Krylov Methods

Mohammad Zandsalimy

December 20, 2021

Krylov methods are explored on large systems of linear equations arising from the implicit solution of incompressible, laminar, 2D energy equation with a given velocity field. The assembled implicit system is solved only once to study the effect of the linear system solver in use. The incompressible energy equation is presented in Eq. 1. The computational domain is a rectangular channel with a size of 40 in x direction and 1 in y direction. The velocity field in all cases is assumed to be fully developed and is given as in Eq. 2. $\overline{u}$ in this equation is a measure of the average flow rate in the channel which for our purposes is assumed to be 3 $[m/s]$. This velocity profile is a parabolic distribution. We assume that the temperature distribution on the bottom and top walls are constant and equal to 0 and 1, respectively. Further, the temperature at outlet is assumed to be fully developed and $T(y) = y$ at the inlet. Also, we have Re= 25, Pr= 0.7, and Ec=0.1. Time step is also set equal to 0.25.

$$\frac{\partial T}{\partial t}+\frac{\partial uT}{\partial x}+\frac{\partial vT}{\partial y}=\frac{1}{\text{RePr}}\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\frac{Ec}{Re}\left[2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)^2\right] \tag{1}$$

$$\begin{cases} u(x,y)=6\overline{u}y(1-y) \\ v(x,y)=0 \end{cases} \tag{2}$$

Second order centered fluxes are used to compute the flux integrals. The equation for flux integration is presented as Eq. 3. Further, the equation for source term is presented as Eq. 4.

$$\begin{aligned}\text{Flux}_{i,j} = &-\left[\frac{\overline{u}_{i+1,j}\overline{T}_{i+1,j}-\overline{u}_{i-1,j}\overline{T}_{i-1,j}}{2\Delta x}\right]-\left[\frac{\overline{v}_{i,j+1}\overline{T}_{i,j+1}-\overline{v}_{i,j-1}\overline{T}_{i,j-1}}{2\Delta y}\right] \\ &+\frac{1}{\text{RePr}}\left[\frac{\overline{T}_{i+1,j}-2\overline{T}_{i,j}+\overline{T}_{i-1,j}}{\Delta x^2}+\frac{\overline{T}_{i,j+1}-2\overline{T}_{i,j}+\overline{T}_{i,j-1}}{\Delta y^2}\right]\end{aligned} \tag{3}$$

$$\begin{aligned}\text{Source}_{i,j} =&\frac{\text{Ec}}{\text{Re}}\left[2\left(\frac{\overline{u}_{i+1,j}-\overline{u}_{i-1,j}}{2\Delta x}\right)^2+2\left(\frac{\overline{v}_{i,j+1}-\overline{v}_{i,j-1}}{2\Delta y}\right)^2\right.\\&\left.+\left(\frac{\overline{u}_{i+1,j}-\overline{u}_{i-1,j}}{2\Delta y}+\frac{\overline{v}_{i,j+1}-\overline{v}_{i,j-1}}{2\Delta x}\right)^2\right]\end{aligned} \tag{4}$$

1 Direct Solver

The δ form of the implicit equation is used to set up the linear system of equations which is presented in Eq. 5. Our baseline numerical grid is 20×10 with 200 equations and unknowns. Subsequent finer meshes are constructed by multiplying the baseline mesh size by 2 in each direction. The fill pattern for coefficients matrix of the assembled system on the baseline mesh is presented in Fig. 1.

$$\begin{aligned}&\frac{\delta\overline{T}_{i,j}}{\Delta t}+\left[\frac{\overline{u}_{i+1,j}\delta\overline{T}_{i+1,j}-\overline{u}_{i-1,j}\delta\overline{T}_{i-1,j}}{2\Delta x}\right]+\left[\frac{\overline{v}_{i,j+1}\delta\overline{T}_{i,j+1}-\overline{v}_{i,j-1}\delta\overline{T}_{i,j-1}}{2\Delta y}\right]\\&\quad-\frac{1}{\text{RePr}}\left[\frac{\delta\overline{T}_{i+1,j}-2\delta\overline{T}_{i,j}+\delta\overline{T}_{i-1,j}}{\Delta x^2}+\frac{\delta\overline{T}_{i,j+1}-2\delta\overline{T}_{i,j}+\delta\overline{T}_{i,j-1}}{\Delta y^2}\right]\\&\quad=\text{Flux}_{i,j}+\text{Source}_{i,j}\end{aligned} \tag{5}$$

