WP1: Overview of recent solver developments

A general framework for deriving pipelined Krylov methods:
application to BiCGStab for large and sparse unsymmetric linear systems

EXA2CT meeting, Bordeaux, 22-23 September 2016

University of Antwerp, Applied Mathematics

Siegfried Cools, Wim Vanroose

Bi-Conjugate Gradients Stabilized (BiCGStab)

Algorithm 4 Standard BiCGStab

function BICGSTAB(A, b, x_0)
  $r_0 := b - Ax_0$; $p_0 := r_0$
  for $i = 0, \dots$ **do**
    $s_i := Ap_i$
    compute (r_0, s_i)
    $\alpha_i := (r_0, r_i) \,/\, (r_0, s_i)$
    $q_i := r_i - \alpha_i s_i$
    $y_i := Aq_i$
    compute (q_i, y_i) ; (y_i, y_i)
    $\omega_i := (q_i, y_i) \,/\, (y_i, y_i)$
    $x_{i+1} := x_i + \alpha_i p_i + \omega_i q_i$
    $r_{i+1} := q_i - \omega_i y_i$
    compute (r_0, r_{i+1})
    $\beta_i := (\alpha_i/\omega_i)\,(r_0, r_{i+1}) \,/\, (r_0, r_i)$
    $p_{i+1} := r_{i+1} + \beta_i\,(p_i - \omega_i s_i)$
  end for
end function

dot-prod
SpMV
axpy

Traditional BiCGStab:
(non-preconditioned)

Global communication

- 3 global reduction phases

Semi-local communication

- 2 non-overlapping SpMVs

Local communication

- 4 axpy(-like) operations

General two-step framework for deriving pipelined Krylov methods:

Step 1. **Avoiding communication:** *merge global reductions*

Step 2. **Hiding communication:** *overlap SpMVs & global reductions*

Step 1. Avoiding global communication

Algorithm 4 Standard BiCGStab

1: **function** BICGSTAB(A, b, x_0)
2: $r_0 := b - Ax_0$; $p_0 := r_0$
3: **for** $i = 0, \ldots$ **do**
4: $s_i := Ap_i$
5: **compute** (r_0, s_i)
6: $\alpha_i := (r_0, r_i) \,/\, (r_0, s_i)$
7: $q_i := r_i - \alpha_i s_i$
8: $y_i := Aq_i$
9: **compute** (q_i, y_i) ; (y_i, y_i)
10: $\omega_i := (q_i, y_i) \,/\, (y_i, y_i)$
11: $x_{i+1} := x_i + \alpha_i p_i + \omega_i q_i$
12: $r_{i+1} := q_i - \omega_i y_i$
13: **compute** (r_0, r_{i+1})
14: $\beta_i := (\alpha_i/\omega_i)\,(r_0, r_{i+1}) \,/\, (r_0, r_i)$
15: $p_{i+1} := r_{i+1} + \beta_i\,(p_i - \omega_i s_i)$
16: **end for**
17: **end function**

dot-prod
SpMV
axpy

(a) *Identify* two global comm. phases for merger (lines 5-6 & 13-14)

(b) *Rewrite* SpMV as recurrence:

$$s_i = Ap_i = w_i + \beta_{i-1}\,(s_{i-1} - \omega_{i-1} z_{i-1})\,,$$

define $w_i := Ar_i$, $z_i := As_i$ and note that $y_i := w_i - \alpha_i z_i$

(c) *Rewrite* dot-product using (b):

$$(r_0, s_i) = (r_0, w_i + \beta_{i-1}\,(s_{i-1} - \omega_{i-1} z_{i-1})),$$

independent of interlying variables

(d) *Move* dot-product (lines 5-6) upward and merge with existing global comm. phase (lines 13-14)

Communication-avoiding BiCGStab (CA-BiCGStab)

