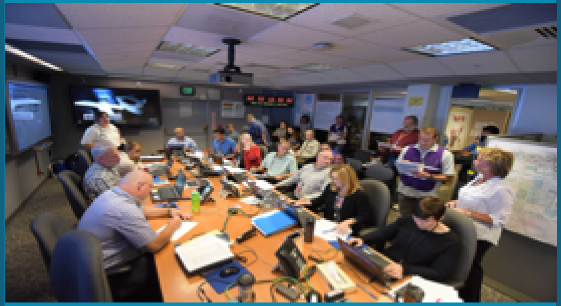
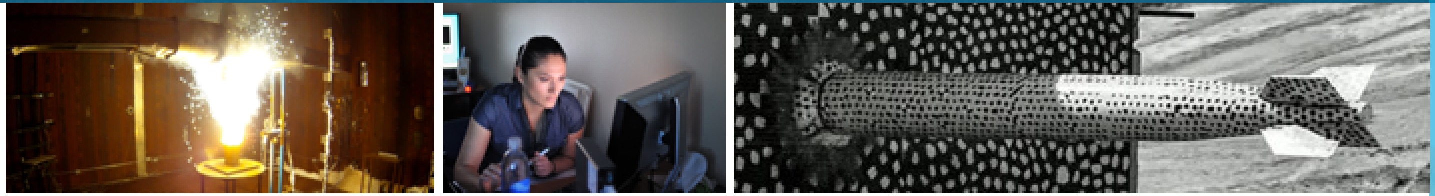
SAND2022-2093C

Mixed Precision Strategies for GMRES





Jennifer Loe, Christian Glusa, Ichitaro Yamazaki, Erik Boman, Sivasankaran Rajamanickam

SIAM Parallel Processing 2022

The idea behind GMRES the (Generalized Minimum RESidual Method):

To solve $Ax = b$, where A is $n \times n$:

1. Build an orthonormal basis for a Krylov subspace:

$$\text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

2. Use an orthogonal projection to find an approximate solution which minimizes the residual:

$$\| b - Ax \|_2$$

GMRES (Generalized Minimum RESidual) Algorithm:

Algorithm GMRES (Modified Gram-Schmidt)

```
1: $\gamma = \|b\|_2$ and $v_1 = b/\gamma$
2: **for** $j = 1 : m$ **do**
3:     $w_j = Av_j$
4:     **for** $i = 1 : j$ **do**
5:         $h_{ij} = v_i^T w_j$
6:         $w_j = w_j - h_{ij} v_i$
7:     **end for**
8:     $h_{j+1,j} = \|w_j\|_2$.
9:     $v_{j+1} = w_j / h_{j+1,j}$
10: **end for**
11: Define the $(m+1) \times m$ matrix $\overline{H} = \{h_{ij}\}$
12: Solve least-squares problem $\overline{H}d = \gamma e_1$ for $d$.
13: $\hat{x} = V_m d$
```

Sparse Matrix-Vector Product (SpMV)

Orthogonalizing the next basis vector

Restart when subspace size gets too large!

See details in “Iterative Methods for Sparse Linear Systems 2nd ed.” by Saad.

Why incorporate lower precisions in GMRES?

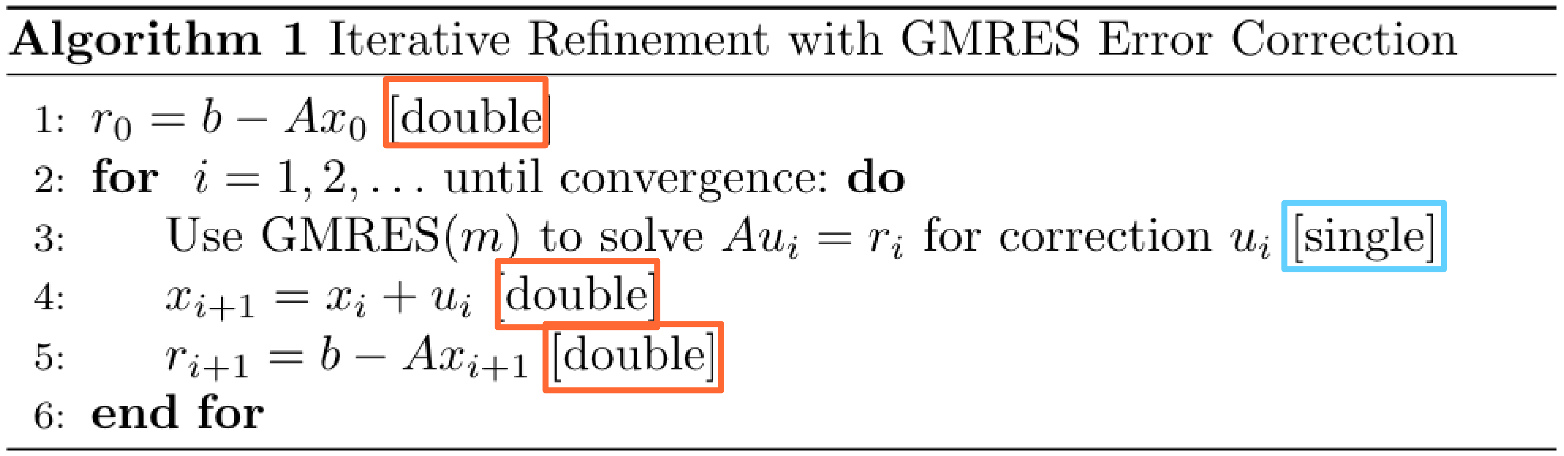
- Reduce data movement to overcome memory-bound algorithms.
- Use cheaper floating-point operations.

Obstacles to lower precision:

- Lower precision computations result in more roundoff error!
- …but applications still need high level of accuracy in solutions.
- Tricky to find where to use lower precision in algorithm while maintaining accuracy.

So how DO we use lower precision in GMRES?

Iterative Refinement with GMRES (GMRES-IR)



Algorithm 1 Iterative Refinement with GMRES Error Correction

```
1: $r_0 = b - Ax_0$ [double]
2: for $i = 1, 2, \ldots$ until convergence: do
3:     Use GMRES($m$) to solve $Au_i = r_i$ for correction $u_i$ [single]
4:     $x_{i+1} = x_i + u_i$ [double]
5:     $r_{i+1} = b - Ax_{i+1}$ [double]
6: end for
```

(At each restart, update solution vector and recompute residuals in double precision.)

Note: We store TWO copies of matrix A (double and single).

