Leon Wilkosz	193101

AMCS 312

Homework 2

Comments on the submitted files

We attach all code files, scripts, and log files for the numerical results. There is one script per exercise (with corresponding name) which saves the numerical results into a corresponding .txt file and modifications of the given ps2 program are named after their exercise and dimension. For convenience, we list them in the following:

- `ps2_ex3_OnShaheen.sh`: shaheen bash script for exercise 3
- `ps2_ex4_OnShaheen.sh`: shaheen bash script for exercise 4
- `ps2_ex5_OnShaheen.sh`: shaheen bash script for exercise 5
- `HW2_results_ex3.txt`: computational results from exercise 3
- `HW2_results_ex4.txt`: computational results from exercise 4
- `HW2_results_ex5.txt`: computational results from exercise 5
- `ps2-1D.c`: 1D poisson solver for exercise 3
- `ps2-2D.c`: 2D poisson solver for exercise 3
- `ps2-3D.c`: 3D poisson solver for exercise 3
- `ps2-1D_ex4a.c`: 1D poisson solver for exercise 4 (a)
- `ps2-2D_ex4a.c`: 2D poisson solver for exercise 4 (a)
- `ps2-3D_ex4b.c`: 3D poisson solver for exercise 4 (a)
- `ps2-1D_ex4b.c`: 1D poisson solver for exercise 4 (b)
- `ps2-2D_ex4b.c`: 2D poisson solver for exercise 4 (b)
- `ps2-3D_ex4b.c`: 3D poisson solver for exercise 4 (b)
- `makefile`
- `matrix_1D.mtx, matrix_2D.mtx, matrix_3D.mtx` matrices for plotting in exercise 3
- `contour_plot1D.pdf, contour_plot2D.pdf, contour_plot3D.pdf` contour plots of matrices

Exercise 3

1.

We record the default options for the computations in this exercise in table 1 and see that indeed the size of the matrix is the only quantity that changes.

Dimension	**Default KSP/PC**	**Matrix Size**
1D	GMRES / ILU (0 fill)	9
2D	GMRES / ILU (0 fill)	9×9
3D	GMRES / ILU (0 fill)	$9 \times 9 \times 9$

Table 1 Default KSP, PC options and Matrix Size for 1D, 2D, and 3D problems.

2.

We record the required data in table 2.

Dimension	Wall-clock Time (s)	FLOPS/sec	Error $\|u - u_{exact}\|_\infty$	KSP Iterations
1D	0.0224	2.59×10^4	0.00110617	1
2D	0.0275	4.94×10^6	0.000221974	11
3D	0.0707	5.56×10^7	0.0215725	15

Table 2 Performance metrics for 1D, 2D, and 3D cases.

3.

In this exercise we plot the assembled matrices in 1D, 2D, and 3D. In the 1D matrix one sees the typical structure of the 1D laplacian, 2 on the diagonals, -1 on the off diagonals. Since the stencils become larger in higher dimension, one row contains more values in 2D and 3D than in 1D. However, we also see that the matrices are still sparse. for the same Δx, the 2D and 3D matrices are much larger since they contain much more grid points. The pattern we see in the plot depends on the ordering of the gridpoints in the linear system and this is not the only possible choice.