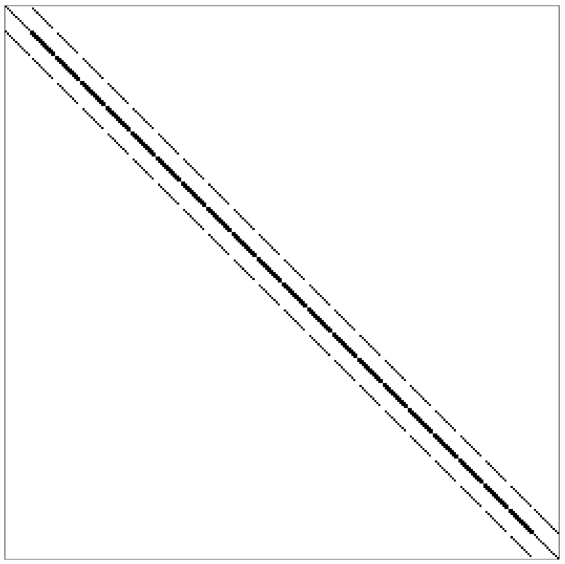

Figure 1: Fill pattern of coefficients matrix for the baseline mesh

An LU solver is utilized for direct solution of the linear system. Run time of the program is accurately calculated using *chrono* library. The run time of one iteration of the solution is presented vs. mesh size in Table 1. All the tests are conducted using level 3 optimization of GNU compiler. As seen in this table, run time increases with increasing number of equations. The order of time increment vs. number of unknowns is also presented in this table which shows $O(N^3)$ increment in run time. We know that solving a sparse matrix using LU decomposition requires $O(N^3)$ operations (according to [1]) which is consistent with the conclusion. Further, the residual is close to zero which means this is an exact solution.

Table 1: Run time vs. mesh refinement for the direct solver

Mesh	Run Time $[ms]$	Residual	Order
20×10	6.989	6.34E-17	
40×20	347.087	2.51E-16	2.8170337
80×40	23942.752	1.16E-15	3.0540739
160×80	1363722.112	6.82E-15	2.9159084

2 Approximate Factorization

We can use approximate factorization to decompose Eq. 5 in x and y directions as in Eq. 6. As a result, Eq. 5 can be rewritten as Eq. 7, which can in turn be written in the decomposed version of Eq. 8. The solution is carried out on rows of cells first, and then on columns. In this way, we get the two step in Eq. 9 for our implicit solver. α and β coefficients in this equation are presented in Eq. 10. The resulting tridiagonal matrices are solved using a Thomas algorithm.

$$[I] + \Delta t[\Delta x] + \Delta t[\Delta y] \approx ([I] + \Delta t[\Delta x])\,([I] + \Delta t[\Delta y]) \tag{6}$$

$$([I] + \Delta t[\Delta x])\,([I] + \Delta t[\Delta y])\overrightarrow{\delta T} = \Delta t\overrightarrow{\mathrm{R}} \tag{7}$$

$$\begin{cases} ([I] + \Delta t[\Delta x])\,\widetilde{\delta T} = \Delta t\overrightarrow{\mathrm{R}} \\ ([I] + \Delta t[\Delta y])\overrightarrow{\delta T} = \widetilde{\delta T} \end{cases} \tag{8}$$

$$\begin{cases} (-\beta_{x_{i-1,j}} - \alpha_x)\widetilde{\delta T}_{i-1,j} + (1 + 2\alpha_x)\widetilde{\delta T}_{i,j} + (\beta_{x_{i+1,j}} - \alpha_x)\widetilde{\delta T}_{i+1,j} = \Delta t\overrightarrow{\mathrm{R}}_{i,j} \\ (-\beta_{y_{i,j-1}} - \alpha_y)\overline{\delta T}_{i,j-1} + (1 + 2\alpha_y)\overline{\delta T}_{i,j} + (\beta_{y_{i,j+1}} - \alpha_y)\overline{\delta T}_{i,j+1} = \widetilde{\delta T}_{i,j} \end{cases} \tag{9}$$