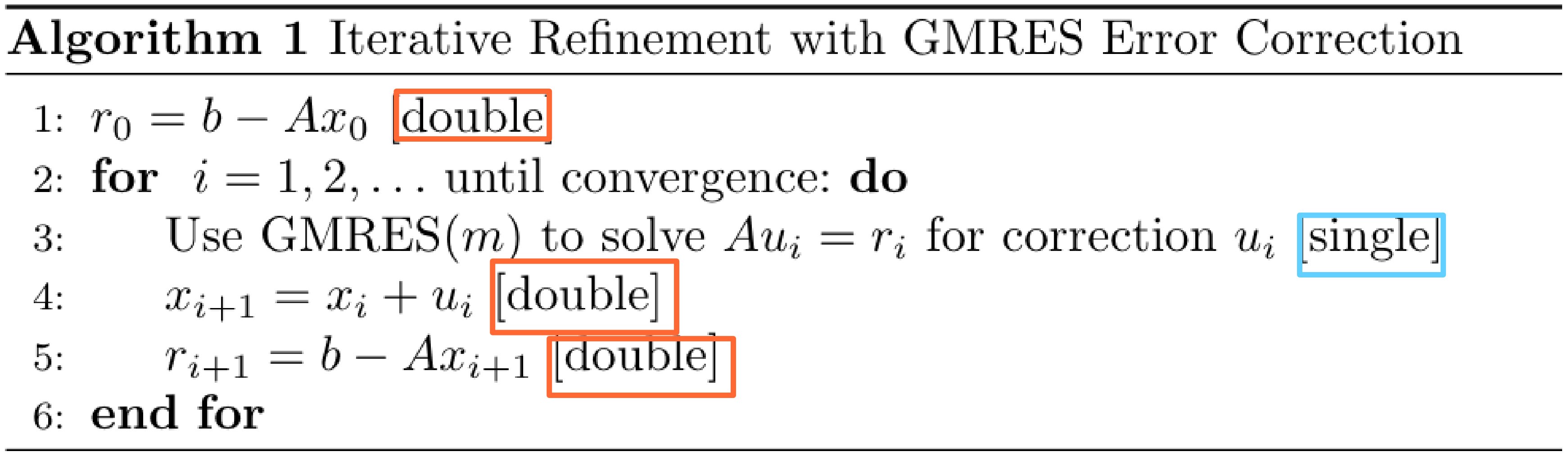
Not a new algorithm. See related works:

- Neil Lindquist, Piotr Luszczek, and Jack Dongarra. *Improving the performance of the GMRES method using mixed-precision techniques.*
- Hartwig Anzt, Vincent Heuveline, and Bjorn Rocker. *Mixed precision iterative refinement methods for linear systems: Convergence analysis based on Krylov subspace methods.*
- Erin Carson and Nicholas J. Higham. *Accelerating the solution of linear systems by iterative refinement in three precisions.*

Implementation of Krylov Solvers in Trilinos

- Belos: Linear Solvers package in Trilinos:
  - All linear algebra kernels are abstracted through “adapter” interface.
  - Solvers interface does not support mixing precisions! Mixed precision must occur through the adapter.
- Kokkos and Kokkos Kernels:
  - Portable parallel linear algebra.
  - Performant BLAS kernels for GPU (single node).
- New Mixed Precision Krylov Solvers Software:
  - New adapter to use Kokkos as the linear algebra backend for solvers
  - Tested performance on a single node with V100 GPU.

Experiment Setup:



Algorithm 1 Iterative Refinement with GMRES Error Correction

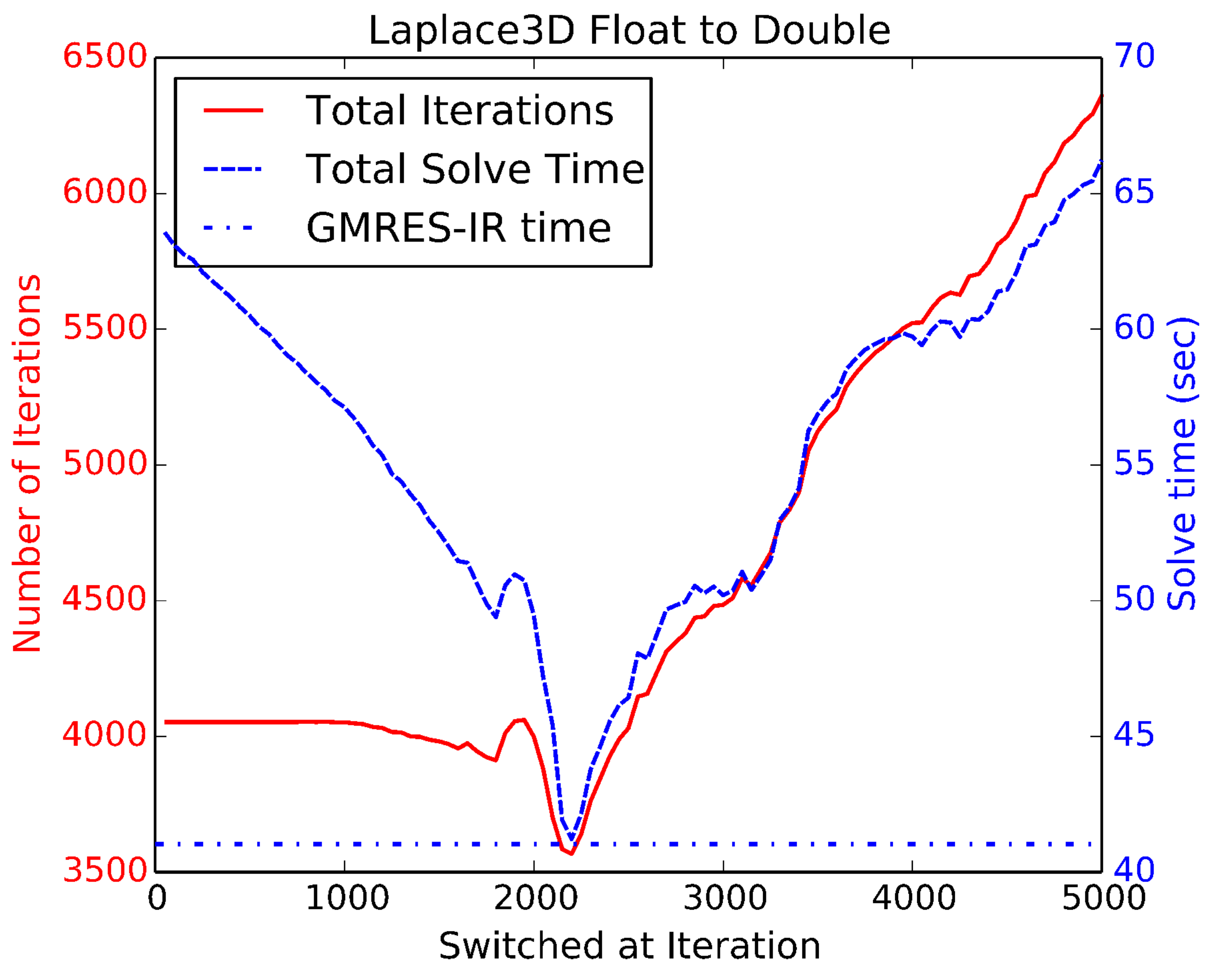
```
1: $r_0 = b - Ax_0$ $\boxed{\text{[double]}}$
2: for $i = 1, 2, \ldots$ until convergence: do
3:     Use GMRES($m$) to solve $Au_i = r_i$ for correction $u_i$ $\boxed{\text{[single]}}$
4:     $x_{i+1} = x_i + u_i$ $\boxed{\text{[double]}}$
5:     $r_{i+1} = b - Ax_{i+1}$ $\boxed{\text{[double]}}$
6: end for
```

<u>Experiment parameters:</u>

- Restarting GMRES at every 50 iterations.
- Recompute residuals in double at each restart (step 4 & 5).
- Stopping when relative residual less than 1e-10. ($||b - Ax||_2/||b|| < 10^{-10}$)
- Tests run on a V100 GPU.

