FREEFEM++ DISTRIBUTED SOLVERS: STATUS AND FUTURE

Pierre Jolivet, IRIT–CNRS

8th FreeFem++ days
December 8

CURRENT DISTRIBUTED SOLVERS

- matrix assembly using domain decomposition
- linear solvers using direct methods, iterative solvers, preconditioners...

Distributed finite elements

- matrix assembly using domain decomposition
- linear solvers using direct methods, iterative solvers, preconditioners...

In FreeFem++, meshes are decomposed either via:

- `load "metis"` or
- `load "scotch"`

Interface with ParMETIS or PT-SCOTCH may come soon

There is no parallelism inside the FreeFem++ kernel:

- no ghost elements
- no distributed meshes
- no distributed matrices

There is no parallelism inside the FreeFem++ kernel:

- no ghost elements
- no distributed meshes
- no distributed matrices

Users can access MPI via extra keywords in FreeFem++-mpi

Cons: parallelism is explicit (and must be hand coded)
Pros: users know what they are doing

In the folder `examples++-hpddm`:

- scalable matrix assembly
- scalable linear solvers

In the folder `examples++-hpddm`:

- scalable matrix assembly
- scalable linear solvers

Two linear algebra backends:

- PETSc
- HPDDM

LIST OF CURRENT (AS OF 07/12/2016) EXAMPLES

- diffusion-2d.edp
- diffusion-2d-PETSc.edp
- diffusion-2d-substructuring.edp
- diffusion-2d-substructuring-PETSc.edp
- diffusion-3d.edp
- diffusion-3d-PETSc.edp
- elasticity-2d.edp
- elasticity-2d-PETSc.edp
- elasticity-2d-substructuring.edp
- elasticity-3d.edp
- elasticity-3d-PETSc.edp
- heat-2d.edp
- heat-2d-PETSc.edp
- heat-3d.edp
- helmholtz-2d.edp
- helmholtz-2d-PETSc.edp
- maxwell-3d.edp
- stokes-2d.edp
- stokes-2d-PETSc.edp
- stokes-3d.edp
- stokes-3d-PETSc.edp

I will add a nonsymmetric problem

LIST OF CURRENT (AS OF 07/12/2016) EXAMPLES

- diffusion-2d.edp
- diffusion-2d-PETSc.edp
- diffusion-2d-substructuring.edp
- diffusion-2d-substructuring-PETSc.edp
- diffusion-3d.edp
- diffusion-3d-PETSc.edp
- elasticity-2d.edp
- elasticity-2d-PETSc.edp
- elasticity-2d-substructuring.edp
- elasticity-3d.edp
- elasticity-3d-PETSc.edp
- heat-2d.edp
- heat-2d-PETSc.edp
- heat-3d.edp
- helmholtz-2d.edp
- helmholtz-2d-PETSc.edp
- maxwell-3d.edp
- stokes-2d.edp
- stokes-2d-PETSc.edp
- stokes-3d.edp
- stokes-3d-PETSc.edp

I will add a nonsymmetric problem

LIST OF CURRENT (AS OF 07/12/2016) EXAMPLES

- diffusion-2d.edp
- diffusion-2d-PETSc.edp
- diffusion-2d-substructuring.edp
- diffusion-2d-substructuring-PETSc.edp
- diffusion-3d.edp
- diffusion-3d-PETSc.edp
- elasticity-2d.edp
- elasticity-2d-PETSc.edp
- elasticity-2d-substructuring.edp
- elasticity-3d.edp
- elasticity-3d-PETSc.edp
- heat-2d.edp
- heat-2d-PETSc.edp
- heat-3d.edp
- helmholtz-2d.edp
- helmholtz-2d-PETSc.edp
- maxwell-3d.edp
- stokes-2d.edp
- stokes-2d-PETSc.edp
- stokes-3d.edp
- stokes-3d-PETSc.edp