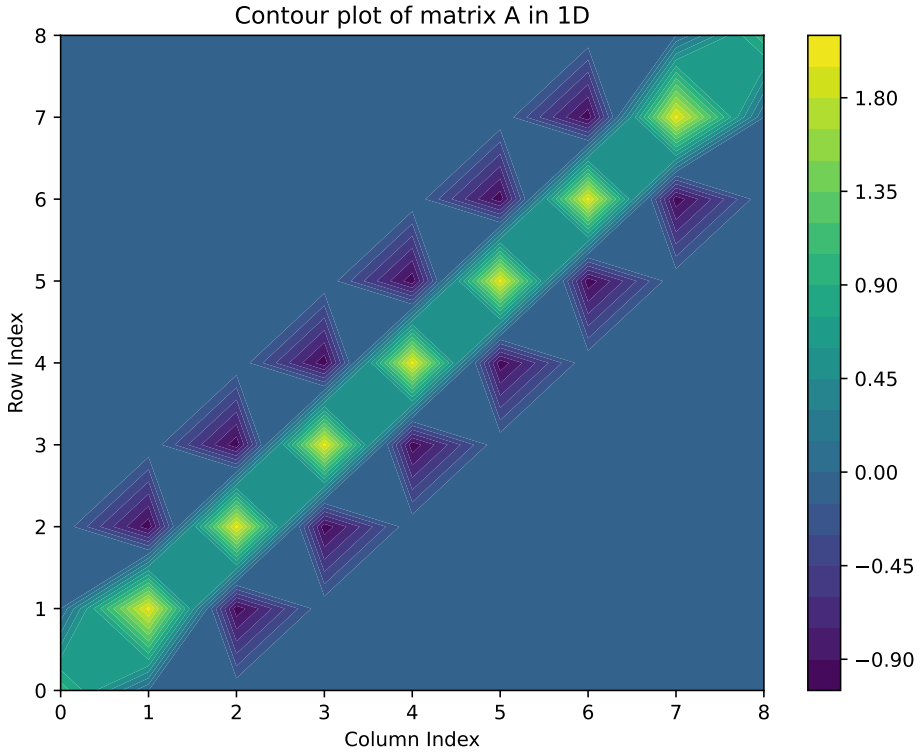

Figure 1 1D Contour plot of matrix

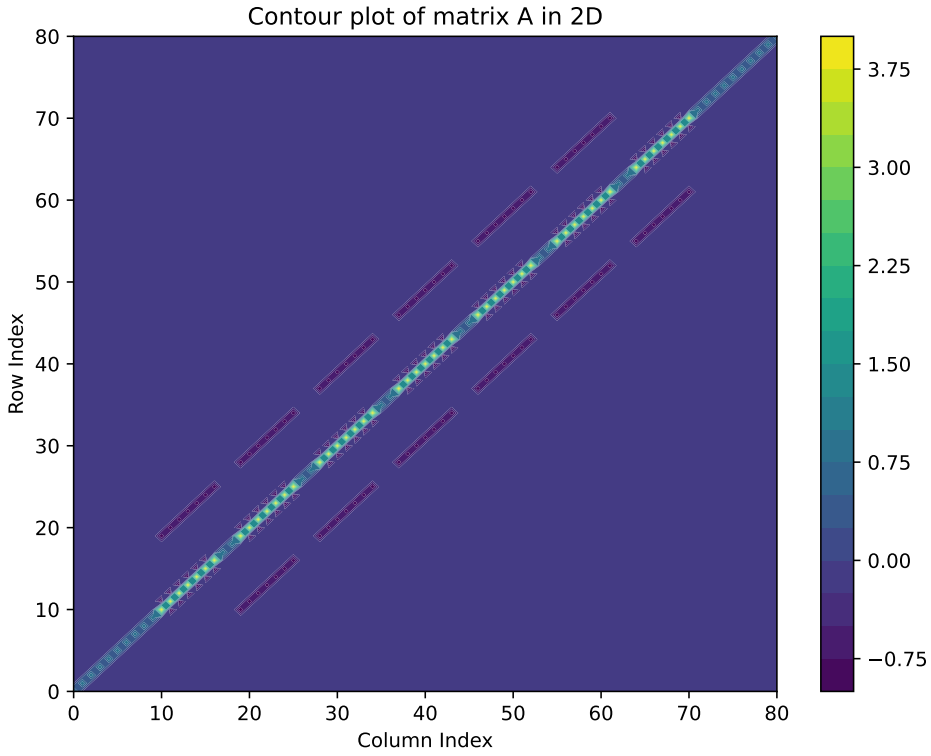

Figure 2 2D Contour plot of matrix

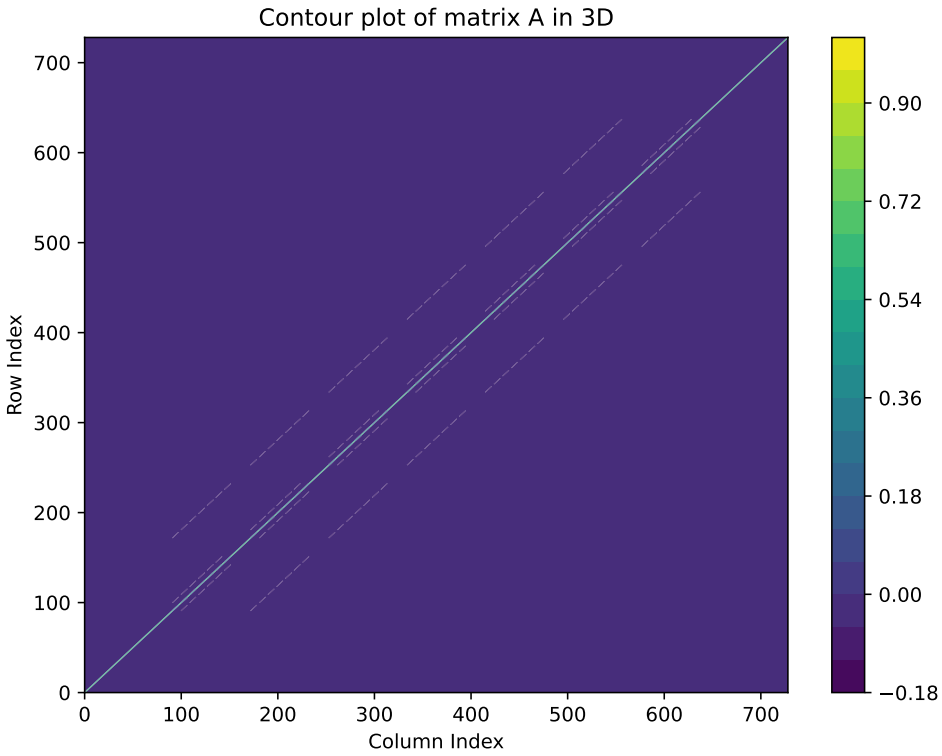

Figure 3 3D Contour plot of matrix

4.

The KSP solver terminates after one iteration. The residual norm can be seen in table 3. Since the numerical error in the infinity norm is given by 0.00110617, we see that the residual error is much smaller. Hence, the error made through discretising (and not the one made when approximating the solution to the linear system) dominates the overall numerical error here.

Iteration	**Residual Norm**
0	6.143×10^{-1}
1	7.765×10^{-17}

Table 3 KSP Iterations and Residual Norms. Error $\|u - u_{exact}\|_\infty = 0.00110617$.

5.

We record the required Eigenvalues in Table 4.

Iteration	**Eigenvalue**
0	0.328089
1	0.51186
2	0.548359
3	0.661475
4	0.785197
5	0.898367
6	1.00172
7	1.06566

Table 4 Eigenvalues computed in successive iterations.

6.

We collect in Table 5 the grid sizes, number of KSP iterations, the infinity norm error, the 1-norm error, the 2-norm error and their normalized variants. If n is the dimension of the vector, then we need to normalize the 1-norm by dividing by n and the 2-norm by dividing by $\sqrt{n}$ to obtain a reasonable error metric. This is because

$$\|\cdot\|_1 \leq n \|\cdot\|_\infty$$

and

$$\|\cdot\|_2 \leq \sqrt{n} \|\cdot\|_\infty$$

and because after normalizations these metrics become discretizations of the L^1 and L^2 norms for functions. In Table 6 we divide the error of one discretization by the error of the next coarser discretization. We see that overall that the respective value is about 4 which is what we need to confirm the convergence rate, since the truncation error of the PDE is expected to be reduced by a factor of 4 every time we double the resolution. We also observe that we need much more KSP iterations with increasing size of the matrix.

