{j,a+1} v_j}{g_{a+1,a+1}}$ ;
20:     $z_{i+1} := \frac{z_{i+1} - \delta_{a-1} z_{i-1} - \gamma_a z_i}{\delta_a}$ ;
21:  end if
22:
23:  if  $a < 0$  then
24:     $g_{j,i+1} := (z_{i+1}, z_j)$ ,  $j = 0, \dots, i + 1$ 
25:  else
26:     $g_{j,i+1} := \begin{cases} (z_{i+1}, v_j) & j = \max(0, i + 1 - 2l), \dots, a + 1 \\ (z_{i+1}, z_j) & j = a + 2, \dots, i + 1 \end{cases}$ 
27:  end if

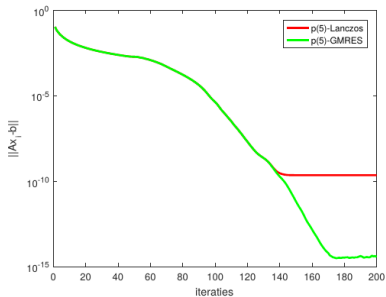
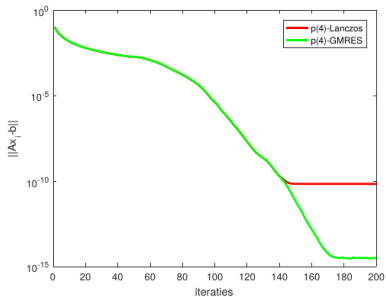
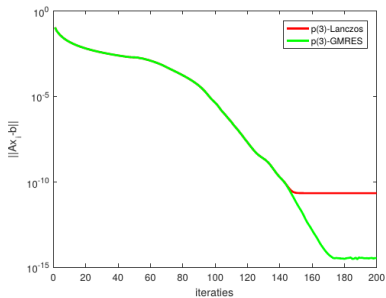
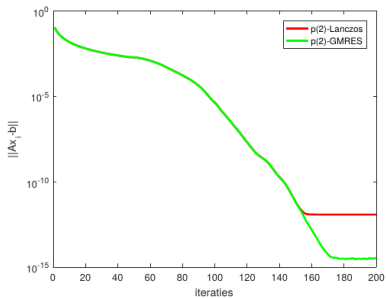
```

Algorithm 4 Schematic representation of $p(l)$ -CG**Input:** $A, M^{-1}, b, x_0, l,$

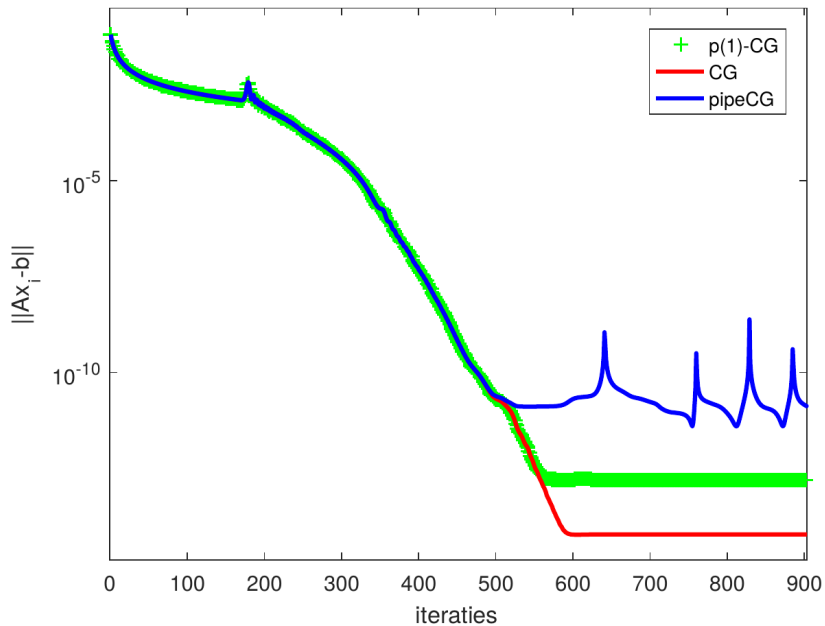
```
1: INITIALIZATION ;
2: for  $i = 0, \dots, m + l$  do
3:   (K1) SPMV ;
4:   if  $i \geq l$  then
5:     MPI.Wait(req(i-1), ...) ;
6:     (K2) SCALAR ;
7:     (K3) SCALAR ;
8:     (K4) AXPY ;
9:   end if
10:  (K5) DOTPR ;
11:  MPI.Iallreduce(..., G(max(0,i-2l+1):i+1,i+1), ..., req(i)) ;
12:  (K6) AXPY ;
13: end for
```



pipelined Lanczos vs classical GMRES



CG vs pipe CG



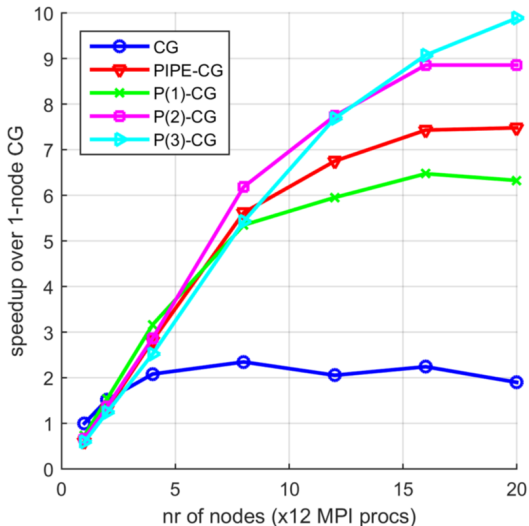


FIG. 5. *Strong scaling experiment on up to 20 nodes (240 processes) for a 5-point stencil 2D Poisson problem with 1,000,000 unknowns. Speedup over single-node classic CG for various pipeline lengths. All methods converged to $\|r_i\|_2/\|b\|_2 = 1.0e-5$ in 1342 iterations.*

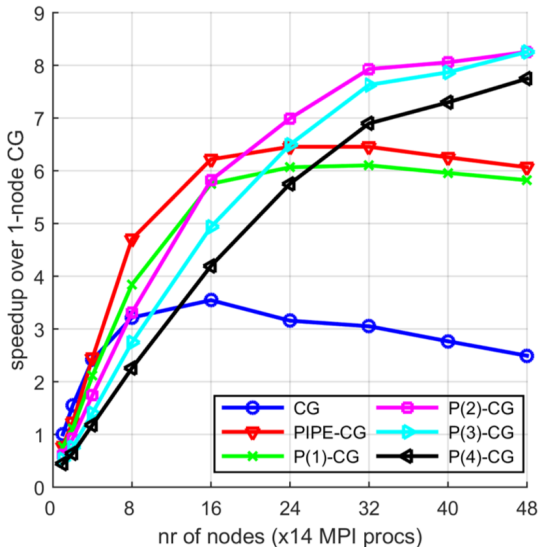
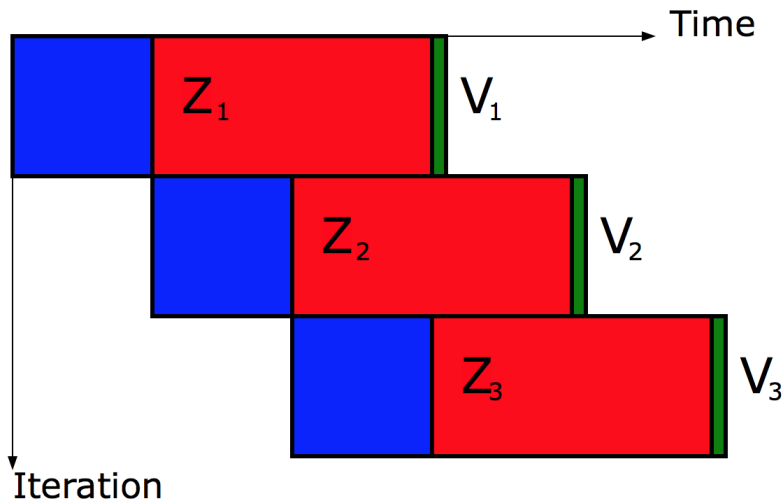
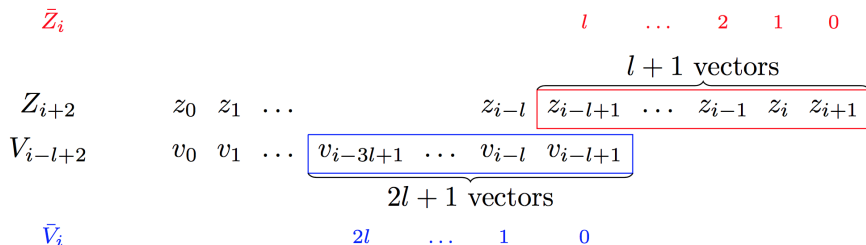


FIG. 6. Strong scaling experiment on up to 48 nodes (672 processes) for a 5-point stencil 2D Poisson problem with 3.062.500 unknowns. Speedup over single-node classic CG for various pipeline lengths. All methods

Pipeline for $l = 2$



Pipeline Storage



Example of pipeline storage for $l = 2$

j	2	z_0	z_0	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