I will add a nonsymmetric problem

List of current (as of 07/12/2016) examples

- diffusion-2d.edp
- diffusion-2d-PETSc.edp
- diffusion-2d-substructuring.edp
- diffusion-2d-substructuring-PETSc.edp
- diffusion-3d.edp
- diffusion-3d-PETSc.edp
- elasticity-2d.edp
- elasticity-2d-PETSc.edp
- elasticity-2d-substructuring.edp
- elasticity-3d.edp
- elasticity-3d-PETSc.edp
- heat-2d.edp
- heat-2d-PETSc.edp
- heat-3d.edp
- helmholtz-2d.edp
- helmholtz-2d-PETSc.edp
- maxwell-3d.edp
- stokes-2d.edp
- stokes-2d-PETSc.edp
- stokes-3d.edp
- stokes-3d-PETSc.edp

I will add a nonsymmetric problem

LIST OF CURRENT (AS OF 07/12/2016) EXAMPLES

- diffusion-2d.edp
- diffusion-2d-PETSc.edp
- diffusion-2d-substructuring.edp
- diffusion-2d-substructuring-PETSc.edp
- diffusion-3d.edp
- diffusion-3d-PETSc.edp
- elasticity-2d.edp
- elasticity-2d-PETSc.edp
- elasticity-2d-substructuring.edp
- elasticity-3d.edp
- elasticity-3d-PETSc.edp
- heat-2d.edp
- heat-2d-PETSc.edp
- heat-3d.edp
- helmholtz-2d.edp
- helmholtz-2d-PETSc.edp
- maxwell-3d.edp
- stokes-2d.edp
- stokes-2d-PETSc.edp
- stokes-3d.edp
- stokes-3d-PETSc.edp

I will add a nonsymmetric problem

All command line options are parsed by PETSc or HPDDM

Additional types:

- `dmatrix` (PETSc backend)
- `[z|d]schwarz` (overlapping Schwarz backend)
- `[z|d]bdd` (BDD backend)
- `[z|d]feti` (FETI backend)

All command line options are parsed by PETSc or HPDDM

Additional types:

- `dmatrix` (PETSc backend)
- `[z|d]schwarz` (overlapping Schwarz backend)
- `[z|d]bdd` (BDD backend)
- `[z|d]feti` (FETI backend)

All types support the operations:

- `set(A, sparams = "...");`
- `y = A * x;` // be careful with TGVs!
- `y =` A^{-1} `* x;`

Most important options to keep in mind:

- `-help`
- `-ksp_type`
- `-pc_type`

Only the linear solvers are interfaced (no nonlinear solver or time integrator)

Most important options to keep in mind:

- `-help`
- `-ksp_type`
- `-pc_type`

Only the linear solvers are interfaced (no nonlinear solver or time integrator)

Be careful when mixing complex and real libraries!

NEW INTERFACES

- PCBDDC preconditioner by Stefano Zampini

NEW INTERFACES

- PCBDDC preconditioner by Stefano Zampini
- block matrices:

$$A = \begin{bmatrix} A_{11} & 0 & A_{13} & 0 \\ 0 & A_{22} & A_{23} & 0 \\ A_{31} & 0 & A_{33} & 0 \\ 0 & 0 & 0 & A_{44} \end{bmatrix}$$

On process 1, `int[int] Ji(2) = [0, 2];`
`matrix[int] Ai(2); Ai(0)=`A_{11}`; Ai(1)=`A_{22}`;`

$$\ldots$$

Then, `dmatrix A(Ai, columns = Ji);`

DD preconditioners, with a recent focus on:

- block iterative methods
- recycled iterative methods

Most important options to keep in mind:

- `-hpddm_help`
- `-hpddm_krylov_method`
- `-hpddm_geneo_nu`

- many new options such as `-hpddm_recycle`
- solve a system with multiple RHS simultaneously
- enlarged Krylov subspace methods

```
Vh u; int p = 32;
real[int] blockRhs(p * u[].n);
real[int] blockSol(p * u[].n);
blockSol = A^-1 * blockRhs;
```

MORE ON BLOCK ITERATIVE METHODS AND RECYCLING [Jolivet and Tournier 2016]