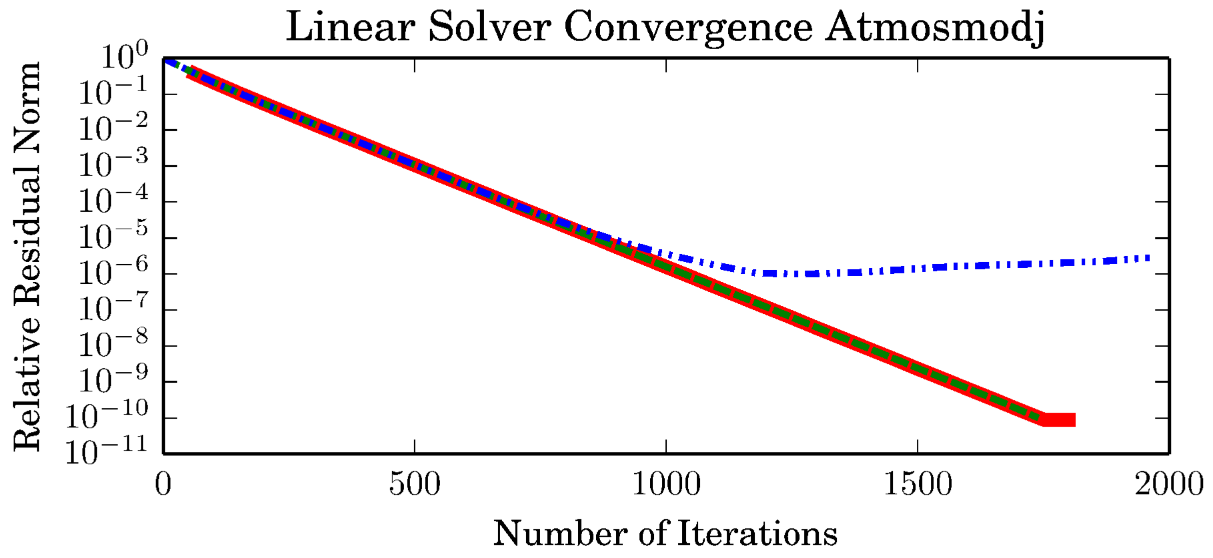
GMRES-IR wins over “switching” strategy (GMRES-FD):

- What if we run GMRES in single precision and then switch to double precision?
- But where to switch?
- GMRES-FD (float-double switch)
  - Min solve time: 41.22s
  - Min iterations: 3567
- GMRES-IR:
  - Solve time: 41.03s
  - Iterations: 4100
- **GMRES-IR attains the same minimum solve time as the switching strategy!**
  No need to choose a switching point!

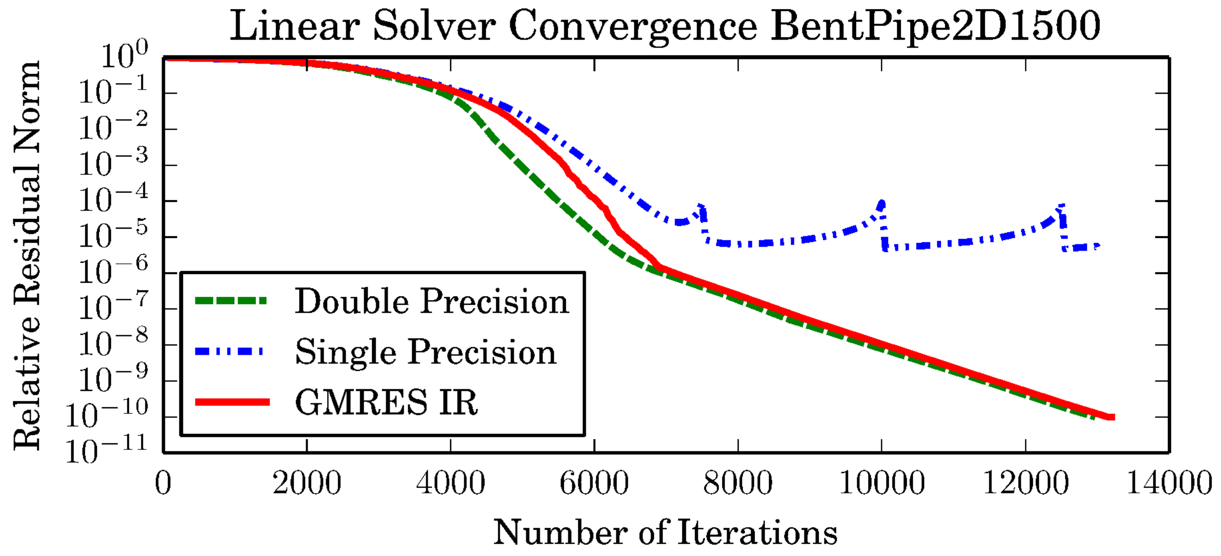


How does convergence of GMRES-IR compare to GMRES double?

Atmosmodj:
- SuiteSparse, cfd
- n = 1,270,432
- GMRES Double: 5.12s, 1740 iterations
- GMRES-IR: 3.78s, 1750 iterations