Grid Size	**Iterations**	$\|u - u_{\mathbf{exact}}\|_\infty$	$\|u - u_{\mathbf{exact}}\|_1$	$\|u - u_{\mathbf{exact}}\|_2$	**Normalized 1-norm**	**Normalized 2-norm**
9x9	8	0.000763959	0.0211785	0.00329941	2.6212e-04	3.6653e-04
17x17	12	0.000196764	0.0216913	0.00164961	7.4938e-05	1.2146e-04
33x33	23	4.91557e-05	0.0218203	0.000824732	2.2506e-05	4.5323e-05
65x65	56	1.29719e-05	0.0227638	0.000430681	8.4922e-06	1.5704e-05
129x129	174	3.76924e-06	0.0256992	0.000245575	1.5412e-06	3.1172e-06

Table 5 Grid Size vs. Error and Iterations

Grid Size	Infinity Norm	1-Norm	2-Norm	Normalized 1-Norm	Normalized 2-Norm
9x9					
17x17	3.882	0.9764	1.999	3.496	3.016
33x33	4.003	0.995	2.000	3.329	2.678
65x65	3.788	0.958	1.915	2.648	2.882
129x129	3.442	0.885	1.753	5.510	5.037

Table 6 Convergence Rates

7.

We print the condition number with respect to the 2-norm for the matrices in the previous discussion in Table 7. This can be done by dividing the largest by the smallest Eigenvalue $\frac{\lambda_{max}}{\lambda_{min}}$. We computed approximations of these Eigenvalues in a previous exercise, when we listed them in the KSP iteration. We see that with growing size the matrices have larger condition numbers. This fits very well to our results on the convergence from the previous exercise, as we need much more KSP iterations for larger matrices, and the condition number describes how strong numerical errors propagate, i.e. how good a linear system can be solved.

Grid Size	Condition Number
9x9	3.249
17x17	8.377
33x33	24.126
65x65	50.329
129x129	89.345

Table 7 Condition Numbers

8.

We record the results of our computations of numerical errors for different combinations of gridsizes and relative tolerances in Table 8. We observe that for certain grid sizes it does not make sense to put effort in reaching very high accuracy in the KSP iteration because at some point the truncation error dominates and we obtain no further improvements by doing so.

Grid Size	RTOL = 1e-1	RTOL = 1e-5	RTOL = 1e-10
$9 \times 9 \times 9$	0.00139236	0.000169304	0.000169304
$17 \times 17 \times 17$	0.00168387	4.2181e-05	4.21805e-05
$33 \times 33 \times 33$	0.00146642	1.05899e-05	1.05776e-05

Table 8 Numerical error in infinity norm for different grid sizes and relative tolerances.

9.

1D grid: The Laplacian in 1D is given by the second derivative

$$\Delta\varphi = \frac{\partial^2\varphi}{\partial x^2}.$$

Using forward and backward finite difference approximations for one derivative, we obtain

$$\varphi'(x) \approx \frac{\varphi(x+h) - \varphi(x)}{h}$$

$$\varphi'(x) \approx \frac{\varphi(x) - \varphi(x-h)}{h}$$

Then we use the again finite differences to approximate the second derivative

$$\varphi''(x) \approx \frac{\varphi'(x+h/2) - \varphi'(x-h/2)}{h} \approx \frac{\frac{\varphi(x+h)-\varphi(x)}{h} - \frac{\varphi(x)-\varphi(x-h)}{h}}{h} \approx \frac{\varphi(x+h) - 2\varphi(x) + \varphi(x-h)}{h^2}.$$