Subspace recycling

Keep information between restart or when solving sequences of linear systems:

$$A_i x_i = b_i \qquad \forall i = 1, 2, \ldots$$

Subspace recycling

Keep information between restart or when solving sequences of linear systems:

$$A_i x_i = b_i \qquad \forall i = 1, 2, \ldots$$

Block methods

Treat multiple right-hand sides simultaneously for faster convergence:

$$AX = B \qquad B \in \mathbb{K}^{n \times p}$$

STATE OF THE ART II

Available options

- *hypre*, DUNE, PARALUTION, SciPy: nothing
- PETSc: Loose GMRES and Deflated GMRES
- Trilinos (Belos): Block GMRES, Block GCRO-DR

Available options

- *hypre*, DUNE, PARALUTION, SciPy: nothing
- PETSc: Loose GMRES and Deflated GMRES
- Trilinos (Belos): Block GMRES, Block GCRO-DR

Why not use Belos?

- no support for variable preconditioning
- no trivial support for languages other than C++

CONTRIBUTIONS

- implementation of (pseudo-)Block GMRES/GCRO-DR
- support for left/right/variable preconditioning
- large-scale results for three different physics

CONTRIBUTIONS

- implementation of (pseudo-)Block GMRES/GCRO-DR
- support for left/right/variable preconditioning
- large-scale results for three different physics

HPDDM

- open-source, `https://github.com/hpddm/hpddm`
- usable in C++, C, Python, or Fortran
- also has (pseudo-)Block CG and Breakdown-Free BCG

SUBSPACE RECYCLING

GCRO-DR

Generalized Conjugate Residual method with inner Orthogonalization and Deflated Restarting

Proposed by [Parks et al. 2006]

Closely related to GMRES-DR by [Morgan 2002]

Proposed by [Parks et al. 2006]

Closely related to GMRES-DR by [Morgan 2002]

Main idea

1. end of GMRES cycle: compute Ritz eigenpairs
2. next restart: use 1. to generate k vectors for Arnoldi basis
3. perform extra orthogonalizations with k vectors

GCRO-DR

Generalized Conjugate Residual method with inner Orthogonalization and Deflated Restarting

Proposed by [Parks et al. 2006]

Closely related to GMRES-DR by [Morgan 2002]

Main idea

1. end of GMRES cycle: compute Ritz eigenpairs
2. next restart: use 1. to generate k vectors for Arnoldi basis
3. perform extra orthogonalizations with k vectors

Overhead:

- persistent storage between cycles/solves
- one additional synchronization per cycle
- small dense (generalized) eigenvalue problem

BLOCK ITERATIVE METHODS

WHY USE BLOCK METHODS?

Numerical aspects

enlarged Krylov subspace $\Longrightarrow$ faster convergence

Performance

- higher arithmetic intensity
- fewer synchronizations with more data

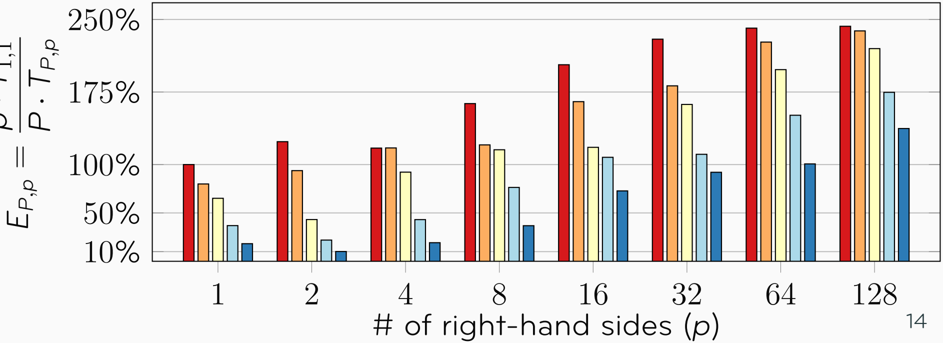
WHY USE BLOCK METHODS?