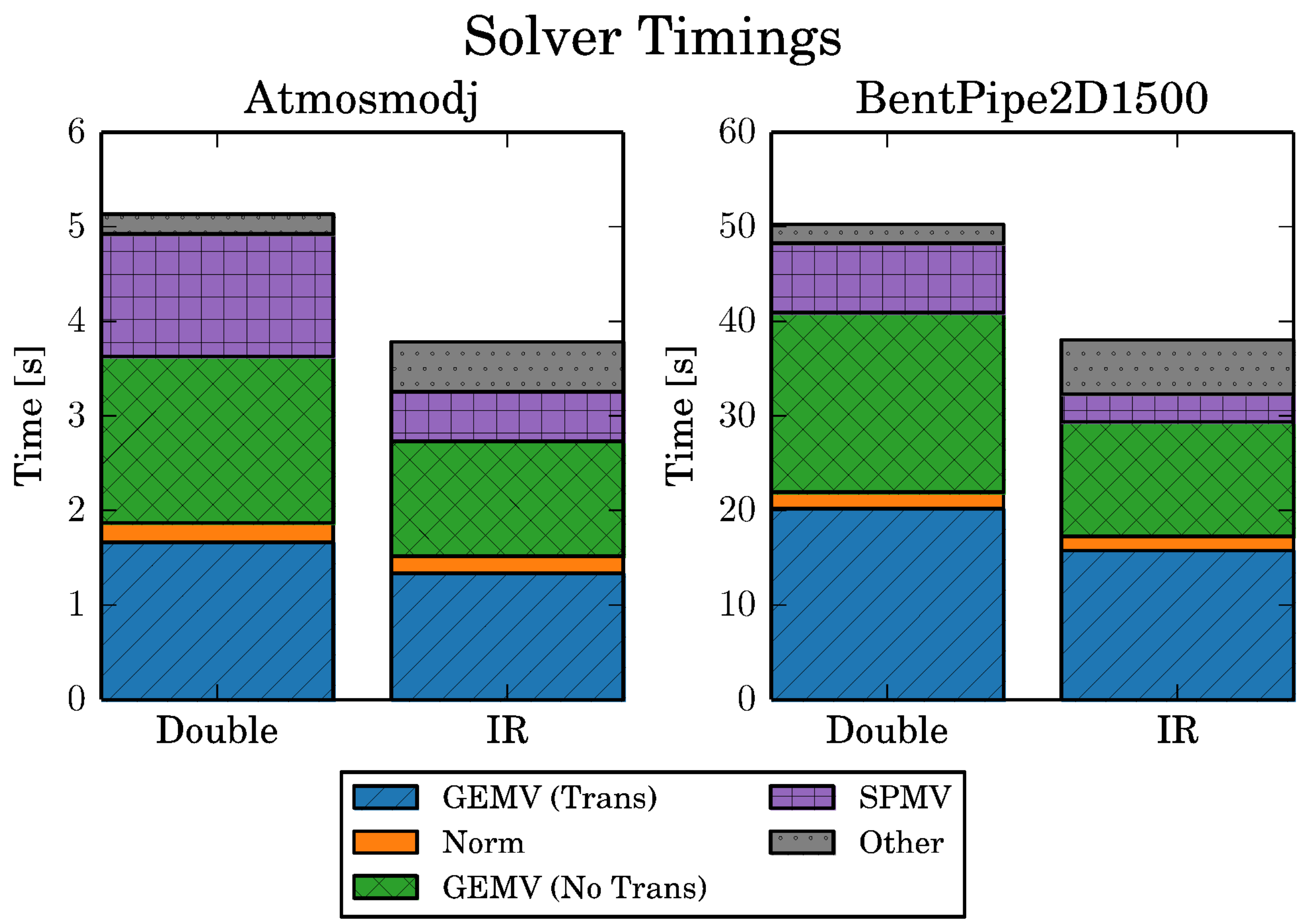


BentPipe2D1500:
- 2D convection-diffusion
- n = 2.25 million
- GMRES Double: 50.26s, 12,967 iterations
- GMRES-IR: 38.03s, 13,150 iterations



GMRES-IR convergence follows convergence of GMRES Double!

Kernel Speedup:



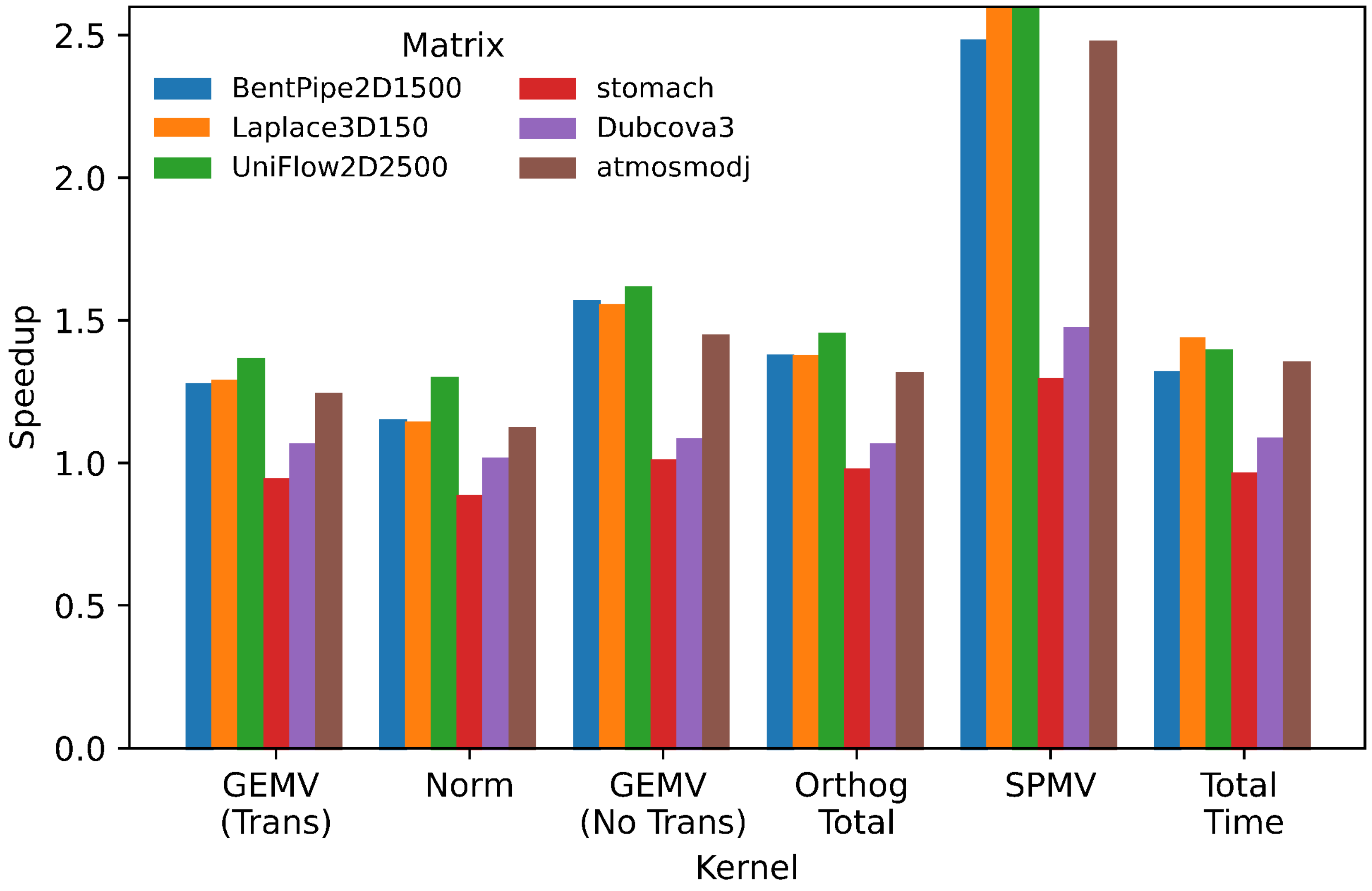
Atmosmodj:

- GMRES Double: 5.12s, 1740 iterations
- GMRES-IR: 3.78s, 1750 iterations

BentPipe2D1500:

- GMRES Double: 50.26s, 12,967 iterations
- GMRES-IR: 38.03s, 13,150 iterations

Kernel speedups compared with other matrices:



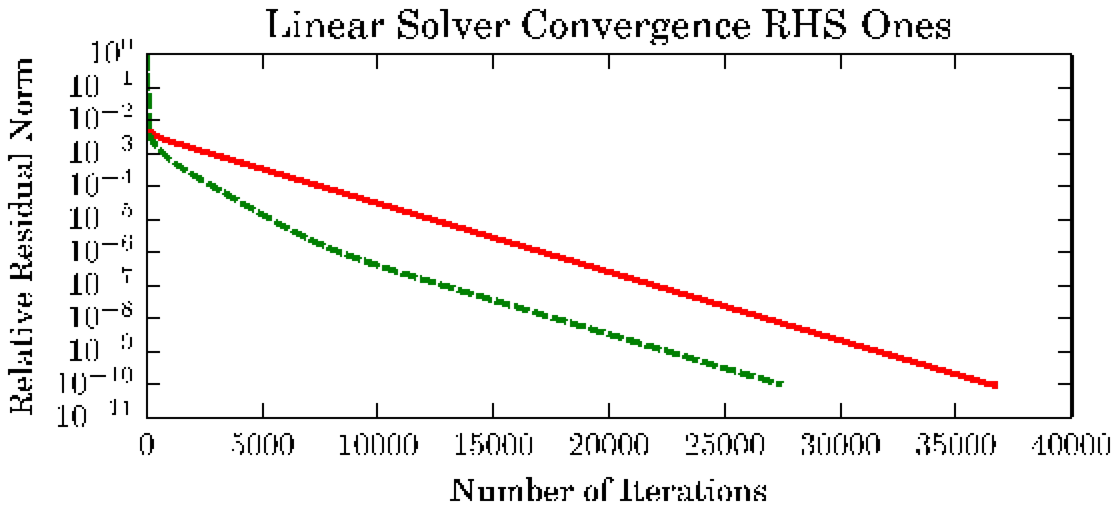
Does GMRES-IR convergence always follow GMRES double?

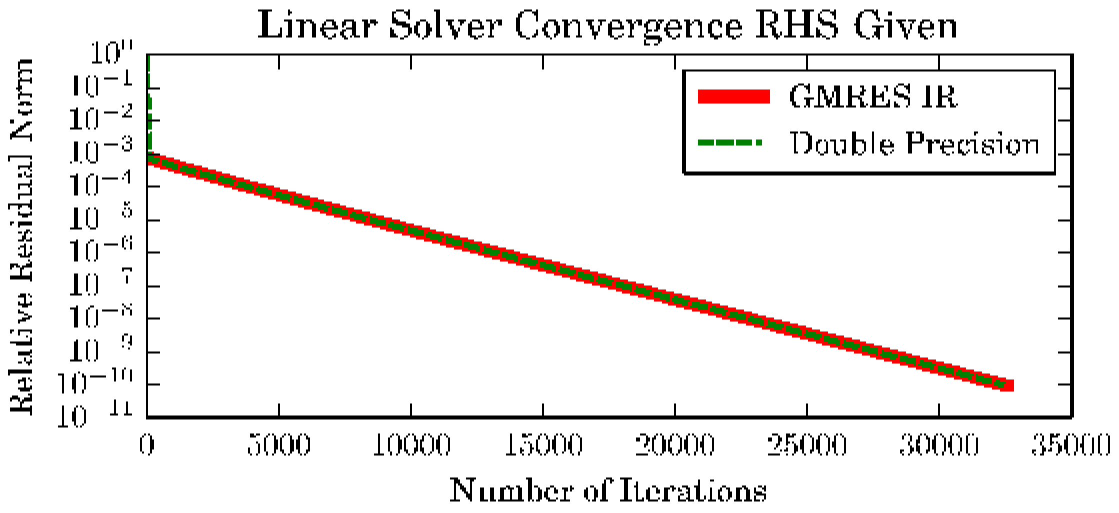
parabolic_fem:

- SuiteSparse, cfd
- n = 525,825

- TOP: right-hand side all ones
- BOTTOM: right-hand side from SuiteSparse

	Double		IR		
RHS Vec	**Time**	**Iters**	**Time**	**Iters**	**Speedup**
RHS_Ones	42.39	27,493	44.63	36,600	0.95
RHS_Given	50.04	32,470	39.16	32,500	1.28
RHS_Norm	54.02	34,960	41.72	35,000	1.29
RHS_Unif	51.98	33,625	41.64	34,150	1.25





A model for L2 cache use with low precision SpMV:

Suppose that A has w nonzero elements per row and n rows (so $nnz = w*n$).

A stored in CSR format with 2 vectors of size $w*n$:

Values of A: $\underline{A_{val}}$ Column indices: $\underline{colId}$ (Ignore vector of row ptrs)

Computing the first dot product of the SPMV:

$$\sum_{i=0}^{w-1} \underline{A_{val}}[i] * \underline{x}[\underline{colId[i]}].$$

Case: fp64 with no cache reuse (i.e. every element of x has to be read into cache every time needed):

$$n * w * [\underline{size(int)} + \underline{2 * size(double)}] = 20wn.$$

Case: fp32 with "perfect" cache reuse (i.e. any elements of x read into cache stay in cache until not needed):

$$n*w*[\underline{size(int)}+\underline{size(float)}]+n*\underline{size(float)} = (8w+4)n.$$

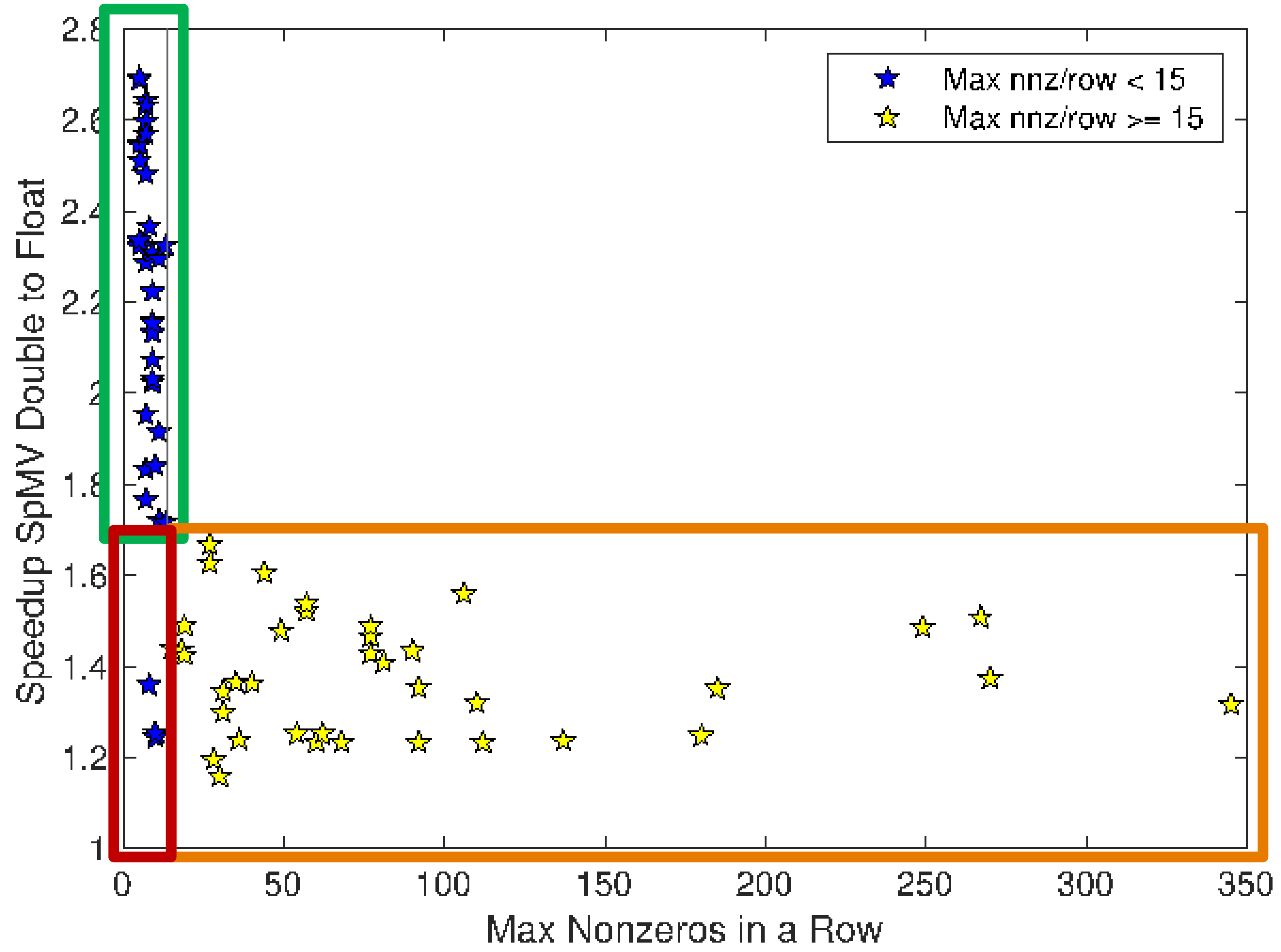
Expected speedup: $\dfrac{20wn}{(8w+4)n} = \dfrac{5w}{2w+1}. \longrightarrow$ 2.5 as w gets large.