Scaling now by h^2 gives $h^2\varphi''(x) \approx \varphi(x+h) - 2\varphi(x) + \varphi(x-h)$. Renaming $\varphi_i := \varphi_i(x),\ \varphi_{i-1} := \varphi(x-h),\ \varphi_{i+1} := \varphi(x+h)$ gives the discrete Laplacian

$$h^2\Delta\varphi_i = \varphi_{i-1} - 2\varphi_i + \varphi_{i+1}.$$

3D grid: The Laplacian in 3D is given by

$$\Delta\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2}.$$

For the x-derivative we obtain the finite difference:

$$\varphi'_x(x) \approx \frac{\varphi(x+h_x,y,z) - \varphi(x,y,z)}{h_x},$$

$$\varphi'_x(x) \approx \frac{\varphi(x,y,z) - \varphi(x-h_x,y,z)}{h_x}$$

and hence iteratively

$$\varphi''_{xx}(x) \approx \frac{\frac{\varphi(x+h_x,y,z)-\varphi(x,y,z)}{h_x} - \frac{\varphi(x,y,z)-\varphi(x-h_x,y,z)}{h_x}}{h_x} = \frac{\varphi(x+h_x,y,z) - 2\varphi(x,y,z) + \varphi(x-h_x,y,z)}{h_x^2}.$$

Similarly, we obtain for y- and z-derivatives:

$$\varphi''_{yy}(y) \approx \frac{\varphi(x,y+h_y,z) - 2\varphi(x,y,z) + \varphi(x,y-h_y,z)}{h_y^2}$$

$$\varphi''_{zz}(z) \approx \frac{\varphi(x,y,z+h_z) - 2\varphi(x,y,z) + \varphi(x,y,z-h_z)}{h_z^2}.$$

Substituting this into the first equation gives

$$\begin{aligned}\Delta\varphi \approx\ & \frac{\varphi(x+h_x,y,z) - 2\varphi(x,y,z) + \varphi(x-h_x,y,z)}{h_x^2} \\ &+ \frac{\varphi(x,y+h_y,z) - 2\varphi(x,y,z) + \varphi(x,y-h_y,z)}{h_y^2} \\ &+ \frac{\varphi(x,y,z+h_z) - 2\varphi(x,y,z) + \varphi(x,y,z-h_z)}{h_z^2}.\end{aligned}$$

Renaming the function values into φ_{ijk} as in the 1D case and scaling by $h_xh_yh_z$ gives:

$$\begin{aligned}h_xh_yh_z\Delta\varphi_{ijk} \approx\ & h_yh_z(\varphi_{i+1,j,k} - 2\varphi_{i,j,k} + \varphi_{i-1,j,k}) \\ &+ h_xh_z(\varphi_{i,j+1,k} - 2\varphi_{i,j,k} + \varphi_{i,j-1,k}) \\ &+ h_xh_y(\varphi_{i,j,k+1} - 2\varphi_{i,j,k} + \varphi_{i,j,k-1}).\end{aligned}$$

Exercise 4

(a)

We compute the Laplacians:

$$\textbf{1D:} \quad \hat{\varphi}(x) = x - x^2, \quad \Delta\hat{\varphi}(x) = -2$$

$$\textbf{2D:} \quad \hat{\varphi}(x,y) = (x - x^2)(y^2 - y), \quad \Delta\hat{\varphi}(x,y) = -2(y^2 - y) + 2(x - x^2)$$

$$\textbf{3D:} \quad \hat{\varphi}(x,y,z) = (x - x^2)(y^2 - y)(z - z^2)$$

$$\Delta\hat{\varphi}(x,y,z) = -2(y^2 - y)(z - z^2) + 2(x - x^2)(z - z^2) - 2(x - x^2)(y^2 - y)$$

and need to set in the code $f = -\Delta\hat{\varphi}$ in each dimension as the boundary condition. By direct computation, we can show that with a uniform step size the discrete laplacian at $x \in [0, 1]$ is identical to the true continuous Laplacian. For instance, in 1D we find

$$\begin{aligned}\frac{\hat{\varphi}(x+h) - 2\hat{\varphi}(x) + \hat{\varphi}(x-h)}{h^2} &= \frac{x + h - (x+h)^2 - 2(x - x^2) + x - h - (x-h)^2}{h^2} \\ &= -2 \\ &= \Delta\hat{\varphi}(x).\end{aligned}$$

Similar computations can be carried out for 2D and 3D. This means that the discretized equation is already an exact equation for the solution at the grid points. As a consequence, there is no truncation error and what we observe as a numerical error is the KSP residual after the last iteration. Hence, we get very good accuracy very fast as can be seen in Table 9, Table 10, and Table 11.