Numerical aspects

enlarged Krylov subspace $\implies$ faster convergence

Performance

- higher arithmetic intensity
- fewer synchronizations with more data



Block Arnoldi

- block orthogonalization
- tall-and-skinny QR (default to CholQR $V^H V = LL^H$)

Block Arnoldi

- block orthogonalization
- tall-and-skinny QR (default to CholQR $V^H V = LL^H$)

About CholQR:

- $V = QR$ with $R = L^H$, $Q = L^{-H} V$
- BLAS 3
- one reduction [Stathopoulos and Wu 2002]
- can rank-reveal (`?pstrf`)

Block Arnoldi

- block orthogonalization
- tall-and-skinny QR (default to CholQR $V^H V = LL^H$)

About CholQR:

- $V = QR$ with $R = L^H$, $Q = L^{-H}V$
- BLAS 3
- one reduction [Stathopoulos and Wu 2002]
- can rank-reveal (`?pstrf`)

Pseudo-block methods

p subspaces, computation and communication steps fused

APPLICATIONS AND NUMERICAL RESULTS

Two examples from the PETSc distribution:

1. ex32 (Poisson's equation)
2. ex56 (linear elasticity)

Geometric Algebraic MG preconditioner [Adams et al. 2004]

Two examples from the PETSc distribution:

1. ex32 (Poisson's equation)
2. ex56 (linear elasticity)

Geometric Algebraic MG preconditioner [Adams et al. 2004]

Nonlinear smoothers

$\Longrightarrow$ FGCRO-DR is mandatory

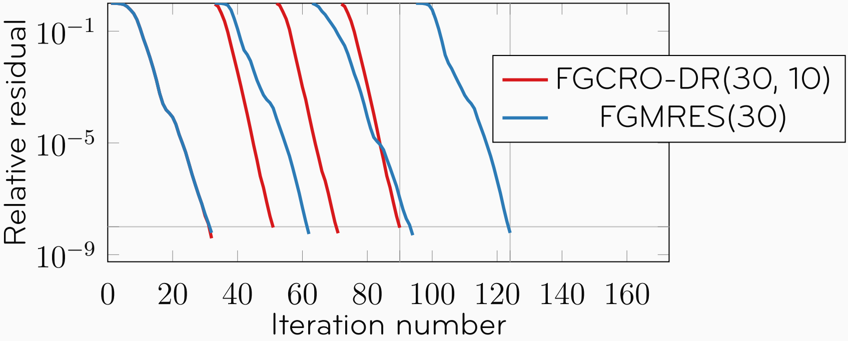
```
mpirun -np 8192 ./ex32 -da_grid_x 4210 -da_refine 2
  -da_grid_y 4210 -ksp_rtol 1e-8 -pc_type gamg
  -pc_gamg_threshold 0.0725 -pc_gamg_square_graph 2
  -ksp_type fgmres -mg_levels_ksp_type gmres
  -mg_levels_ksp_max_it 3
```

- assemble a single linear system/preconditioner
- solve a sequence with multiple RHSs:

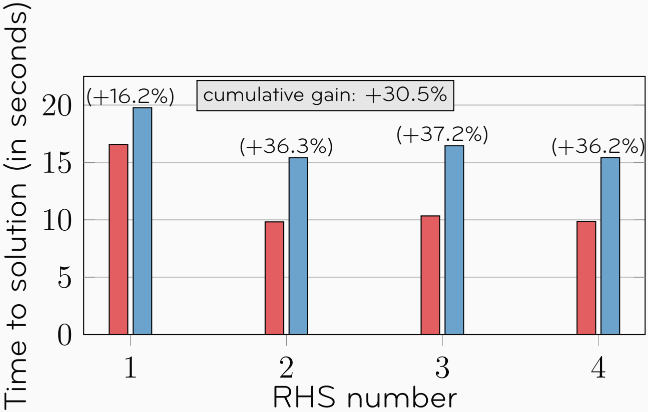
$$Ax_i = b_i \qquad \forall i \in [\![1, 4]\!]$$

