UK Atomic Energy Authority

UKAEA-STEP-PR(23)02

Julita Inca, Andrew Davis, Aleksander Dubas

Benchmarking preconditioners

Benchmarking preconditioners

Julita Inca, Andrew Davis, Aleksander Dubas

This is a preprint of a paper submitted for publication in
Physics of Plasmas

UK Atomic Energy Authority

Plasma Physics and Controlled Fusion, May 2021

Benchmarking Preconditioners

Julita Inca Chiroque[1], Andrew Davis[1] and Aleksander Dubas[1]

[1] *United Kingdom Atomic Energy Authority, Culham Centre for Fusion Energy, Culham, Abingdon, UK*

Reception date of the manuscript: dd/mm/aaaa
Acceptance date of the manuscript: dd/mm/aaaa
Publication date: dd/mm/aaaa

Abstract— Preconditioners are meant to improve both, the efficiency and robustness of iterative techniques while solving very large linear systems on a Krylov subspace. However, determining which preconditioner is suitable to be applied on a certain multiphysic simulation requires a combination of knowledge of preconditioning matrices techniques, types of matrices, Krylov subspaces, iterative methods, among other Linear Algebra's foundation. The present work provides a benchmark of the most popular preconditioners available today, emphasising their respective performance in terms of time of solution of the Finite Element problem, usage of memory, number of iterations, the value of |R| achieved when converged. The performance evaluation is made using the University Cambridge Research Computing Service (CDS3) using Message Passing Interface (MPI) implementations that allows parallelisation using 2 nodes of the cluster. The overview is restricted to three phenomema solved by Partial Differential Equations (PDEs) with the Finite Elements Method taking into consideration million Degrees of Freedom (DoF) to be solved. Along with the preconditioners, a variety of options were tested to optimise its performance.

Keywords— preconditioners, MOOSE, HPC, Heat Conduction 2D, Compute Finite Strain Elastic Stress 3D, Navier Stokes 3D

I. Introduction

Solving a Partial Differential Equation (PDE) involves at some point, if using the Finite Element Method (FEM), the resolution of large sparse linear equation system.

To solve a large sparse linear equation system, iterative methods have been used for many years. Lately, a wide-ranging of preconditioners have been developed to speed up the convergence of iterative methods; even more, the purpose is to make them converged.

At the moment, it is still questionable whether or not it is possible to apply a specific preconditioner to a certain type of PDE. This is a desired goal for many researchers nowadays.

This work provides benchmark measurements of preconditioners that helps the simulation of basic multiphysic phenomena to be used in the future, in the simulation of complex systems as part of a Tokamak at UKAEA.

The physic phenomena problems are the Heat Conduction 2D, Compute Finite Strain Elastic Stress 3D and the Incompressible Navier Stoke 3D.

The benchmark combines a variety of preconditioners such as the Additive Schwartz Method (ASM), Lower Upper Method (LU), Incomplete Lower Upper actorization Method (ILU), Block Jacobi Method (Bjacobi), Jacobi Method, Geometric Algebraic Multigrid Method (GAMG), and the Parallel Algebraic Multigrid (HypreBoomerAMG).

Contact data: Julita Inca Chiroque, julita.incachiroque@ukaea.uk

The set of benchmark problems also take into consideration the Newton method, the Preconditioned Jacobian-Free Newton Krylov (PJFNK) and the use of 112 MPI cores (2 nodes of a cluster) to test parallelisation performance.

The Krylov Subspace assessed in this work to perform the benchmark are as follows: Conjugate Gradient (cg), BiConjugate Gradient (bcg), Conjugate Gradient Squared (cgs), Biconjugate Gradient Stabilized (bcgs), Conjugate Residual (cr), Minimum Residual (minres), Generalized Minimal Residual (gmres), Flexible Generalized Minimal Residual (fgmres), Chebyshev, Richardson, PREONLY, Transpose-Free Quasi-Minimal Residual (1) (tfqmr), Transpose-Free Quasi-Minimal Residual (2) (tcqmr), Least Squares (lsqr).

This paper primarily presents results of the faster preconditioners, including its optimisation flags, while converging a solution for the multiphysic phenomena already described.