	1		z_1	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9
	0			z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}
			0	1	2	3	4	5	6	7	8	9
							i					

j	4	v_0	v_0	v_0	v_0	v_0	v_0	v_1	v_2	v_3	v_4	
	3			v_1	v_1	v_1	v_1	v_2	v_3	v_4	v_5	
	2				v_2	v_2	v_2	v_3	v_4	v_5	v_6	
	1					v_3	v_3	v_4	v_5	v_6	v_7	
	0						v_4	v_5	v_6	v_7	v_8	
			0	1	2	3	4	5	6	7	8	9
							i					

Contributions to PETSC

- ▶ KSPPGMRES: pipelined GMRES (thanks to Jed Brown)
- ▶ KSPPIPECG: pipelined CG
- ▶ KPPPIPECR: pipelined conjugate residuals
- ▶ KSPGROPPCG: Gropp variant where overlap is done with SPMV and Preconditioner
- ▶ KSPPIPEBCGS: pipelined BiCGStab



Under development:

- ▶ p(l)-CG: in the pull requests of Petsc, being reviewed.

petsc / PETSc / petsc / Pull requests

Scools/pipelcg

#865 **OPEN**

 petscForkpipelcg  scools/pipelcg



 master

[Overview](#)

[Commits](#)

[Activity](#)

Soliciting feedback from applications.






- ▶ Summary:
 - ▶ We have shown that by adding auxiliary variables and additional recurrences it is possible to hide the latencies of the dot products in Krylov methods.
 - ▶ It is possible to write a version of Conjugate Gradients where dot-products are hidden behind multiple SPMV.
 - ▶ *Asynchronous = dot product can take the time of multiple SpMV to complete.*
 - ▶ Better scaling in the strong scaling limit, where global reductions latencies rise and volume of computations per core diminishes.

- ▶ Summary:
 - ▶ We have shown that by adding auxiliary variables and additional recurrences it is possible to hide the latencies of the dot products in Krylov methods.
 - ▶ It is possible to write a version of Conjugate Gradients where dot-products are hidden behind multiple SPMV.
 - ▶ *Asynchronous = dot product can take the time of multiple SpMV to complete.*