** Thanks to Christian Trott and Luc-Berger Vergiat for help in creating this model!

SpMV Speedup vs Nonzero Structure of Matrix:

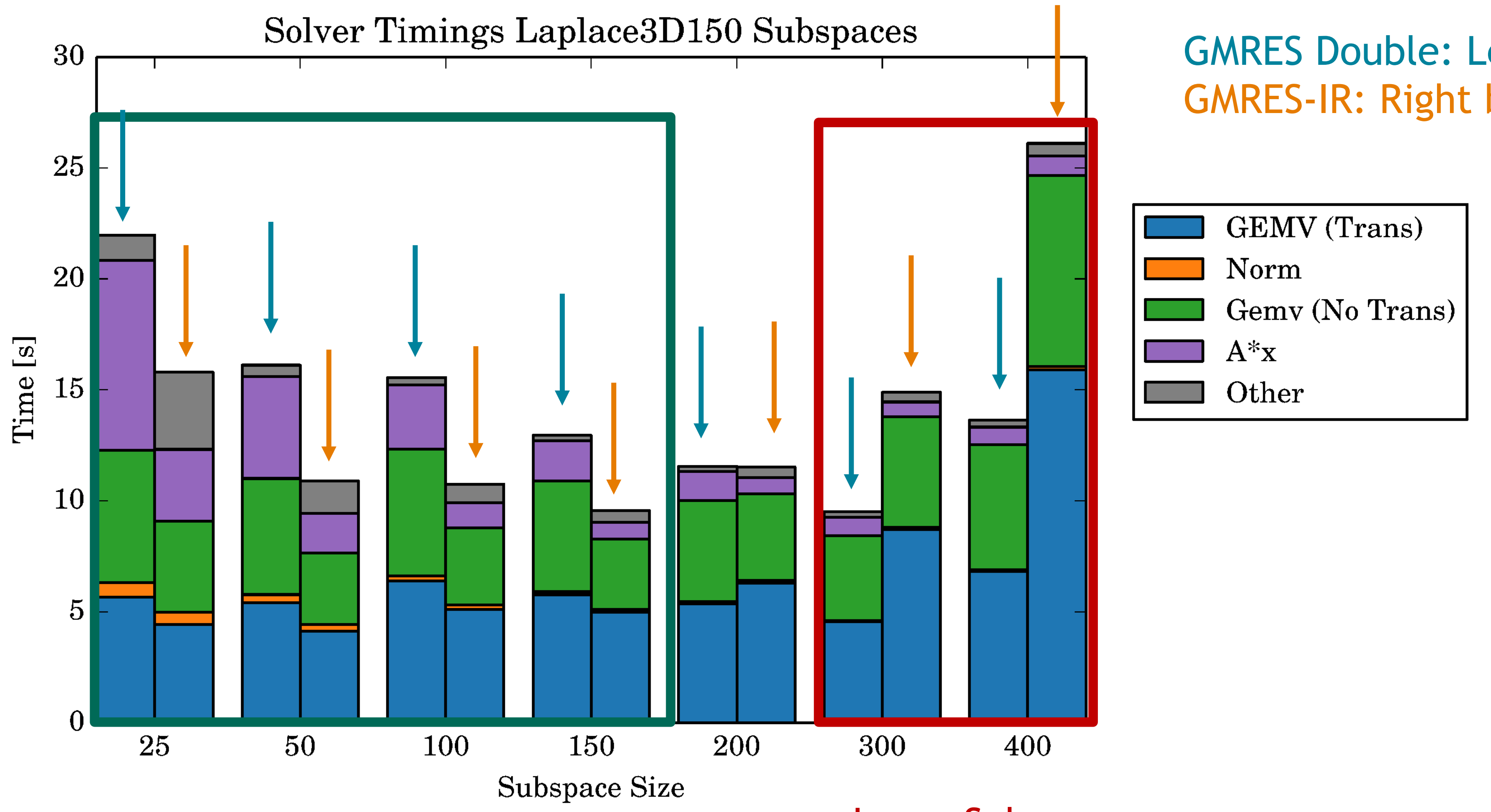
Very good speedup for matrices w/ small nnz/row.

Three smallest matrices in test set.



Large max nonzeros per row; low SpMV speedup

How does the Krylov subspace (restart) size affect solve time?