These results are in regards of the total execution time of the simulation, the time of the calculation of the Finite Element problem (represented by the `FEProblem::solve` parameter), the memory usage for the entire simulation, the time that the preconditioner used for the simulation, the value of the norm of the residual (|R|), and the number of iterations.

The findings of this work reinforces two foundations: preconditioners perform better while parallelising the execution, and the use of the optimisation flags of the preconditioners speed up the time and helps converging solutions of the large sparse linear equation system.

II. The cluster

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service [1], provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council [2]. CSD3 is a our new Cascade Lake cluster that contains 112 nodes dedicated to UKAEA. Each node consists of 56 cores Intel Xeon Platinum 8276 CPU @ 2.20GHz, and 192GB of memory. The interconnection uses Mellanox HDR100 with full-bandwidth, via fat tree with 3:1 oversubscription.

III. The Multiphysic Framework

Multiphysic Object Oriented Simulation Environment (MOOSE) is a framework which is built using many software libraries like PETSc (Portable, Extensible Toolkit for Scientific Computation), HYPRE (High Performance Preconditioners) and libMesh (Mesh generator). MOOSE provides packages that represent multiphysic phenomena based on solutions of PDEs, in a form of *Kernel*.

The computational toolkit MOOSE allows simulation of a PDE by creating the mesh of the phenomena using LibMesh; and using PETSc to calculate the unknown variables of a problem. MOOSE supports HPC using the MPI library. PETSc already comprises software to use different numerical elements such as Krylov subspaces, solver types, different types of matrices, and preconditioning iterative methods. The combination of these elements are crucial to solve a system of linear equations [3].

IV. The preconditioners

A linear system equation is commonly defined by the formula: $A\vec{x} = \vec{b}$; where `A` is a quadratic matrix, `b` is a known vector, and `x` is a vector to be solved. In this scenario, MOOSE supports through PETSc, the construction of the matrix, the organization of the entries while solving the linear equation system, and the solver type to the linear equation system. In this case, it was tested: NEWTON and PJFNK. Whenever this formula does not convergence due to condition or nature of the matrix, a preconditioner comes in the form of a matrix by helping the convergence of the linear equation system. Different preconditioners transform the initial matrix calculation by applying different algorithms. One parameter to choose the best approximation of the simulation is the value of the norm of the residual |R|.
Conforming to [4], the PETSc software comprises of 42 types of preconditioners. This list was obtained from the its database by using the output of the command `pc_type`:

```
asm, bjacobi, cholesky, eisenstat, exotic, gamg,
hypre, hypre_boomeramg, hypre_euclid, composite,
hypre_parasails, hypre_pilut, icc, ilu, jacobi,
ksp, lu, mg, ml, none, pfmg, redundant, sor, tfs,
telescope, , pbjacobi, vpbjacobi, shell, mat, nn,
fieldsplit, galerkin, cp, patch, lsc, svd, bddc,
 redistribute, kaczmarz, gasm, syspfmg, lmvm
```

V. The simulations

a. Heat Conduction 2D (steady state)

This simulation allows a mechanism of distributing heating in a certain two dimensional domain without a production of internal heat. It is governed by the Laplace equation (1) to find out the Temperature of each point of a mesh.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

This elliptic PDE can be solve by using finite differences for the second derivative as it is stated in (2).

$$u_{x,y+h} + u_{x,y-h} + u_{x+h,y} + u_{x-h,y} - 4u_{x,y} = 0 \tag{2}$$

where $u(x,y) \approx T(x,y)$, $h = 20/n$, $n =$ size of an interval.

The MOOSE's Kernel that solves this simulation is the Heat Conduction [5]. The type of element set is QUAD4 which indicates a form of square, and the calculation of the temperature has to be done in the 4 vertex (node of the square). A first order of Lagrange is also configured for the solution.

Fig.1 shows the conditions set in the first experiment. The temperature that is going to be spread throughout the steel square of 1m x 1m is constricted to the temperatures defined in the Boundary Conditions (BC). On the top, the Dirichlet BC was imposed with a fixed temperature of 373^o. In the bottom, another Dirichlet BC was fixed to 298^o. On the left, a Neumann BC that uses Heat Flux is defined to disperse the temperature in the Y axis from 3000^o. On the right, another Neumann BC is establish to use the adiabatic mode with not heat transfer.