 - ▶ Better scaling in the strong scaling limit, where global reductions latencies rise and volume of computations per core diminishes.
- ▶ Lessons Learned:
 - ▶ Rather technical to implement.
 - ▶ Tremendous challenges in the **numerical analysis to control the propagation of rounding errors and avoiding breakdowns.**

Opportunities and Future work

- ▶ The dramatic increase in parallelism with very lightweight cores requires a way to deal with the global reductions.
- ▶ Introduce deep pipelining in various other Krylov methods:
 - ▶ Blocked Krylov methods.
 - ▶ LSQR and CGLS frequently used in inverse and data science problems. There are hardly any preconditioners and often implemented and run on cloud infrastructure.
- ▶ Combine pipelining and communication avoiding techniques such as *s*-step methods,
 - ▶ I. Yamazaki, M. Hoemmen, P. Luszczek and J. Dongarra, (MS40, MS82).
- ▶ Understand performance on large systems and in the presence of noise,
 - ▶ Eller and Gropp (MS40)
 - ▶ Morgan, Knepley and Sanan (MS40).
- ▶ Replace *Bulk Synchronous Parallel* (BSP) implementations of Krylov subspace methods with *Pipelined Asynchronous Parallel* (PAP) where preconditioners, SPMV and dot-product are operating asynchronously.

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), C48C71. (2013)
-  P. Ghysels and W. Vanroose, *Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm*, Parallel Computing, **40**, (2014), Pages 224238
-  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*, arxiv: 1601.07068. To appear in SIAM SIMAX.
-  S. Cools and W. Vanroose, *The communication-hiding pipelined BiCGstab method for the parallel solution of large unsymmetric linear systems*, Parallel Computing Volume 65, July 2017, Pages 1-20
-  J. Cornelis, S. Cools, W. Vanroose, *The Communication-Hiding Conjugate Gradient Method with Deep Pipelines*, arxiv 1801.04728 (2018) Submitted.