$$\begin{cases} \alpha_x = \dfrac{\Delta t}{\mathrm{RePr}\Delta x^2} \\ \alpha_y = \dfrac{\Delta t}{\mathrm{RePr}\Delta y^2} \\ \beta_{x_{i,j}} = \dfrac{u_{i,j}\Delta t}{2\Delta x} \\ \beta_{y_{i,j}} = \dfrac{u_{i,j}\Delta t}{2\Delta y} \end{cases} \tag{10}$$

The run time of one iteration of the solution is presented vs. mesh size in Table 2. All the tests are conducted using level 3 optimization of GNU compiler. As seen in this table, run time increases with increasing number of equations. The order of time increment vs. number of unknowns is also presented in this table which shows $O(N^1)$ increment in run time. According to [1], solving a tridiagonal system with n equations only requires $O(n)$ operations. Further, we need to solve m tridiagonal systems of size n and n tridiagonal systems of size m in our setup. In this setting, m and n are the number of rows and columns of the mesh, respectively. As a result, our time increment should be of order $O(2 \times n \times m)$ which is equal to $O(2N)$. This is consistent with our result. Furthermore, the residual in the approximate factored version of the problem increases with mesh size. The error of approximate factorization shows itself in this table very well. This error seems too high at first glance, but we can conduct a more careful study to confirm the correctness of this calculation which is presented in the next section.

Table 2: Run time vs. mesh refinement for the approximate factorization

Mesh	Run Time $[ms]$	Residual	Order
20×10	0.007	2.79E-03	
40×20	0.034	4.47E-03	1.1400540
80×40	0.155	7.17E-03	1.0943308
160×80	0.654	1.21E-02	1.0385112

3 Direct Solution of Approximate Factorization

To confirm the residuals of the approximate factorized version of the equation which are solved using a Thomas algorithm, we assemble the approximate factorized equations and solve them directly using the LU decomposition. This means, do the multiplication in Eq. 7 and assemble the entire matrix at once. Obviously, this time there are some extra terms (compared to Eq. 5). The fill pattern for coefficients matrix of the assembled system on the baseline mesh is presented in Fig. 2.

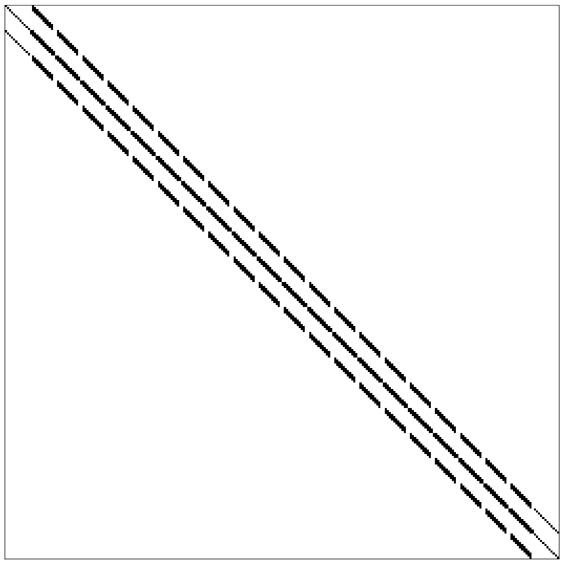

Figure 2: Fill pattern of the approximated factorized matrix for the baseline mesh

An LU solver is utilized for direct solution of this linear system. The run time of one iteration of the solution is presented vs. mesh size in Table 3. All the tests are conducted using level 3 optimization of GNU compiler. As seen in this table, run time increases with increasing number of equations. The order of time increment vs. number of unknowns is also presented in this table which shows $O(N^3)$ increment in run time. The residuals in this table confirm the correctness of our approximate solution in the previous section. We can see that run time is just as high as the results of the direct solution of the original matrix while the residuals are as bad as the ones from the tridiagonal solution of approximate factorization. This means, we are getting the worst of the both worlds.