GMRES Double: Left bars

GMRES-IR: Right bars

Small Subspace:
GMRES-IR fastest

Large Subspace:
(Less practical in reality.)
GMRES double fastest

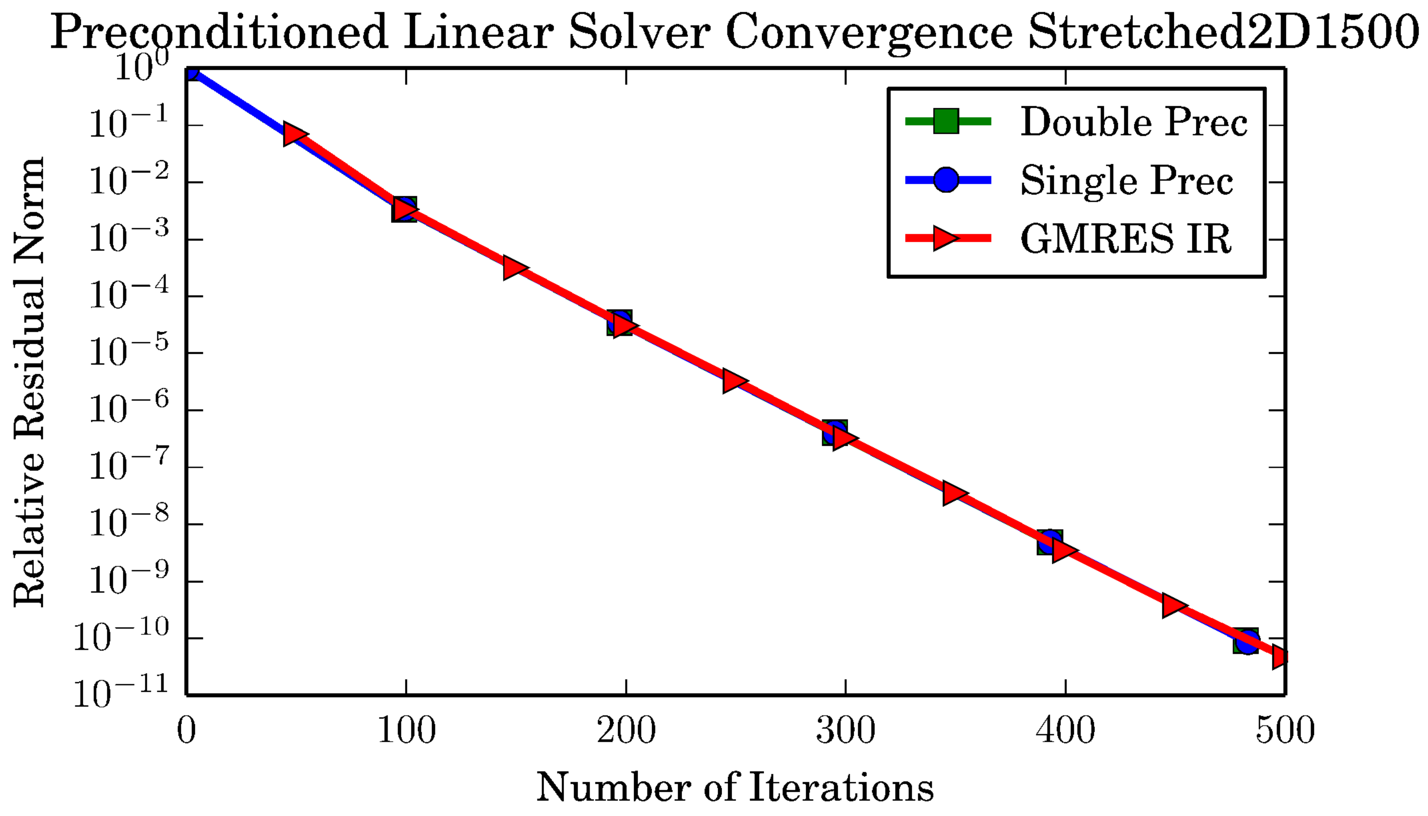
How does preconditioning affect GMRES-IR convergence?

Stretched2D1500:

- 2D Laplacian on Stretched Grid
- n = 2.25 million

Polynomial Preconditioner:

- GMRES Polynomial
- GMRES double: *double precision poly preconditioner*
- GMRES-IR: *single precision poly preconditioner*

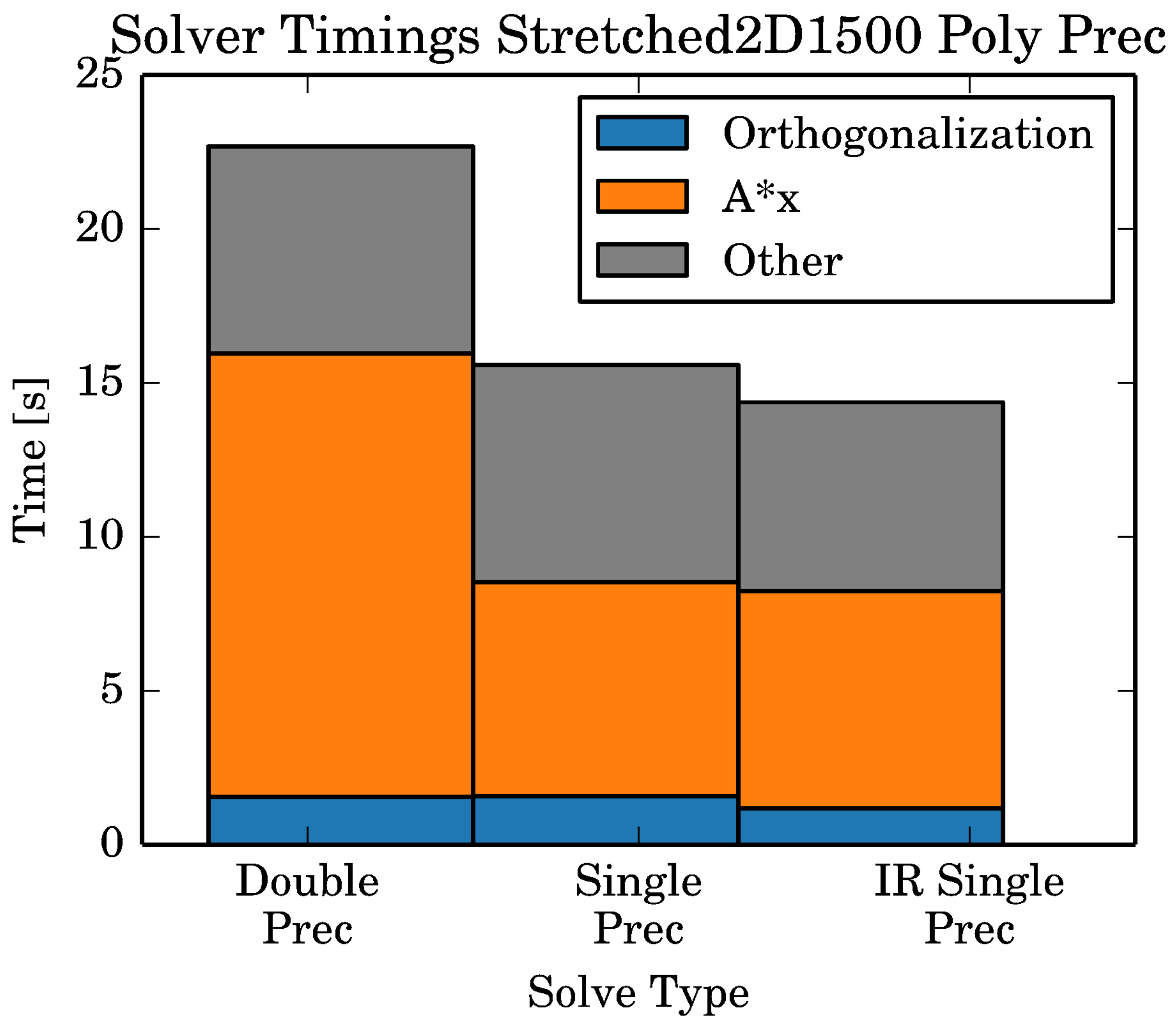


Preconditioned GMRES-IR convergence still follows convergence of GMRES Double!

Polynomial Preconditioning

<u>LEFT:</u> GMRES double w/ **fp64** polynomial preconditioner.
<u>MIDDLE</u>: GMRES double w/ **fp32** polynomial preconditioner.
<u>RIGHT:</u> GMRES-IR w/ **fp32** polynomial preconditioner.

Polynomial preconditioning shifts main expense to SpMV rather than dense orthogonalization kernels.