For Poisson's equation I

```
mpirun -np 8192 ./ex32 -da_grid_x 4210 -da_refine 2
  -da_grid_y 4210 -ksp_rtol 1e-8 -pc_type gamg
  -pc_gamg_threshold 0.0725 -pc_gamg_square_graph 2
  -ksp_type fgmres -mg_levels_ksp_type gmres
  -mg_levels_ksp_max_it 3
```

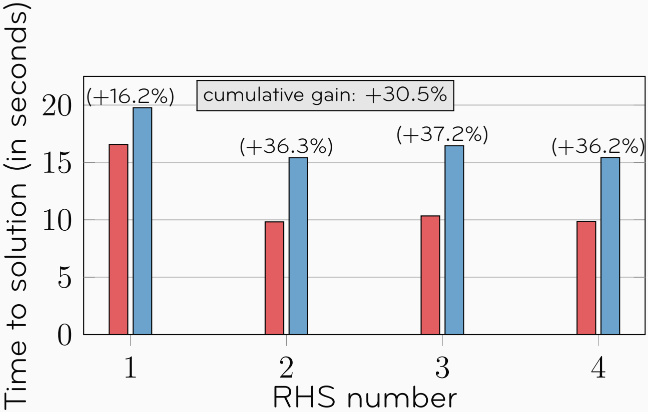


For Poisson's equation II



RECYCLING

For Poisson's equation II



recycling $\Longrightarrow$ relax preconditioner setup parameters

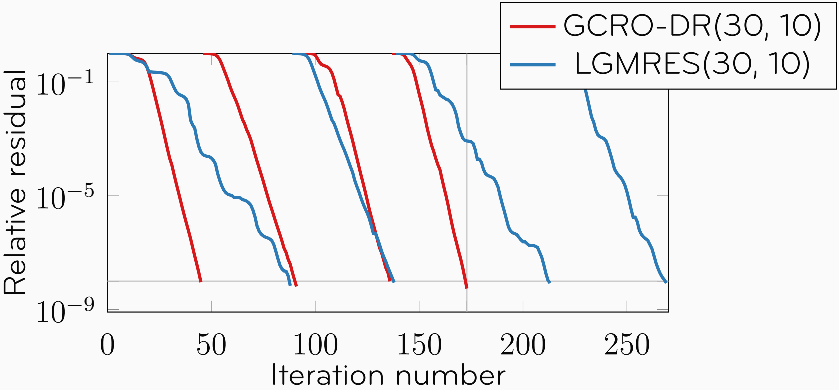
Comparison with Loose GMRES by [Baker et al. 2005]

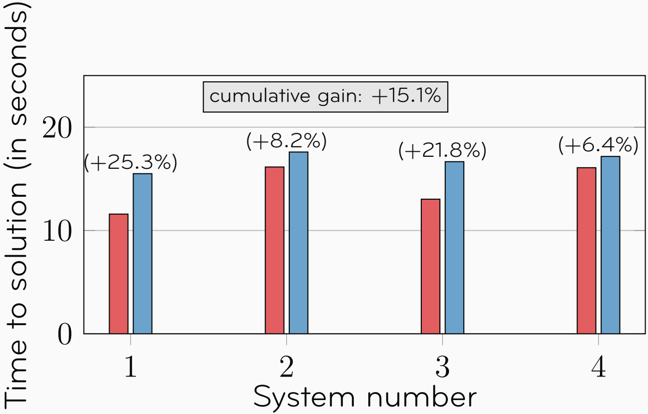
```
mpirun -np 8000 ./ex56 -ne 399 -ksp_rtol 1e-8
  -ksp_type lgmres -ksp_pc_side right -pc_type gamg
  -ksp_lgmres_augment 10
```

- small, moving inclusion (high contrast in E)
- assemble multiple linear systems/preconditioners
- PETSc doesn't implement flexible LGMRES

Comparison with Loose GMRES by [Baker et al. 2005]

```
mpirun -np 8000 ./ex56 -ne 399 -ksp_rtol 1e-8
  -ksp_type lgmres -ksp_pc_side right -pc_type gamg
  -ksp_lgmres_augment 10
```




cumulative gain: +15.1%
(+25.3%)
(+8.2%)
(+21.8%)
(+6.4%)
Time to solution (in seconds)
0
10
20
1
2
3
4
System number

$$\nabla \times (\nabla \times \mathbf{E}) - \mu_0 \left(\cancel{\omega^2 \varepsilon} + \mathrm{i} \omega \sigma \right) \mathbf{E} = 0$$