Table 3: Run time vs. mesh refinement for direct solution of the approximate factorized matrix

Mesh	Run Time $[ms]$	Residual	Order
20×10	11.086	2.79E-03	
40×20	325.962	4.47E-03	2.4389465
80×40	23762.124	7.17E-03	3.0939081
160×80	1550920.249	1.21E-02	3.0141584

4 GMRES

A GMRES solver is developed (with no preconditioning) to solve the original system of linear equations from the direct solver. First thing to note here is that GMRES can do more than one iteration to get a better approximate solution to the linear system. While very useful, I think we should avoid conducting more than one GMRES iterations per solution. This is to have a better view on the residuals as well as the solution time. Otherwise, we will get errors close to zero which are not good to make conclusions. The run time of one iteration of the solution is presented vs. mesh size in Tables 4, 5, and 6 using 20, 30, and 40 GMRES vectors, respectively. All the tests are conducted using level 3 optimization of GNU compiler. As seen in these tables, run time increases with increasing number of equations. The order of time increment vs. number of unknowns is not constant in this case which shows a nonlinear behavior. This is mainly due to the LU decomposition of the solution to the least squares problem in the GMRES iteration. As seen here, using 20 vectors, the residual is less than the approximate factorization for 20×10 and 40×20 meshes. However, we are getting higher error for finer meshes for the solution with 20 GMRES vectors. On the other hand, in the case of 30 and 40 GMRES vectors, the residual is lower than the approximate factorization on the 80×40 mesh but we are still getting higher errors on the finest mesh. Using more GMRES vectors results in a decrement in the GMRES method residual while the run time does not change too much.

Table 4: Run time vs. mesh refinement for one GMRES iteration on the original matrix with 20 GMRES vectors

Mesh	Run Time [ms]	Residual	Order
20×10	2.003	2.69E-07	
40×20	18.118	1.67E-04	1.5885947
80×40	221.733	9.88E-03	1.8066639
160×80	3205.317	4.75E-02	1.9267858

Table 5: Run time vs. mesh refinement for one GMRES iteration on the original matrix with 30 GMRES vectors

Mesh	Run Time [ms]	Residual	Order
20×10	2.151	6.11E-10	
40×20	26.599	9.13E-06	1.8141463
80×40	323.134	2.18E-03	1.8013443
160×80	4712.905	2.77E-02	1.9332061

Table 6: Run time vs. mesh refinement for one GMRES iteration on the original matrix with 40 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	2.805	1.44E-12	
40×20	34.298	5.19E-07	1.8060259
80×40	422.802	5.99E-04	1.8118929
160×80	6170.405	1.78E-02	1.9336555

Next, let's set up this solver so that the output of the GMRES iteration is always within an error tolerance. Remember that we did not do this previously to have a better look at run time of the method. The GMRES iteration is converged to 10^{-10} before any other calculations. The results are presented in tables 7, 8, and 9. As seen here, the residual is very small in all cases, but the run time is higher compared to the previous test, obviously. Further, we are getting rate of convergence up to order 2.6 which is always nice to have.

Table 7: Run time vs. mesh refinement for converged GMRES on the original matrix with 20 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	2.628	2.65E-12	
40×20	81.516	1.16E-13	2.4775230
80×40	2610.005	4.55E-13	2.5004128
160×80	104596.288	4.70E-13	2.6623176

Table 8: Run time vs. mesh refinement for converged GMRES on the original matrix with 30 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	11.848	3.38E-17	
40×20	72.281	8.94E-13	1.3044865
80×40	2562.808	3.05E-13	2.5739826
160×80	99423.244	3.43E-13	2.6388929

Table 9: Run time vs. mesh refinement for converged GMRES on the original matrix with 40 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	6.856	1.44E-12	
40×20	97.398	2.92E-16	1.9142266
80×40	2520.935	1.81E-13	2.3469615
160×80	95462.797	3.60E-13	2.6214539