**For polynomial preconditioning details, see:
Jennifer Loe, Erik Boman, and Heidi Thornquist. *Polynomial Preconditioned GMRES in Trilinos: Practical Considerations for High-Performance Computing*

Results from SuiteSparse Matrices:

				Double		IR		
UF id	Matrix Name	N	prec	Time	Iters	Time	Iters	Speedup
2266	atmosmodj	1,270,432		5.12	1740	3.78	1750	1.35
2267	atmosmodl	1,489,752		1.61	446	1.23	450	1.31
1858	crashbasis	160,000		0.55	431	0.52	450	1.07
1849	Dubcova3	146,698		1.15	1131	1.05	1150	1.10
1852	FEM_3D_thermal2	147,900		0.84	775	0.80	800	1.05
1853	parabolic_fem	525,825		42.39	27493	44.63	36600	0.95
1367	SiO2	155,331		18.23	17385	16.86	17600	1.08
895	stomach	213,360		0.51	359	0.52	400	0.98
2259	thermomech_dM	204,316		0.27	88	0.27	100	1.00
894	lung2	109,460	j 1	0.46	206	0.49	250	0.94
1266	hood	220,542	j 42	13.98	5762	9.04	5000	1.55
805	cfd2	123,440	p 25	6.05	1092	4.55	1100	1.33
1431	filter3D	106,437	p 25	25.24	4449	18.12	4450	1.39
2649	Transport	1,602,111	p 25	8.35	339	8.73	450	0.96
	BentPipe2D1500	2,250,000		50.26	12967	38.03	13150	1.32
	Laplace3D150	3,375,000		16.93	2387	11.75	2400	1.44
	UniFlow2D2500	6,250,000		29.62	2905	21.17	3000	1.40
	Stretched2D1500	2,250,000	p 40	22.66	482	14.37	500	1.58

*prec column:
p = polynomial prec w/ degree
j = Jacobi prec w/ block size

Example PDE stencil problems from previous slides.

Results from SuiteSparse Matrices:

				Double		IR		
UF id	Matrix Name	N	prec	Time	Iters	Time	Iters	Speedup
2266	atmosmodj	1,270,432		5.12	1740	3.78	1750	1.35
2267	atmosmodl	1,489,752		1.61	446	1.23	450	1.31
1858	crashbasis	160,000		0.55	431	0.52	450	1.07
1849	Dubcova3	146,698		1.15	1131	1.05	1150	1.10
1852	FEM_3D_thermal2	147,900		0.84	775	0.80	800	1.05
1853	parabolic_fem	525,825		42.39	27493	44.63	36600	0.95
1367	SiO2	155,331		18.23	17385	16.86	17600	1.08
895	stomach	213,360		0.51	359	0.52	400	0.98
2259	thermomech_dM	204,316		0.27	88	0.27	100	1.00
894	lung2	109,460	j 1	0.46	206	0.49	250	0.94
1266	hood	220,542	j 42	13.98	5762	9.04	5000	1.55
805	cfd2	123,440	p 25	6.05	1092	4.55	1100	1.33
1431	filter3D	106,437	p 25	25.24	4449	18.12	4450	1.39
2649	Transport	1,602,111	p 25	8.35	339	8.73	450	0.96
	BentPipe2D1500	2,250,000		50.26	12967	38.03	13150	1.32
	Laplace3D150	3,375,000		16.93	2387	11.75	2400	1.44
	UniFlow2D2500	6,250,000		29.62	2905	21.17	3000	1.40
	Stretched2D1500	2,250,000	p 40	22.66	482	14.37	500	1.58

*prec column:
p = polynomial prec w/ degree
j = Jacobi prec w/ block size

Quickly converging problems; not much room for speedup from GMRES-IR.

Results from SuiteSparse Matrices:

				Double		IR		
UF id	**Matrix Name**	**N**	**prec**	**Time**	**Iters**	**Time**	**Iters**	**Speedup**
2266	atmosmodj	1,270,432		5.12	1740	3.78	1750	1.35
2267	atmosmodl	1,489,752		1.61	446	1.23	450	1.31
1858	crashbasis	160,000		0.55	431	0.52	450	1.07
1849	Dubcova3	146,698		1.15	1131	1.05	1150	1.10
1852	FEM_3D_thermal2	147,900		0.84	775	0.80	800	1.05
1853	parabolic_fem	525,825		42.39	27493	44.63	36600	0.95
1367	SiO2	155,331		18.23	17385	16.86	17600	1.08
895	stomach	213,360		0.51	359	0.52	400	0.98
2259	thermomech_dM	204,316		0.27	88	0.27	100	1.00
894	lung2	109,460	j 1	0.46	206	0.49	250	0.94
1266	hood	220,542	j 42	13.98	5762	9.04	5000	1.55
805	cfd2	123,440	p 25	6.05	1092	4.55	1100	1.33
1431	filter3D	106,437	p 25	25.24	4449	18.12	4450	1.39
2649	Transport	1,602,111	p 25	8.35	339	8.73	450	0.96
	BentPipe2D1500	2,250,000		50.26	12967	38.03	13150	1.32
	Laplace3D150	3,375,000		16.93	2387	11.75	2400	1.44
	UniFlow2D2500	6,250,000		29.62	2905	21.17	3000	1.40
	Stretched2D1500	2,250,000	p 40	22.66	482	14.37	500	1.58

*prec column:
p = polynomial prec w/ degree
j = Jacobi prec w/ block size

Right-hand side made more difficult convergence.

Results from SuiteSparse Matrices:

				Double		IR		
UF id	Matrix Name	N	prec	Time	Iters	Time	Iters	Speedup
2266	atmosmodj	1,270,432		5.12	1740	3.78	1750	1.35
2267	atmosmodl	1,489,752		1.61	446	1.23	450	1.31
1858	crashbasis	160,000		0.55	431	0.52	450	1.07
1849	Dubcova3	146,698		1.15	1131	1.05	1150	1.10
1852	FEM_3D_thermal2	147,900		0.84	775	0.80	800	1.05
1853	parabolic_fem	525,825		42.39	27493	44.63	36600	0.95
1367	SiO2	155,331		18.23	17385	16.86	17600	1.08
895	stomach	213,360		0.51	359	0.52	400	0.98
2259	thermomech_dM	204,316		0.27	88	0.27	100	1.00
894	lung2	109,460	j 1	0.46	206	0.49	250	0.94
1266	hood	220,542	j 42	13.98	5762	9.04	5000	1.55
805	cfd2	123,440	p 25	6.05	1092	4.55	1100	1.33
1431	filter3D	106,437	p 25	25.24	4449	18.12	4450	1.39
2649	Transport	1,602,111	p 25	8.35	339	8.73	450	0.96
	BentPipe2D1500	2,250,000		50.26	12967	38.03	13150	1.32
	Laplace3D150	3,375,000		16.93	2387	11.75	2400	1.44
	UniFlow2D2500	6,250,000		29.62	2905	21.17	3000	1.40
	Stretched2D1500	2,250,000	p 40	22.66	482	14.37	500	1.58

*prec column:
p = polynomial prec w/ degree
j = Jacobi prec w/ block size

Very good speedup for SuiteSparse test problems!

Future Work:

- Implement GMRES-IR in Tpetra solvers in Belos package of Trilinos
- Make GMRES (double) with single precision preconditioning available in Tpetra Belos solvers.
- Incorporate half precision computations (fp16 and bfloat16).
- Test performance on other (non-NVIDIA) GPU architectures- AMD and Intel.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.