AMS and MueLu:

1. only support eddy current formulation
2. are not trivial to use with high-order edge elements

AMS cannot deal with multiple RHSs

$$\nabla \times (\nabla \times \mathbf{E}) - \mu_0 \left(\cancel{\omega^2 \varepsilon} + \mathrm{i}\omega\sigma \right) \mathbf{E} = 0$$

AMS and MueLu:

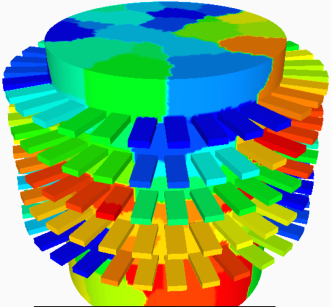
1. only support eddy current formulation
2. are not trivial to use with high-order edge elements

AMS cannot deal with multiple RHSs

$$\mathcal{M}^{-1}_{\text{ORAS}} = \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i,$$

cf. [Gander 2006]

- B_i^{-1} may be applied to multiple vectors at once
- B_i makes "more sense"?



Credits: P.-H. Tournier

$$\nabla \times (\nabla \times \mathbf{E}) - \mu_0 \left(\cancel{\omega^2 \varepsilon} + \mathrm{i}\omega\sigma \right) \mathbf{E} = 0$$

AMS and MueLu:

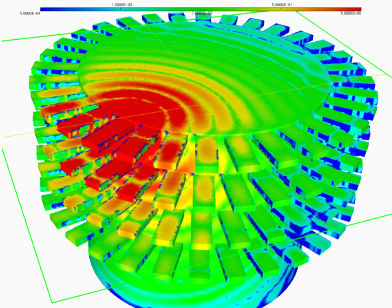
1. only support eddy current formulation
2. are not trivial to use with high-order edge elements

AMS cannot deal with multiple RHSs

$$\mathcal{M}_{\text{ORAS}}^{-1} = \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i,$$

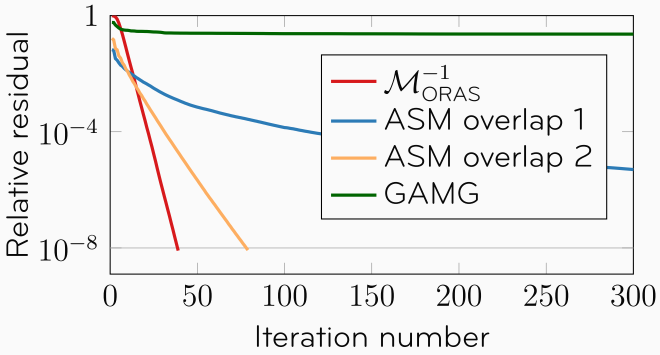
cf. [Gander 2006]

- B_i^{-1} may be applied to multiple vectors at once
- B_i makes "more sense"?



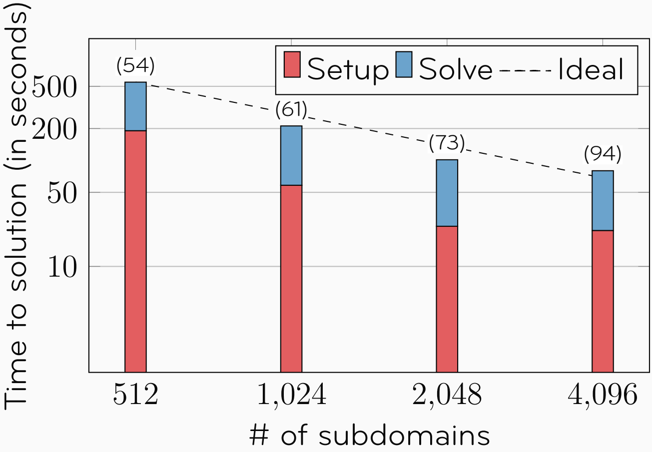
Credits: P.-H. Tournier

Robustness of the preconditioner



- 50 million double-precision complex unknowns
- degree 3 edge elements
- 512 subdomains, 1 thread per subdomain