Grid Size	**Infinity Norm Error**
17	5.55112×10^{-17}
33	1.94289×10^{-16}
65	5.55112×10^{-16}

Table 9 Numerical Error in Infinity Norm for 1D

Grid Size	**Infinity Norm Error**
17×17	5.23059×10^{-8}
33×33	2.39077×10^{-7}
65×65	9.99247×10^{-8}

Table 10 Numerical Error in Infinity Norm for 2D

Grid Size	**Infinity Norm Error**
$17 \times 17 \times 17$	6.44696×10^{-8}
$33 \times 33 \times 33$	4.60936×10^{-8}
$65 \times 65 \times 65$	2.84432×10^{-8}

Table 11 Numerical Error in Infinity Norm for 3D

(b)

We compute first the Laplacians:

$$\textbf{1D:} \quad \hat{\varphi}(x) = 3x + \sin(20x), \quad \Delta\hat{\varphi}(x) = -400\sin(20x)$$

$$\textbf{2D:} \quad \hat{\varphi}(x,y) = 3x + \sin(20xy), \quad \Delta\hat{\varphi}(x,y) = -400(x^2 + y^2)\sin(20xy)$$

$$\textbf{3D:} \quad \hat{\varphi}(x,y,z) = 3x + 3z + \sin(20xyz), \quad \Delta\hat{\varphi}(x,y,z) = -400(x^2y^2 + x^2z^2 + y^2z^2)\sin(20xyz)$$

We adjust the code adding the boundary condition $g = \varphi$ and the new right-hand side $f = -\Delta\varphi$. Table 12, Table 13, and Table 14 show the numerical error for different dimensions and grid sizes. Table 15, and Table 17 show the convergence rates in the sense that the error of the respective grid is divided by the error at the next coarser grid. Since there is no coarser grid, the first row is empty. We expect that the numerical error shrinks by a factor of four every time we double the resolution. Since the quotients in the tables are about 4, we can confirm that we see the convergence rates.

Grid Size	Infinity Norm Error
17	0.250283
33	0.058908
65	0.01456

Table 12 1D Numerical Errors in Infinity Norm

Grid Size	Infinity Norm Error
17x17	0.118495
33x33	0.0279685
65x65	0.00650328

Table 13 2D Numerical Errors in Infinity Norm

Grid Size	Infinity Norm Error
17x17x17	0.0794387
33x33x33	0.0207824
65x65x65	0.00511722

Table 14 3D Numerical Errors in Infinity Norm

Grid Size	Convergence Rate
17	
33	4.247
65	4.044

Table 15 1D Convergence Rates

Grid Size	Convergence Rate
17x17	
33x33	4.236
65x65	4.299

Table 16 2D Convergence Rates

Grid Size	Convergence Rate
17x17x17	
33x33x33	3.819
65x65x65	4.062

Table 17 3D Convergence Rates

Exercise 5

(a)

1.

Table 18 shows the default options. We run this on a Laptop with 8 parallel processes and we see that the default options change compared to sequential case. Now the default uses block jacobi, which was not the case when running with one process only.

Option	Value
KSP Type	gmres
Restart Size	30
Orthogonalization	Classical Gram-Schmidt
Tolerance	10^{-5} (relative), 10^{-50} (absolute)
PC Type	bjacobi
Number of Blocks	8
Sub-PC Type	ilu (0 levels of fill)
Matrix Type	mpiaij

Table 18 Default KSP and PC options

2.

The default Krylov and preconditioner combination as seen in the previous part is not recommendable for our problem in particular. Both, gmres and ilu are univerisal and make no use of the symmetric structure. However, we see in Table 19, it is still much better than not applying any preconditioning at all. We also see in Table 19 that CG works much better, which makes sense because it is specifically designed for symmetric positive definite matrices.

3.