Algorithm 5 Communication avoiding BiCGStab

function CA-BICGSTAB(A, b, x_0)
  $r_0 := b - Ax_0$; $w_0 := Ar_0$; $\alpha_0 := (r_0, r_0) \,/\, (r_0, w_0)$; $\beta_{-1} := 0$
  for $i = 0, \ldots$ **do**
    $p_i := r_i + \beta_{i-1}\,(p_{i-1} - \omega_{i-1} s_{i-1})$
    $s_i := w_i + \beta_{i-1}\,(s_{i-1} - \omega_{i-1} z_{i-1})$
    $z_i := A s_i$
    $q_i := r_i - \alpha_i s_i$
    $y_i := w_i - \alpha_i z_i$
    compute (q_i, y_i) ; (y_i, y_i)
    $\omega_i := (q_i, y_i) \,/\, (y_i, y_i)$
    $x_{i+1} := x_i + \alpha_i p_i + \omega_i q_i$
    $r_{i+1} := q_i - \omega_i y_i$
    $w_{i+1} := A r_{i+1}$
    compute (r_0, r_{i+1}) ; (r_0, w_{i+1}) ; (r_0, s_i) ; (r_0, z_i)
    $\beta_i := (\alpha_i / \omega_i)\,(r_0, r_{i+1}) \,/\, (r_0, r_i)$
    $\alpha_{i+1} := (r_0, r_{i+1}) \,/\, ((r_0, w_{i+1}) + \beta_i\,(r_0, s_i) - \beta_i \omega_i\,(r_0, z_i))$
  end for
end function

dot-prod
SpMV
axpy

CA-BiCGStab:
(non-preconditioned)

Global communication

- 2 global red. phases (vs. 3)

Semi-local communication

- 2 non-overlapping SpMVs

Local communication

- 6 axpy(-like) operations (vs. 4)

Status after Step 1:

no. global comm. phases reduced from 3 to 2, at the cost of 2 additional axpys

Note: further reduction from 2 to 1 global comm. phase possible, but not recommended (see later).

Step 2. Hiding global communication

Algorithm 5 Communication avoiding BiCGStab

1: **function** CA-BICGSTAB(A, b, x_0)
2: $r_0 := b - Ax_0$; $w_0 := Ar_0$; $\alpha_0 := (r_0, r_0) / (r_0, w_0)$; $\beta_{-1} := 0$
3: **for** $i = 0, \dots$ **do**
4: $p_i := r_i + \beta_{i-1} (p_{i-1} - \omega_{i-1} s_{i-1})$
5: $s_i := w_i + \beta_{i-1} (s_{i-1} - \omega_{i-1} z_{i-1})$
6: $z_i := A s_i$
7: $q_i := r_i - \alpha_i s_i$
8: $y_i := w_i - \alpha_i z_i$
9: **compute** (q_i, y_i) ; (y_i, y_i)
10: $\omega_i := (q_i, y_i) / (y_i, y_i)$
11: $x_{i+1} := x_i + \alpha_i p_i + \omega_i q_i$
12: $r_{i+1} := q_i - \omega_i y_i$
13: $w_{i+1} := A r_{i+1}$
14: **compute** (r_0, r_{i+1}) ; (r_0, w_{i+1}) ; (r_0, s_i) ; (r_0, z_i)
15: $\beta_i := (\alpha_i / \omega_i) (r_0, r_{i+1}) / (r_0, r_i)$
16: $\alpha_{i+1} := (r_0, r_{i+1}) / ((r_0, w_{i+1}) + \beta_i (r_0, s_i) - \beta_i \omega_i (r_0, z_i))$
17: **end for**
18: **end function**

dot-prod
SpMV
axpy

(a) *Identify* SpMV / global reduction pairs (lines 6 & 9 and 13 & 14)

(b) *Rewrite* SpMVs as recurrences:

$$z_i := A s_i = t_i + \beta_{i-1} (z_{i-1} - \omega_{i-1} v_{i-1}),$$
$$w_{i+1} := A r_{i+1} = y_i - \omega_i (t_i - \alpha_i v_i),$$

define $t_i := A w_i$, $v_i := A z_i$

(c) *Check* SpMV / global reduction pairwise dependencies:

- line 9 independent of v_i? **yes**
- line 14 indep. of t_{i+1}? **yes**

(d) *Insert* new SpMVs *below* corresponding global comm. phases

Pipelined BiCGStab (p-BiCGStab)

Algorithm 6 Pipelined BiCGStab

function PIPE-BICGSTAB(A, b, x_0)
    $r_0 := b - Ax_0$; $w_0 := Ar_0$; $t_0 := Aw_0$;
    for $i = 0, \ldots$ **do**
        $p_i := r_i + \beta_{i-1}\,(p_{i-1} - \omega_{i-1} s_{i-1})$
        $s_i := w_i + \beta_{i-1}\,(s_{i-1} - \omega_{i-1} z_{i-1})$
        $z_i := t_i + \beta_{i-1}\,(z_{i-1} - \omega_{i-1} v_{i-1})$
        $q_i := r_i - \alpha_i s_i$
        $y_i := w_i - \alpha_i z_i$
        compute (q_i, y_i) ; (y_i, y_i)
        $\omega_i := (q_i, y_i) \,/\, (y_i, y_i)$
        overlap $v_i := Az_i$
        $x_{i+1} := x_i + \alpha_i p_i + \omega_i q_i$
        $r_{i+1} := q_i - \omega_i y_i$
        $w_{i+1} := y_i - \omega_i\,(t_i - \alpha_i v_i)$
        compute (r_0, r_{i+1}) ; (r_0, w_{i+1}) ; (r_0, s_i) ; (r_0, z_i)
        $\beta_i := (\alpha_i/\omega_i)\,(r_0, r_{i+1}) \,/\, (r_0, r_i)$
        $\alpha_{i+1} := (r_0, r_{i+1}) \,/\, ((r_0, w_{i+1}) + \beta_i\,(r_0, s_i) - \beta_i \omega_i\,(r_0, z_i))$
        overlap $t_{i+1} := Aw_{i+1}$
    end for
end function

dot-prod
SpMV
axpy

p-BiCGStab:

(non-preconditioned)

Global communication

- 2 global red. phases (vs. 3)

Semi-local communication

- 2 overlapping SpMVs

Local communication

- 8 axpy(-like) operations (vs. 4)

Status after Step 2:

both global comm. phases are overlapped with SpMV computations ('hidden'), at the cost of 4 additional axpys compared to standard BiCGStab

Preconditioned pipelined BiCGStab

Algorithm 8 Preconditioned Pipelined BiCGStab

function P-PIPE-BICGSTAB(A, M^{-1}, b, x_0)
  $r_0 := b - Ax_0$; $\hat{r}_0 := M^{-1}r_0$; $w_0 := A\hat{r}_0$; $\hat{w}_0 := M^{-1}w_0$
  $t_0 := A\hat{w}_0$; $\alpha_0 := (r_0, r_0) \,/\, (r_0, w_0)$; $\beta_{-1} := 0$
  for $i = 0, \dots$ **do**
    $\hat{p}_i := \hat{r}_i + \beta_{i-1}\,(\hat{p}_{i-1} - \omega_{i-1}\hat{s}_{i-1})$
    $s_i := w_i + \beta_{i-1}\,(s_{i-1} - \omega_{i-1}z_{i-1})$
    $\hat{s}_i := \hat{w}_i + \beta_{i-1}\,(\hat{s}_{i-1} - \omega_{i-1}\hat{z}_{i-1})$
    $z_i := t_i + \beta_{i-1}\,(z_{i-1} - \omega_{i-1}v_{i-1})$
    $q_i := r_i - \alpha_i s_i$
    $\hat{q}_i := \hat{r}_i - \alpha_i \hat{s}_i$
    $y_i := w_i - \alpha_i z_i$
    compute (q_i, y_i) ; (y_i, y_i)
    $\omega_i := (q_i, y_i) \,/\, (y_i, y_i)$
    overlap $\hat{z}_i := M^{-1}z_i$
    overlap $v_i := A\hat{z}_i$
    $x_{i+1} := x_i + \alpha_i\hat{p}_i + \omega_i\hat{q}_i$
    $r_{i+1} := q_i - \omega_i y_i$
    $\hat{r}_{i+1} := \hat{q}_i - \omega_i\,(\hat{w}_i - \alpha_i\hat{z}_i)$
    $w_{i+1} := y_i - \omega_i\,(t_i - \alpha_i v_i)$
    compute (r_0, r_{i+1}) ; (r_0, w_{i+1}) ; (r_0, s_i) ; (r_0, z_i)
    $\beta_i := (\alpha_i/\omega_i)\,(r_0, r_{i+1}) \,/\, (r_0, r_i)$
    $\alpha_{i+1} := (r_0, r_{i+1}) \,/\, ((r_0, w_{i+1}) + \beta_i\,(r_0, s_i) - \beta_i\omega_i\,(r_0, z_i))$
    overlap $\hat{w}_{i+1} := M^{-1}w_{i+1}$
    overlap $t_{i+1} := A\hat{w}_{i+1}$
  end for
end function