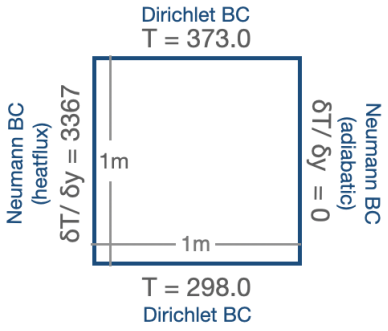

Fig. 1: Boundary Conditions for Heat Conduction 2D

Additional configuration:

```
Heat capacity of steel is 460 J/(kg°C)
Thermal conductivity of steel is 29.7
```

b. Compute Finite Strain Elastic Stress 3D

One of the most performed calculations in the fusion engineering are coupled thermo-mechanical simulations, where the displacement field is directly related to the absolute temperature.

The material's properties utilizes in this experiment are: density steel, stress, strain, and the elasticity tensor steel. The stress of the material is calculated on each point of the mesh, as well as the displacement suffered on each direction (x, y, z).

The governing equation for this thermo-mechanical problem are defined in the Small Linearized Total Strain formula (3), and the calculation of Compute Linear Elastic Stress Stresses (4) in Tensor Mechanics:

$$\varepsilon = \frac{1}{2}(\mathbf{u}\nabla + \nabla\mathbf{u}) \, when \, \frac{\partial u}{\partial x} << 1 \tag{3}$$

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl}^{total} \tag{4}$$

The stress and the displacement are configured in MOOSE by using [6] [7] [8] and the kernel Rank Two Scalar Aux [9]. The second order of Lagrange is configured for the solution, and the type element is HEX20 where the element is a cube with 20 points distributed as 3 nodes per side of the cube.

Fig.2 shows that the DirichletBC is used for the three directions. On `x`, it is set that there is not going to be a displacement, as well as in the `z` direction. On `y`, it is set a slightly deformation of -0.01

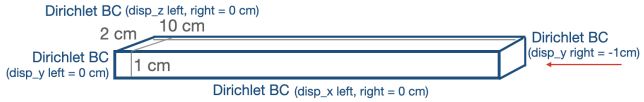

Fig. 2: Boundary Conditions for the Compute Finite Strain Elastic Stress 3D

Additional configuration:

```
Density of steel is 7800
Youngs modulus is 200e9
Poissons ratio is 0.3
```

c. *The Incompressible Navier Stoke 3D*

This simulation allows a mechanism of distributing a viscous incompressible fluid in a certain three dimensional domain. It is governed by the following equations (c) (c) (c) to find out the Velocity in the three dimensions of each point of a mesh.

$$\begin{aligned}
&\rho\left(\frac{\partial v_x}{\partial t} + v_x\frac{\partial v_x}{\partial x} + v_y\frac{\partial v_x}{\partial y} + v_z\frac{\partial v_x}{\partial z}\right) = \\
&\mu\left[\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2}\right] - \frac{\partial P}{\partial x} + \rho g_x \\
&\rho\left(\frac{\partial v_y}{\partial t} + v_x\frac{\partial v_y}{\partial x} + v_y\frac{\partial v_y}{\partial y} + v_z\frac{\partial v_y}{\partial z}\right) = \\
&\mu\left[\frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} + \frac{\partial^2 v_y}{\partial z^2}\right] - \frac{\partial P}{\partial y} + \rho g_y \\
&\rho\left(\frac{\partial v_z}{\partial t} + v_x\frac{\partial v_z}{\partial x} + v_y\frac{\partial v_z}{\partial y} + v_z\frac{\partial v_z}{\partial z}\right) = \\
&\mu\left[\frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} + \frac{\partial^2 v_z}{\partial z^2}\right] - \frac{\partial P}{\partial z} + \rho g_z
\end{aligned}$$

In order to represent the continuity equation that allows the conservation of the fluid is calculated with (5):

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0 \tag{5}$$

The MOOSE's kernel that solves these equations on the three dimensions (x, y, z) is called INSMomentumLaplaceForm [10]. The corresponding MOOSE's kernel to calculate the pressure is INSMass [11].

The type element set is HEX8 that represents a cube with eight nodes where the calculation of the velocity and the pressure, is done.