We record the required quantities in Table 19. We see that compared to our base model, these models can use at least an order higher number of FLOPS/sec due to parallel execution.
For GMRES with and without preconditioning, we read off the table that the execution time is in fact not proportional to the number of iterations. This might be due to high message passing activities which we also see in Table 19.
We plot the curve of the residual of CG with and without preconditioning in Figure 4. Though not exactly, we see an overall monotonically decreasing trend. The small spikes in the not preconditioned CG can come from rounding errors that amplify strongly due to the large condition number of the matrix.
Overall we see that preconditioning pays off in all cases. The direct solvers are about an order slower than the parallelizable algorithms, which was also to be expected.

Method	KSP Iterations	Error	Wall Clock Time (s)	Flops/sec	MPI Msg Count
GMRES no PC (30 vecs)	10000	0.0099491	1.455e+01	2.895e+08	4.135e+04
GMRES no PC (15 vecs)	10000	0.0217839	1.470e+01	1.763e+08	4.268e+04
GMRES no PC (60 vecs)	10000	0.00241019	2.062e+01	3.634e+08	4.068e+04
GMRES no PC (200 vecs)	10000	5.24349e-06	4.335e+01	5.283e+08	4.021e+04
GMRES Jacobi ILU (30 vecs)	7225	1.17312e-06	2.705e+01	1.260e+08	2.988e+04
GMRES Jacobi ILU (15 vecs)	5881	1.16662e-06	3.188e+01	5.742e+07	2.511e+04
GMRES Jacobi ILU (60 vecs)	3943	1.17264e-06	2.040e+01	1.542e+08	1.605e+04
GMRES Jacobi ILU (200 vecs)	1723	1.07208e-06	1.346e+01	2.918e+08	6.940e+03
CG without PC	2048	0.000399514	6.701e+00	3.458e+05	6.156e+03
CG with Jacobi	373	0.00039954	7.011e+00	8.329e+04	1.131e+03
LU Direct Solver	1	4.81122e-08	4.956e+01	4.745e+08	0
Cholesky (ND)	1	4.81122e-08	1.414e+02	1.107e+06	0
Cholesky (RCM)	1	4.81122e-08	1.515e+02	1.033e+06	0

Table 19 Poisson Solver Results

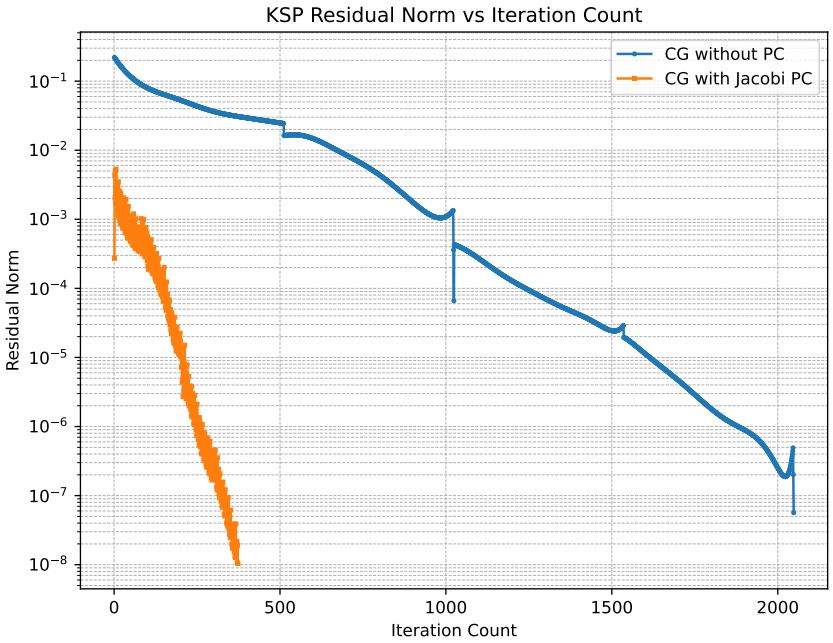

Figure 4 Redidual curves of CG with and without PC

4.