dot-prod
SpMV
axpy

Like for any pipelined method, including a preconditioner is easy.

p-BiCGStab:
(preconditioned)

Global communication

- 2 global red. phases (vs. 3)

Semi-local communication

- 2 overlapping Prec + SpMVs

Local communication

- 11 axpy(-like) operations (vs. 4)

Pipelined BiCGStab vs. Improved BiCGStab

L.T. Yang and R.P. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing, pp. 324–328, IEEE, 2002.

	GLRED	SPMV	Flops (AXPY + DOT-PROD)	Time (GLRED + SPMV)	Memory
BiCGStab	3	2	20	3 GLRED + 2 SPMV	7
IBiCGStab	1	2	30	1 GLRED + 2 SPMV	10
p-BiCGstab	2	2^*	38	2 max(GLRED, SPMV)	11

If time(GLRED) $\approx$ time(SPMV):

- speed-up factor p-BiCGStab/BiCGStab = 2.5
- speed-up factor IBiCGStab/BiCGStab = 1.66

If time(GLRED) $\gg$ time(SPMV):

- speed-up factor p-BiCGStab/BiCGStab = 2.5
- speed-up factor IBiCGStab/BiCGStab = 3.0

Pipelined BiCGStab vs. Improved BiCGStab

L.T. Yang and R.P. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing, pp. 324–328, IEEE, 2002.

	GLRED	SPMV	Flops (AXPY + DOT-PROD)	Time (GLRED + SPMV)	Memory
BiCGStab	3	2	20	3 GLRED + 2 SPMV	7
IBiCGStab	1	2	30	1 GLRED + 2 SPMV	10
p-BiCGstab	2	2*	38	2 max(GLRED, SPMV)	11

Is combination of both methods (1 GLRED overlapped with all SPMVs) possible?

Theoretically, yes, however:

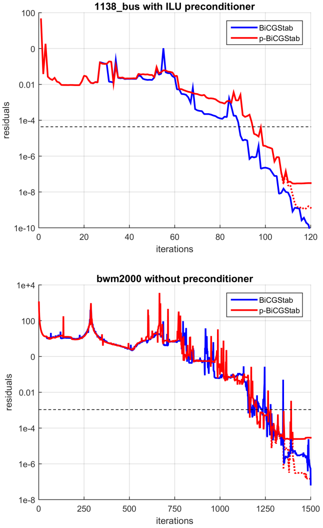
- no. axpys is <u>much</u> larger → algorithm robustness decreases (rounding errors)
- one <u>extra SpMV</u> required → increase in computational cost
  → further loss of robustness/flop equivalence