Fig.3 shows the Boundary Conditions (BC) configuration. The DirichletBC is used for the six walls on each dimension. On `x`, the velocity is set with zero on the top, bottom, front and back. On the left, the velocity is set to 0.005 and on the right, the pressure is set to zero. For the `y` dimension, the velocity is set to zero to all the walls (top, bottom, front, back, left, right). Similarly, on the `z` direction, the velocity is set to zero for all its sides.

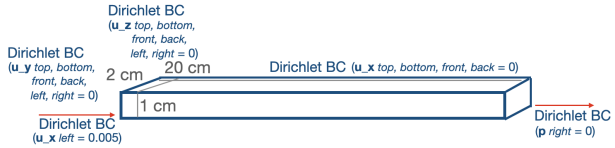

Fig. 3: Boundary Conditions for Incompressible Navier Stoke 3D

One way to solve Navier Stokes is by applying the Streamline upwind Petrov–Galerkin pressure-stabilizing (SUPG-PSPG) method to achieve stabilization in the solution obtained. The integration by parts for the Pressure calculation is disabled.

Additional configuration:

```
Density is 1000
Viscosity is 0.001
```

VI. The Experiment

a. *Applying KSP options to preconditioners*

According to [12], for large-scale sparse linear systems, Krylov-subspace methods are among the most powerful techniques. Then, a combination of 360 options were performed along the preconditioners in this experiment. As follows, the list of KSP and KSP options tested:

- Conjugate Gradient (cg)
- BiConjugate Gradient (bcg)
- Conjugate Gradient Squared (cgs)
- Biconjugate Gradient Stabilized (bcgs)
- Conjugate Residual (cr)
- Minimum Residual (minres)
- Generalized Minimal Residual (gmres)
- Flexible Generalized Minimal Residual (fgmres)
- Chebyshev
- Richardson
- PREONLY
- Transpose-Free Quasi-Minimal Residual (1) (tfqmr)
- Transpose-Free Quasi-Minimal Residual (2) (tcqmr)
- Least Squares (lsqr)

Additionally, other configurations are set such as the positioning of the KSP type (right, left, or left - right). The Scalable Nonlinear Equations Solvers (SNES) type such Newton based nonlinear solver that uses a line search (newtonls) and Newton based nonlinear solver that uses a trust region (newtontr) were tested. In regards to the tolerances, MOOSE provides limits for both, the absolute and relative tolerance in the linear and non-linear calculations respectively.

b. Applying options of the preconditioners

Seven preconditioners were selected to do the benchmark

- Additive Schwartz Method (ASM)
- Lower Upper Method (LU)
- Incomplete Lower Upper actorization Method (ILU)
- Block Jacobi Method (Bjacobi)
- Jacobi Method
- Geometric Algebraic Multigrid Method (GAMG)
- Parallel Algebraic Multigrid (HypreBoomerAMG)

c. Getting the best options for preconditioners

Having the faster results from the combination of the KSP types and SNES options; the outcome is tested along the own options of the preconditioners already listed. Finally, the faster results are performed for millions of DoFs cases.

VII. Results

a. Heat Conduction 2D

The faster times and number of solve converged outputs are getting by using the NEWTON method over the PJFNK method. Likely, the MPI performed better among the other modes such OpenMP and Hybrid (OpenMP+MPI). Then, the results shown as follows are restricted to these conditions:

DoF: **163865601**
Mesh Size: **12800 x 12800**
MPI cores: **112 (2 nodes)**
Solve Type: **NEWTON**

HypreBommerAMG is the fastest preconditioner that helps converging. It is configured with FGMRES, CGS and CG:

Preconditioner with KSP options	Execution T(s) (FE solve) (s)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_right -ksp_type=**fgmres**	213.35 (67.033)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -pc_factor_shift=NONZERO -ksp_type=**cgs** -snes_type=newtonls	219.75 (69.514)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_right -pc_factor_shift=NONZERO -ksp_type=**cg** -snes_type=newtontr	220.20 (69.951)