Scalability of the preconditioner



- 119 million double-precision complex unknowns
- degree 2 edge elements

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1				
GCRO-DR	1				

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2
pseudo-BGCRO-DR	8	1,357.8	1,508	377	2.3
pseudo-BGCRO-DR	32	1,376.1	469	–	2.2

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2
pseudo-BGCRO-DR	8	1,357.8	1,508	377	2.3
pseudo-BGCRO-DR	32	1,376.1	469	–	2.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2
pseudo-BGCRO-DR	8	1,357.8	1,508	377	2.3
pseudo-BGCRO-DR	32	1,376.1	469	–	2.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain
- alternative #1 to #8 $\implies$ 158$\times$ fewer iterations

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2
pseudo-BGCRO-DR	8	1,357.8	1,508	377	2.3
pseudo-BGCRO-DR	32	1,376.1	469	–	2.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain
- alternative #1 to #8 $\implies$ 158$\times$ fewer iterations
- GCRO-DR always performs fewer iterations than GMRES

alternative	p	solve	# of it.	per RHS	eff.
GMRES	1	3,078.4	20,068	627	–
GCRO-DR	1	1,836.9	10,701	334	1.7
pseudo-BGMRES	32	1,577.9	653	–	2.0
BGMRES	32	724.8	158	–	4.2
pseudo-BGCRO-DR	8	1,357.8	1,508	377	2.3
pseudo-BGCRO-DR	32	1,376.1	469	–	2.2
BGCRO-DR	8	677.6	524	131	4.5
BGCRO-DR	32	992.3	127	–	3.1

- $(m, k) = (50, 10)$ for solving 32 RHSs
- 2,048 subdomains and 2 threads per subdomain
- alternative #1 to #8 $\implies$ 158× fewer iterations
- GCRO-DR always performs fewer iterations than GMRES
- working on all 32 RHSs is costly (#5/#7 vs. #6/#8)

CONCLUSION

Summary:

- you should use FreeFem++-mpi
- many examples to start from

Future work:

- other fancy preconditioners
- different applications

Summary:

- you should use FreeFem++-mpi
- many examples to start from

Future work:

- other fancy preconditioners
- different applications

Thank you!

Adams, Mark F., Harun H. Bayraktar, Tony M. Keaveny, and Panayiotis Papadopoulos (2004). "Ultrascalable Implicit Finite Element Analyses in Solid Mechanics With Over a Half a Billion Degrees of Freedom". In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC04*. IEEE Computer Society.

Baker, Allison H., Elizabeth R. Jessup, and Thomas Manteuffel (2005). "A Technique for Accelerating the Convergence of Restarted GMRES". In: *SIAM Journal on Matrix Analysis and Applications* 26.4, pp. 962–984.

Gander, Martin J. (2006). "Optimized Schwarz Methods". In: *SIAM Journal on Numerical Analysis* 44.2, pp. 699–731.

Jolivet, Pierre and Pierre-Henri Tournier (2016). "Block Iterative Methods and Recycling for Improved Scalability of Linear Solvers". In: *Proceedings of the 2016 International Conference on High Performance Computing, Networking, Storage and Analysis*. SC16. IEEE.

Morgan, Ronald B. (2002). "GMRES with Deflated Restarting". In: *SIAM Journal on Scientific Computing* 24.1, pp. 20–37.

Parks, Michael L., Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti (2006). "Recycling Krylov Subspaces for Sequences of Linear Systems". In: *SIAM Journal on Scientific Computing* 28.5, pp. 1651–1674.

Stathopoulos, Andreas and Kesheng Wu (2002). "A Block Orthogonalization Procedure with Constant Synchronization Requirements". In: *SIAM Journal on Scientific Computing* 23.6, pp. 2165–2182.