Convergence results: p-BiCGStab

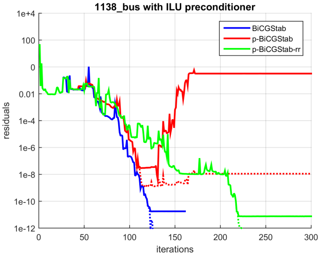
MatrixMarket collection unsymmetric test problems `tol = 1e-6`

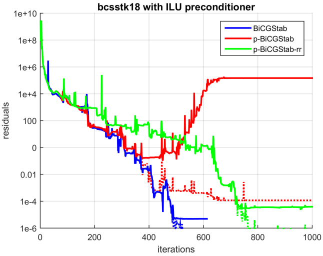
Matrix	Prec	$\kappa(A)$	N	#nnz	$\|r_0\|_2$	BiCGStab		p-BiCGStab	
						iter	$\|r_i\|_2$	iter	$\|r_i\|_2$
1138_bus	ILU	8.6e+06	1138	4054	4.3e+01	89	1.4e-05	95	1.8e-05
add32	ILU	1.4e+02	4960	19,848	8.0e-03	19	5.9e-09	19	5.9e-09
bcsstk14	ILU	1.3e+10	1806	63,454	2.1e+09	315	1.6e+03	322	1.2e+03
bcsstk18	ILU	6.5e+01	11,948	149,090	2.6e+09	84	2.2e+03	102	2.0e+03
bcsstk26	ILU	1.7e+08	1922	30,336	3.5e+09	113	2.8e+03	107	2.9e+03
bcsstm25	-	6.1e+09	15,439	15,439	6.9e+07	928	6.8e+01	825	6.5e+01
bfw782a	ILU	1.7e+03	782	7514	3.2e-01	72	1.1e-07	65	6.2e-08
bwm2000	-	2.4e+05	2000	7996	1.1e+03	1156	6.6e-04	1162	9.1e-04
cdde6	ILU	1.8e+02	961	4681	5.8e-01	9	2.2e-07	9	2.2e-07
fidap014	-	3.5e+16	3251	65,747	2.7e+06	121	2.6e+00	123	2.6e+00
fs_760_3	ILU	1.0e+20	760	5816	1.6e+07	930	1.4e+01	709	1.1e+01
jagmesh9	-	6.0e+03	1349	9101	6.8e+00	1022	6.4e-06	996	6.6e-06
jpwh_991	ILU	1.4e+02	991	6027	3.8e-01	9	2.9e-07	9	2.9e-07
orsreg_1	ILU	6.7e+03	2205	14,133	4.8e+00	25	2.7e-06	25	2.7e-06
pde2961	ILU	6.4e+02	2961	14,585	2.9e-01	31	1.3e-07	31	4.5e-08
rdb3200l	-	1.1e+03	3200	18,880	1.0e+01	149	5.1e-06	145	9.1e-06
s3dkq4m2	-	1.9e+11	90,449	2,455,670	6.8e+01	3736	5.8e-05	3500	6.2e-05
saylr4	ILU	6.9e+06	3564	22,316	3.1e-03	40	3.0e-09	39	1.5e-09
sherman3	ILU	5.5e+18	5005	20,033	1.8e+01	98	3.7e-06	83	9.1e-06
sstmodel	-	2.7e+18	3345	22,749	7.9e+00	6968	7.7e-06	4399	7.5e-06
utm5940	ILU	4.3e+08	5940	83,842	3.6e-01	223	4.0e-08	244	3.5e-07
Average iter deviation wrt BiCGStab								-3.5%	