Later, the following options of the **HypreBommerAMG** preconditioner are tested along with the FGMRES method, using right preconditioning; as the CG, and the CGS method.

```
-pc_hypre_boomeramg_strong_threshold=0.25
-pc_hypre_boomeramg_max_levels=15
-pc_hypre_boomeramg_max_levels=20
-pc_hypre_boomeramg_agg_nl=2
-pc_hypre_boomeramg_agg_nl=3
-pc_hypre_boomeramg_agg_num_paths=4
-pc_hypre_boomeramg_agg_num_paths=5
-pc_hypre_boomeramg_agg_num_paths=6
-pc_hypre_boomeramg_truncfactor=0.2
-pc_hypre_boomeramg_truncfactor=0.3
-pc_hypre_boomeramg_truncfactor=0.4
-pc_hypre_boomeramg_P_max=2
```

Then, the best combination tested that accomplishes the best time for solving the Heat Conduction 2D is:

`hypre:`

```
-pc_type=hypre -pc_hypre_type=boomeramg
-ksp_right -pc_factor_shift=NONZERO
-ksp_type=cg -snes_type=newtontr
-pc_hypre_boomeramg_truncfactor=0.4
```

Similarly, it is listed the best combination of options for:

`bjacobi:`

```
-pc_type=bjacobi -ksp_type=cg -snes_type=newtontr
```

`asm:`

```
-pc_type=asm -ksp_type=cgs -snes_type=newtonls
-pc_asm_overlap=32 -pc_asm_type=interpolate
```

`gamg:`

```
-pc_type=gamg --pc_factor_shift=NONZERO --ksp_type=cg
```

`jacobi:`

```
-pc_type=jacobi --ksp_left --ksp_type=tcqmr
```

`ilu:`

```
-pc_type=ilu -ksp_left -ksp_type=cr
-pc_type_ilu_levels=32
```

`lu:`

```
-pc_type=lu -ksp_left -ksp_type=fgmres
```

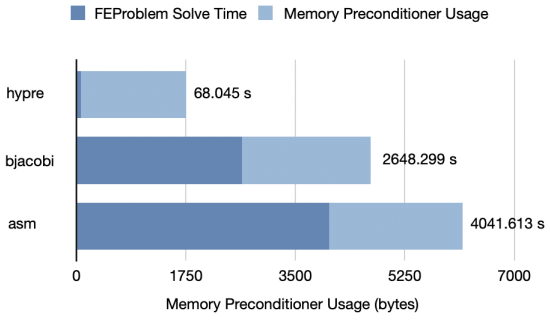

Fig. 4: Resources to solve Heat Conduction 2D

HypreBoomerAMG is ≈ 40 times faster than **Bjacobi**.
HypreBoomerAMG is ≈ 60 times faster than **ASM**.

Preconditioner	Execution Time (s)	FEProblem Time(s)	Total Memory (MB)	Preconditioner Memory (b)	|R|	Iterations
hypre	214.225	68.045	584.015	1688	2.13E-06	15
bjacobi	2793.066	2648.299	887.941	2056	2.51E-05	13669
asm	4188.600	4041.613	1773.483	2136	3.62E-05	7521

Walltime for this experience: **2 hours** | *Hypre* times is the Average of 4runs | *Bjacobi* times is the Average of 3 runs) | *ASM* time has 1 run

TABLE 1: BENCHMARKING OF HEAT CONDUCTION 2D

As it is seen in *Fig. 4* and in `Table 1`, the computational resources are better used by the **HypreBoomerAMG** preconditioner. Both, the total of memory and the memory of the preconditioner are less in comparison to the other two preconditioners, Bjacobi and ASM.

HypreBoomerAMG uses $\approx 300MG$ less than **Bjacobi**.
HypreBoomerAMG uses $\approx 1.2GB$ less than **ASM**.

In regards to the norm of the residual |R|, **HypreBoomerAMG** achieves the less value; as well as the number of iterations spent to converge a solution.

b. Compute Finite Strain Elastic Stress 3D

The benchmarking for this PDE also included the previous statements as in the Heat Conduction 2D. Once again, the NEWTON method defeats the PJFNK method in execution time. The MPI performed better among the other parallelisation modes such OpenMP and Hybrid (OpenMP+MPI). Formerly, the results for this experience are constrained to:

DoF: **4000083**
Mesh Size: **200 x 40 x 40**
MPI cores: **112 (2 nodes)**
Solve Type: **NEWTON**

One more time, **HypreBommerAMG** is the fastest preconditioner that helps converging the Displacement 3D when it is configured with KSP types: GMRES and FGMRES as it is shown as follows:

Preconditioner with KSP options	Execution T(s) (FE solve) (s)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_type=**fgmres** -ksp_right	231.300 (177.401)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -pc_factor_shift=NONZERO -ksp_type=**gmres** -snes_type=newtonls	232.225 (179.325)
-pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_left -ksp_type=**gmres** -pc_factor_shift=NONZERO	252.775 (200.313)

The options for **HypreBommerAMG** are tested along with the options of the FGMRES and GMRES method:

```
-pc_hypre_boomeramg_strong_threshold=0.70
-pc_hypre_boomeramg_max_levels=15
-pc_hypre_boomeramg_max_levels=20
-pc_hypre_boomeramg_agg_nl=2
-pc_hypre_boomeramg_agg_nl=3
-pc_hypre_boomeramg_agg_num_paths=4
-pc_hypre_boomeramg_agg_num_paths=5
-pc_hypre_boomeramg_agg_num_paths=6
-pc_hypre_boomeramg_truncfactor=0.2
-pc_hypre_boomeramg_truncfactor=0.3
-pc_hypre_boomeramg_truncfactor=0.4
-pc_hypre_boomeramg_P_max=2
```

The parameter `-pc_hypre_boomeramg_strong_threshold` changed from the previous experience (Heat Conduction 2D) because this time, the experiment is in 3D. Finally, the best combination tested that accomplishes the best time is:

`hypre:`

```
-pc_type=hypre -pc_hypre_type=boomeramg
-ksp_left -pc_factor_shift=NONZERO
-ksp_type=gmres -pc_hypre_boomeramg_P_max=2
```

For the rest of preconditioners, the best combinations are:

`gamg:`

```
-pc_type=gamg -pc_factor_shift=NONZERO -ksp_right
-ksp_left -ksp_type=bcgs -pc_gamg_agg_nsmooths=1
```

`jacobi:`

```
-pc_type=jacobi -ksp_left -ksp_type=cg
-pc_jacobi_rowmax
```

`bjacobi:`

```
-pc_type=bjacobi -ksp_right -ksp_type=cg
-snes_type=newtonls
```

`asm:`

```
-pc_type=asm -ksp_type=bcgs
-snes_type=newtonls -pc_asm_type=interpolate
```

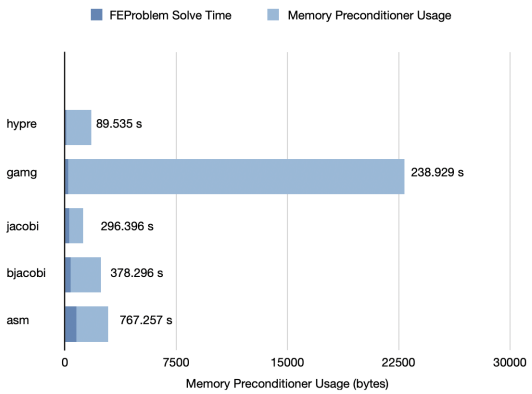

Fig. 5: Resources to solve Compute Finite Strain Elastic Stress 3D

HypreBoomerAMG is ≈ 2 times faster than **GAMG**.
HypreBoomerAMG is ≈ 3 times faster than **Jacobi**.
HypreBoomerAMG is ≈ 4 times faster than **Bjacobi**.
HypreBoomerAMG is ≈ 8 times faster than **ASM**.

Preconditioner	Execution Time (s)	FEProblem Time(s)	Total Memory (MB)	Preconditioner Memory (b)	|R|	Iterations
hypre	185.275	89.535	816.027	1688	0.002418092	115
gamg	334.725	238.929	1145.523	22648	0.002295673	366
jacobi	390.825	296.396	806.058	920	0.002461506	8030
bjacobi	472.550	378.296	911.490	2056	0.002443643	5521
asm	861.075	767.257	1117.737	2136	0.00111105	3967

- indicates that either the execution was killed due to lack of memory, not converged, nor in walltime (an hour and 10 minutes) | Execution Time and FEProblem Time are the average of 4 runs

TABLE 2: BENCHMARKING OF COMPUTE FINITE STRAIN ELASTIC STRESS 3D

As it is seen in *Fig. 5* and in `Table 2`, the computational resources are better used by the **HypreBoomerAMG** preconditioner. Both, the total of memory and the memory of the preconditioner are less in comparison to the best second option: the **GAMG** preconditioner.