We record the results of our simulations in Table 20 and observe that the Jacobi-preconditioned needs significantly fewer iterations with growing resolution. This is because the CG method depends strongly on the condition number of the system and (Jacobi-)preconditioning helps to deal with high condition numbers. The ICC can give further lower iteration counts for our problem because it makes use of the symmetric structure. Jacobi preconditioning does not do that.

Method	**KSP Iterations**	**Error**	**Wall Clock Time (s)**	**Flops/sec**	**MPI Msg Count**
CG no PC (0 refine)	17	0.00076388	1.313e+00	3.931e+03	6.300e+01
CG no PC (2 refine)	73	4.91819e-05	6.171e-01	1.088e+05	3.080e+02
CG no PC (4 refine)	299	3.07512e-06	1.452e+00	6.516e+05	1.212e+03
CG no PC (6 refine)	1227	1.94253e-07	4.377e+00	3.276e+06	4.924e+03
CG with Jacobi PC (0 refine)	13	0.000763991	7.761e-01	7.003e+03	5.100e+01
CG with Jacobi PC (2 refine)	41	4.91809e-05	5.552e-01	9.475e+04	1.800e+02
CG with Jacobi PC (4 refine)	129	3.07288e-06	1.868e+00	2.934e+05	5.080e+023
CG with Jacobi PC (6 refine)	450	1.98955e-07	4.506e+00	1.654e+06	1.816e+03

Table 20 Poisson Solver Results CG Refinements

5.

We record the data on the multigrid methods to Table 21. One can clearly see that these are in terms of the wall clock time much faster than most of the other methods. Moreover, Hypre's BoomerAMG has on top a surprisingly small message count.

Method	KSP Iterations	Error	Wall Clock Time (s)	Flops/sec	MPI Msg Count
PCGAMG 2 levels	7	7.1221e-07	5.842e+00	1.433e+06	3.198e+03
PCGAMG 4 levels	7	7.1221e-07	6.522e+00	1.283e+06	3.198e+03
PCGAMG 12 levels	7	7.1221e-07	6.412e+00	1.306e+06	3.198e+03
PCGAMG 3 levels, 3 choices for KSP, PC	7	1.94253e-07	4.407e+00	2.022e+06	3.340e+03
W-cycle multigrid	7	0.00205431	3.941e+00	3.065e+06	3.646e+03
CG with Jacobi PC (2 refine)	41	4.91809e-05	5.552e-01	9.475e+04	1.800e+02
MG type additive	20	0.00638369	4.586e+00	3.988e+06	4.454e+03
MG type full	4	8.96908e-05	5.474e+00	2.101e+06	5.722e+03
MG Galerkin Process	7	0.00161951	5.351e+00	2.341e+06	3.212e+03
Hypre BoomerAMG	4	1.16158e-06	3.716e+00	1.572e+05	3.200e+01

Table 21 Poisson Solver Results CG Refinements

(b)

In Table 22 we record the KSP convergence for CG + Block Jacobi and CG + BoomerAMG. We observe that the multigrid variant needs much less iterations at the 3rd and 4th refinement. However, The average runtimes are slightly slower with BoomerAMG. Unfortunately, we receive an error when refining further and attempting to solve this with BoomerAMG:

```
 srun:  job 3684994 queued and waiting for resources
srun:  job 3684994 has been allocated resources
xpmem_attach error:  :  No such file or directory
srun:  error:  nid00163:  task 220:  Killed
srun:  Terminating StepId=3684994.
```

	KSP iterations	Average Runtime
CG + Block Jacobi + ICC (refine 3)	155	2.305e+00
CG + Block Jacobi + ICC (refine 4)	270	4.397e+00
CG + Block Jacobi + ICC (refine 5)	536	1.635e+01
CG + Block Jacobi + ICC (refine 6)	1082	1.178e+02
CG + BoomerAMG (refine 3)	5	3.316e+01
CG + BoomerAMG (refine 4)	4	1.917e+01

Table 22 KSP iterations in 3D