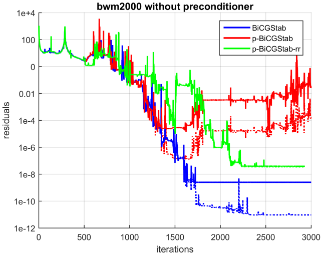
Convergence results: p-BiCGStab

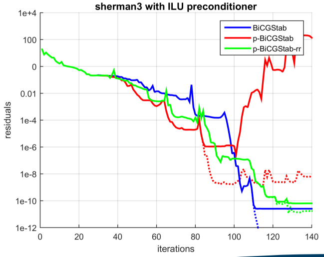


Robustness and attainable accuracy: p-BiCGStab-rr

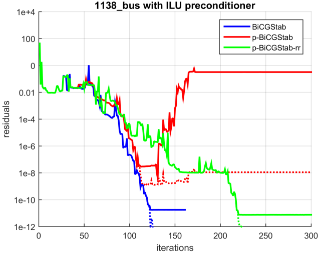


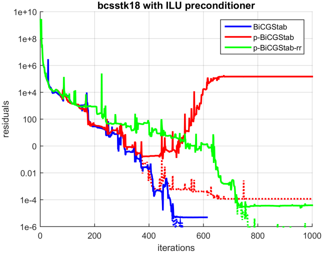






Robustness and attainable accuracy: p-BiCGStab-rr





Residual replacement every rr-th iteration
(non-automated, i.e. rr is a parameter of the method, but chosen large s.t. no. res. repl. is small)

$$r_i := b - Ax_i, \qquad \hat{r}_i := M^{-1} r_i, \qquad w_i := A\hat{r}_i,$$

$$s_i := A\hat{p}_i, \qquad \hat{s}_i := M^{-1} s_i, \qquad z_i := A\hat{s}_i.$$

- ⊕ increased **maximal attainable accuracy**: comparable to BiCGStab level
- ⊕ increased **robustness**: negates instable true residual behaviour
- ⊖ increased **number of iterations** possible

Robustness and attainable accuracy: p-BiCGStab-rr

MatrixMarket collection unsymmetric test problems `tol = 1e-20`

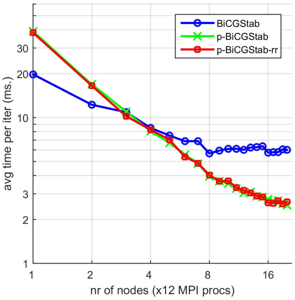
Matrix	$\|r_0\|_2$	BiCGStab		p-BiCGStab		p-BiCGStab-rr			
		iter	$\|r_i\|_2$	iter	$\|r_i\|_2$	iter	$\|r_i\|_2$	k	#nrr
1138_bus	4.3e+01	124	1.8e-11	130	4.0e-09	220	7.4e-12	35	3
add32	8.0e-03	46	7.8e-18	42	5.0e-16	51	5.7e-18	10	2
bcsstk14	2.1e+09	559	7.3e-06	444	6.6e-01	522	3.8e-03	200	2
bcsstk18	2.6e+09	523	4.8e-06	450	1.1e-01	725	2.8e-05	50	7
bcsstk26	3.5e+09	414	1.1e-05	216	5.7e-01	475	8.6e-04	30	6
bcsstm25	6.9e+07	-	3.2e+00	-	3.8e+00	-	4.6e+00	1000	9
bfw782a	3.2e-01	117	9.5e-14	106	5.1e-13	133	2.6e-15	20	4
bwm2000	1.1e+03	1733	2.5e-09	1621	1.4e-05	2231	3.8e-08	500	3
cdde6	5.8e-01	151	8.1e-14	147	2.0e-11	159	2.1e-15	10	13
fidap014	2.7e+06	-	4.3e-03	-	9.7e-03	-	4.3e-03	50	3
fs_760_3	1.6e+07	1979	1.2e-05	1039	5.1e-02	4590	1.1e-05	900	3
jagmesh9	6.8e+00	6230	2.4e-14	3582	5.8e-09	9751	1.1e-11	500	13
jpwh_991	3.8e-01	53	1.3e-14	54	1.8e-12	63	2.5e-15	10	4
orsreg_1	4.8e+00	51	4.0e-11	52	3.7e-09	56	5.9e-12	10	3
pde2961	2.9e-01	50	4.5e-15	48	3.4e-13	52	1.4e-15	10	3
rdb3200l	1.0e+01	178	3.7e-08	167	9.9e-08	181	3.4e-08	100	1
s3dkq4m2	6.8e+01	-	1.0e-05	-	1.4e-05	-	1.3e-05	1000	9
saylr4	3.1e-03	52	4.0e-12	43	7.5e-11	46	1.8e-12	10	4
sherman3	1.8e+01	111	2.5e-11	100	2.8e-07	128	6.2e-11	20	4
sstmodel	7.9e+00	-	5.1e-06	-	3.1e-06	-	4.5e-06	1000	9
utm5940	3.6e-01	256	3.0e-12	248	4.3e-08	395	2.9e-11	100	3
Average iter deviation wrt BiCGStab				-11.0%		22.1%			
Average #nrr wrt p-BiCGStab-rr iter									2.4%