HypreBoomerAMG uses $\approx 300MG$ less than **GAMG**.
In regards to the norm of the residual |R|, **HypreBoomerAMG** achieves approximately the same value as the following three best times's preconditioners.

In terms of the number of iterations, the **HypreBoomerAMG** utilizes less number of iterations in comparison to the rest preconditioners shown in `Table 2`.

c. Incompressible Navier Stokes 3D

The results
DoF: **4545964**
Mesh Size: **600 x 30 x 60**
MPI cores: **112 (2 nodes)**
Solve Type: **NEWTON**

Preconditioner with KSP options	Execution T(s) (FE solve) (s)
-pc_type=**asm** -ksp_type=**cgs** -snes_type=newtonls	294.65 (256.809)
-pc_type=**asm** -ksp_type=**gmres** -ksp_gmres_restart=100 -snes_type=newtonls	425.35 (387.943)
-pc_type=**asm** -ksp_type=**gmres** -ksp_gmres_modifiedgramschmidt	1289.00 (1251.671)

The **ASM** preconditioners's options that are tested along with the CGS and GMRES KSP types are listed as follows:

```
-pc_asm_type=basic
-pc_asm_type=restrict
-pc_asm_type=interpolate
-pc_asm_type=none
-pc_asm_overlap=4
-pc_asm_overlap=16
-pc_asm_overlap=32
-pc_asm_overlap=64
-sub_pc_type=lu
```

Lastly, the final combination that provides the best time to solve converged is:

`asm:`

```
-pc_type=asm -pc_asm_type=interpolate
-ksp_type=cgs -snes_type=newtonls
```

The rest of preconditioners that converges and accomplishes the best times using the combination of KSP types and preconditioner's options are:

`hypre:`

```
-pc_type=hypre -pc_hypre_type=boomeramg
-pc_hypre_boomeramg_strong_threshold=0.7
-ksp_type=gmres -snes_type=newtonls
-pc_hypre_boomeramg_agg_nl=3
```

`bjacobi:`

```
-pc_type=bjacobi
```

`lu:`

```
-pc_type=lu -ksp_type=cg -snes_type=newtonls
```

`ilu:`

```
-pc_type=ilu -ksp_type=gmres
-ksp_gmres_restart=400 -snes_type=newtonls
```

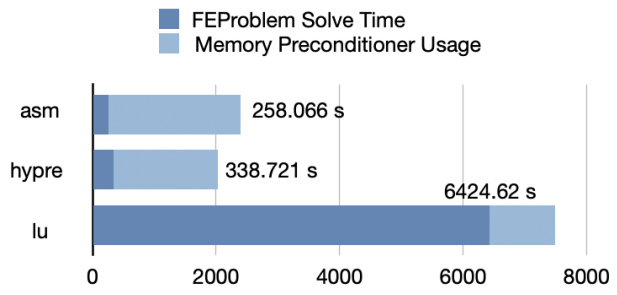

Fig. 6: Resources to solve the Incompressible Navier Stokes 3D

ASM is ≈ 1.3 times faster than **HypreBoomerAMG**.
ASM is ≈ 24 times faster than **LU**.

As it is seen in *Fig. 6* and in `Table 3`, the fastest computational time is achieved by the **ASM** preconditioner. However, the amount of memory spent is not the more convenient if comparing with the second best choice preconditioner **HypreBoomerAMG** preconditioner. **HypreBoomerAMG** uses $\approx 100MG$ less than **ASM**.
In regards to the norm of the residual |R|, **ASM** achieves good approximation, and in terms of the number of iterations, the **ASM** utilizes more iterations as it is shown in `Table 3`.

One fact in this last experience is that is not necessarily to combine the options of the preconditioners aside the KSP type options. The time achieved in both cases are very similar. Specially for the FEM solving time: 256.809 (with KSP types) versus 258.066 (preconditioner's options).