5 Preconditioned GMRES

The GMRES solver of previous section is modified with a preconditioning matrix which is simply the coefficients from the approximate factorization. This algorithm can do more than one iteration to get a better approximate solution to the linear system. First, The run time of one iteration of the solution is presented vs. mesh size in Tables 10, 11, and 12 using 20, 30, and 40 GMRES vectors, respectively. All the tests are conducted using level 3 optimization of GNU compiler. As seen in these tables, run time increases with increasing number of equations. The order of time increment vs. number of unknowns is not constant in this case which shows a nonlinear behavior. This is mainly due to the LU decomposition of the solution to the least squares problem in the GMRES iteration. As seen here, using any number of GMRES vectors (20, 30, or 40), the residual is less than the approximate factorization as well as the simple GMRES on all meshes. On the other hand, The run time of the preconditioned version of GMRES is pretty similar to the original version. Higher accuracy in the same amount of time? I'll take it, thank you very much. Further, the run time here is higher than the approximate factorization. Using more GMRES vectors results in a decrement in the preconditioned GMRES residual while the run time does not change too much.

Table 10: Run time vs. mesh refinement for one GMRES iteration on the original matrix with 20 GMRES vectors

Mesh	Run Time [ms]	Residual	Order
20×10	2.184	1.96E-16	
40×20	17.775	1.70E-11	1.5124024
80×40	227.962	3.80E-08	1.8404359
160×80	3308.72	2.10E-06	1.9297040

Table 11: Run time vs. mesh refinement for one preconditioned GMRES iteration on the original matrix with 30 GMRES vectors

Mesh	Run Time [ms]	Residual	Order
20×10	2.284	1.71E-10	
40×20	26.006	2.70E-15	1.7546050
80×40	333.923	3.63E-10	1.8412995
160×80	4875.509	1.34E-07	1.9339827

Table 12: Run time vs. mesh refinement for one preconditioned GMRES iteration on the original matrix with 40 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	3.095	2.05E-15	
40×20	35.273	4.03E-15	1.7552765
80×40	445.895	3.73E-12	1.8300339
160×80	6217.843	1.03E-08	1.9008191

Next, let's set up this solver so that the output of the GMRES iteration is accurate to an error tolerance. The preconditioned GMRES iteration is converged to 10^{-10} before any other calculations. The results are presented in tables 13, 14, and 15. As seen here, the residual is very small in all cases (smaller than the same test for the original GMRES), but the run time is higher, obviously. Compared to the original GMRES, preconditioned version can reach the accuracy tolerance in much lower time.

Table 13: Run time vs. mesh refinement for converged preconditioned GMRES on the original matrix with 20 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	6.561	1.96E-16	
40×20	36.354	8.38E-17	1.2350632
80×40	651.563	5.65E-16	2.0818593
160×80	12645.457	7.49E-13	2.1392853

Table 14: Run time vs. mesh refinement for converged preconditioned GMRES on the original matrix with 30 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	12.336	3.17E-17	
40×20	26.25	2.70E-15	0.5447214
80×40	661.245	4.73E-16	2.3273978
160×80	13678.188	5.84E-14	2.1852742

Table 15: Run time vs. mesh refinement for converged preconditioned GMRES on the original matrix with 40 GMRES vectors

Mesh	Run Time $[ms]$	Residual	Order
20×10	8.646	2.05E-15	
40×20	36.312	4.03E-15	1.0351708
80×40	876.011	3.34E-16	2.2962153
160×80	12074.482	5.39E-13	1.8924343

6 Discussion

The run time of all the tests are plotted vs. number of equations in the linear system in Fig. 3. According to this figure, the approximate factorization is the fastest in all cases and the direct solver is the slowest of the bunch. On the other hand, the original GMRES is slower than the preconditioned version of the algorithm. The GMRES iterations in this figure are converged to 10^{-10} tolerance. Fig. 4 shows the rum time of all the tests vs. residual. In this figure, only one GMRES iteration is conducted. As seen here, the run time of the direct method is the highest while giving the most accurate results. On the other hand, approximate factorization gives the least accurate results in the least amount of time. The sweet spot, of course, belongs to preconditioned GMRES which gives decent accuracy in a decent amount of time. The preconditioned GMRES displays a strange behavior in the residual with 30 vectors. I have triple checked my results, there is no problem to be found. I think this is due to the very small residuals in the solution. We cannot really compare small residuals confidently. Fig. 5 displays the same results only for the converged GMRES. As seen here, we are getting pretty much zero residual in decent amount of time for both GMRES methods. If I had the choice, I would go for the preconditioned version.