Performance benchmark: strong scaling results

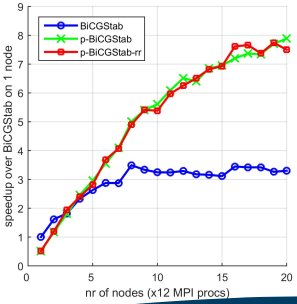
- PETSc implementation using MPICH-3.1.3 communication
- Hardware specs: 1-20 LYNX nodes (12-240 cores)
- Benchmark problem: 1.000.000 unknowns 2D model with asymmetric stencil

$$A^{stencil} = \begin{pmatrix} & 1 & \\ 1 & -4 & \varepsilon \\ & \varepsilon & \end{pmatrix}, \qquad \text{with } \varepsilon = 0.999$$

CPU time i.f.o. number of nodes



Speedup over 1-node BiCGStab

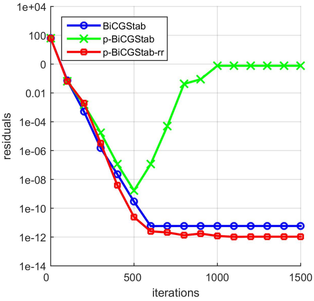


Performance benchmark: accuracy results

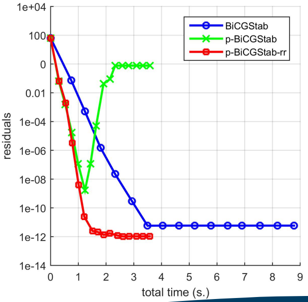
- PETSc implementation using MPICH-3.1.3 communication
- Hardware specs: 20 LYNX nodes (240 cores)
- Benchmark problem: 1.000.000 unknowns 2D model with asymmetric stencil

$$A^{stencil} = \begin{pmatrix} & 1 & \\ 1 & -4 & \varepsilon \\ & \varepsilon & \end{pmatrix}, \quad \text{with } \varepsilon = 0.999$$

Residuals i.f.o. number of iterations



Residuals i.f.o. total time spent



Conclusions and outlook

Overall conclusions

- General framework for deriving pipelined variants of existing Krylov methods
- Prove-of-concept: successfully applied to BiCGStab
- p-BiCGStab displays good performance, but slightly decreased robustness
- Residual replacement strategy improves robustness and max. att. accuracy

Status summary

- p-BiCGStab prototype Matlab code available
- p-BiCGStab PETSc implementation finished

  $\rightarrow$ to be made publicly available in open-source PETSc distribution asap
- Technical report nearly (80%) completed

  $\rightarrow$ to be submitted as full paper in $\sim$ 1-2 months

References

P. Ghysels, T.J. Ashby, K. Meerbergen, and W. Vanroose. Hiding global communication latency in the GMRES algorithm on massively parallel machines. SIAM Journal on Scientific Computing, 35(1):C48–C71, 2013.

P. Ghysels and W. Vanroose. Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. Parallel Computing, 40(7):224–238, 2014.

H.A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 13(2):631–644, 1992.

H.A. Van der Vorst and Q. Ye. Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. SIAM SISC, 22(3):835–852, 2000.

L.T. Yang and R.P. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing, pp. 324–328, IEEE, 2002.