Preconditioner	Execution Time (s)	FEProblem Time(s)	Total Memory (MB)	Preconditioner Memory (b)	|R|	Iterations
asm	295.450	258.066	283.683	2136	1.23E-19	1273
hypre	375.825	338.721	103.233	1688	1.47E-19	170
lu	6464.000	6424.620	94.745	1064	2.10E-21	8

Walltime for this experience: **2 hours** | *ASM* and *HypreBoomerAMG (4 runs)* | *LU (2 runs)*

TABLE 3: BENCHMARKING OF INCOMPRESSIBLE NAVIER STOKES 3D

VIII. CONCLUSIONS

The benchmarking of the preconditioners is done with the following preconditioners: **hypre_boomeramg**, **bjacobi**, **asm**, **gamg**, **jacobi**, **lu** and **ilu**. The experiment is performed using 2 nodes (112 MPI cores) in the CSD3 cluster, and the use of the NEWTON's solver type. Then, it is concluded:

1.- The preconditioner **hypre_boomeramg** demonstrates better performance and usage of computational resources to solve the **Heat Conduction 2D** problem with more than 160 millions of DoFs. Due to the small difference (in seconds), it is recommendable to use the following combinations:

- -pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_type=**fgmres** -ksp_right
- -pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_type=**cg** -ksp_right -pc_factor_shift=NONZERO -snes_type=newtontr -pc_hypre_boomeramg_truncfactor=0.4

2.- The best preconditioner to converge a solution for the **Compute Finite Strain Elastic Stress 3D** phenomena is **hypre_boomeramg** when solving more than 4 millions DoFs. The following combination reaches the best time:

- -pc_type=**hypre** -pc_hypre_type=**boomeramg** -ksp_type=**gmres** -ksp_left -pc_factor_shift=NONZERO -pc_hypre_boomeramg_P_max=2

3.- The best preconditioner to converge a solution for the **Incompressible Navier Stokes 3D** phenomena is **asm** when solving more than 4 millions DoFs. The following combination reaches the best time:

- -pc_type=**asm** -ksp_type=**cgs** -snes_type=newtonls
- -pc_type=**asm** -ksp_type=**cgs** -pc_asm_type=interpolate -snes_type=newtonls

IX. DISCUSSION AND FURTHER WORK

The Heat Conduction 2D ends up with a convergence by using the CG and FGMRES KSP types. CG usually helps the solution of symmetric matrix, specially Symmetric Definite Positive, whereas FGMRES is a KSP type prone to solve non-symmetric. In this matter, a further research involves to find out the type of the matrix in this problem and what is the impact of adding internal heating.

The other two 3D problems converged by using both, the HypreBoomerAMG and the ASM. A deep analysis of these two algorithms are essential to understand the blocks for ASM and how it is distributed when larger geometries and transient systems are involved.

X. ACKNOWLEDGMENT

This document is intended for publication in the open literature. It is made available on the understanding that it may not be further circulated and extracts or references may not be published prior to publication of the original when applicable, or without the consent of the UKAEA Publications Officer, Culham Science Centre, Building K1/0/83, Abingdon, Oxfordshire, OX14 3DB, UK.

REFERENCES

[1] www.csd3.cam.ac.uk.

[2] www.dirac.ac.uk.

[3] S. Abhyankar, J. Brown, E. M. Constantinescu, D. Ghosh, B. F. Smith, and H. Zhang, "Petsc/ts: A modern scalable ode/dae solver library," *arXiv preprint arXiv:1806.01437*, 2018.

[4] https://www.mcs.anl.gov/petsc/documentation/tutorials/HandsOnExercise, "PETSc Users Manual," *Mathematics and Computer Science Division*, vol. Revision 3.15, no. 1, pp. 8–11, 2021.

[5] https://mooseframework.inl.gov/source/kernels/HeatConduction.html.

[6] https://mooseframework.inl.gov/modules/tensor_mechanics/Strains.html.

[7] https://mooseframework.inl.gov/modules/tensor_mechanics/Stresses.html.

[8] https://mooseframework.inl.gov/modules/tensor_mechanics/StressDivergence.html

[9] https://mooseframework.inl.gov/source/auxkernels/RankTwoScalarAux.html.

[10] https://mooseframework.inl.gov/source/kernels/INSMomentumLaplaceForm.html.

[11] https://mooseframework.inl.gov/source/kernels/INSMass.html.

[12] A. Ghai, C. Lu, and X. Jiao, "A Comparison of Preconditioned Krylov SubspaceMethods for Large-Scale Nonsymmetric Linear Systems," p. 4, 2018.