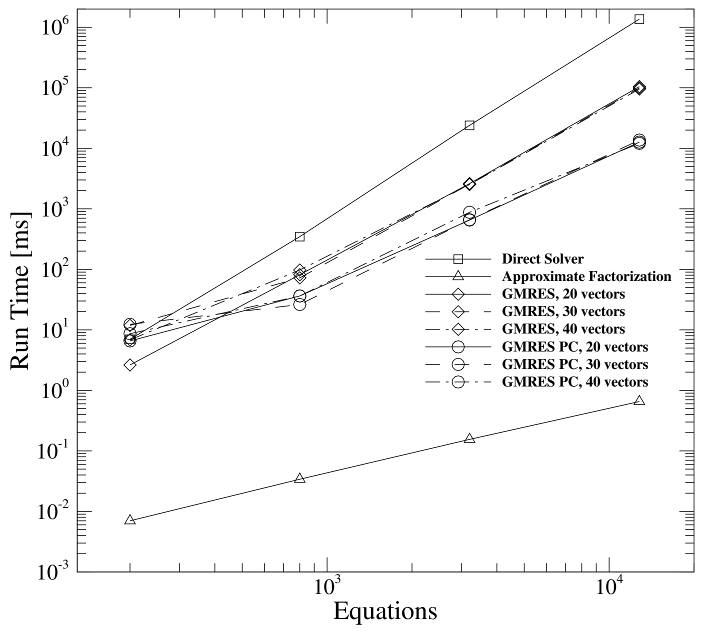

Figure 3: Run time vs. number of equations

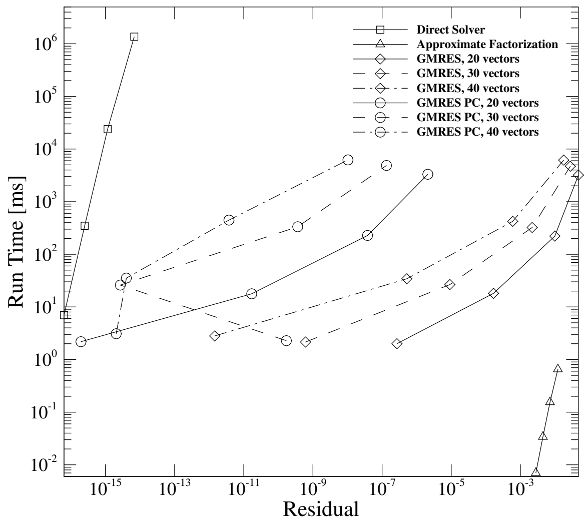

Figure 4: Run time vs. residual with a single GMRES iteration

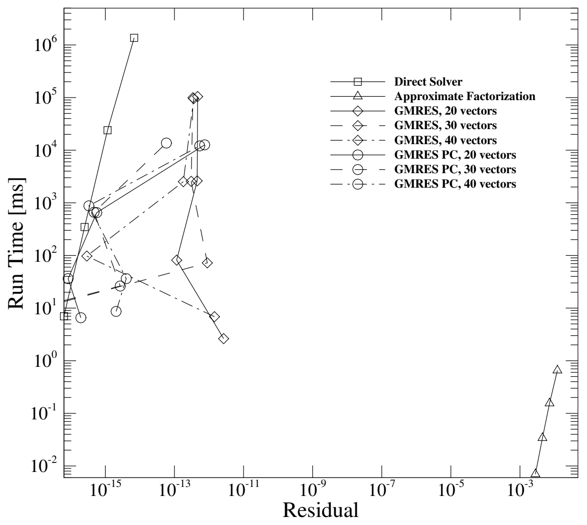

Figure 5: Run time vs. residual with converged GMRES

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing.* Cambridge University Press, 2002.