A Cyclic Reduction Solver for the Poisson Equation

MATH 478 Project Report

Written By: Noah LeFrançois (260706235)

Supervised By: Prof. Jean-Christophe Nave

April 27, 2020

Abstract

The application of the Characteristic Mapping Method [1] [2] to the incompressible Euler equations is an ongoing area of research; applications include the simulation of smoke and fire for computer graphics [3]. This method requires the solution of a Poisson equation in order to obtain the stream function of a fluid from its vorticity; currently a simple Fourier Transform (FT) solver is used. This project seeks to implement a more efficient solution using a Fourier Analysis and Cyclic Reduction (FACR) solver in order to increase speed when running simulations on large grids. In this paper we outline the theory behind each Poisson equation solver and present some results from our implementation of the FACR algorithm.

1 Introduction

The realistic modelling of natural fluid behaviour is a challenging area of numerical analysis, with a variety of applications from aerospace engineering to nuclear physics. Numerical solutions of the partial differential equations (PDEs) arising in the study of fluid mechanics often require the use of very large, sparse matrices. While relatively simple approaches can be implemented to invert the full matrices, this process can become prohibitively inefficient as matrix sizes increase with the size and resolution of a problem domain. As such, the development of more efficient methods which take advantage of sparse matrix structure are vital in order to produce fluid solvers with adequate speed for practical applications.

One such PDE which arises regularly is the 2D Poisson equation, which has the form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = p(x, y) \tag{1}$$

where p is a source term and u is some function for which we would like to solve. This equation can also be written using the Laplacian operator $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. In this report we implement a method for efficiently solving Eq. 1, Fourier Analysis and Cyclic Reduction (FACR). This can be applied to the modelling of smoke flow discussed in previous work [2], [3], the implementation of which currently uses a Fourier Transform method.

The remainder of this paper will be organized as follows. In section 2 we lay out the theoretical formulation of our model, in section 3 we apply our solver to the 2D Poisson

equation with a source term and present the results for accuracy and run-time, and in section 4 we make some concluding remarks about future improvements which can be made to our implementation.

2 Theory

For an incompressible fluid, the velocity field is expressed as the curl of a potential Ψ, called the stream function. The vorticity ω of this fluid can be expressed as the Laplacian of Ψ; we can rewrite Eq. 1 to describe this relationship by making the replacements $\Psi = u$ and $\omega = p$.

In the numerical model used by [2], [3], we have ω after each time step and want to solve Eq. 1 to find Ψ in order to calculate the velocity. This velocity can then be used to compute the next advection step, obtaining ω at the next time step and then repeating the process.

When the Poisson equation is discretized, we obtain a sparse matrix which needs to be inverted in order to obtain the solution u. Two techniques for efficiently doing this are the Fourier Transform (FT) method and the Cyclic Reduction (CR) method. FACR, in problems where it is applicable, can provide an even faster solution. For our 2D problem, we denote L_x,L_y as the lengths of the x and y domains; these axes are discretized with spacing h and indices i=1,2,...I & j=1,2,...,J. The x axis has periodic boundary conditions, while the y axis has Dirichlet boundary conditions where we specify u=0 at the top and bottom of the domain.

The FT method allows us to efficiently find u in terms of p in Fourier space, as in Eq. 3. [4] In the case of Dirichlet boundary conditions, the FFT and IFFT functions can be replaced with sine transforms since the coefficients of all cosine terms will be zero.

$$\hat{p}_{mn} = FFT(p_{ij}) \tag{2}$$

$$\hat{u}_{mn} = \frac{2(cos(\frac{\pi m}{I}) + cos(\frac{\pi n}{J}) - 2)}{h^2}\hat{p}_{mn} \tag{3}$$

$$u_{ij} = IFFT(\hat{u}_{mn}) \tag{4}$$

The CR method allows us to reduce the problem to solving J-1 tridiagonal systems. [4] On a grid of size (2^f+1)x(2^f+1), we can solve for the vector $u_{J/2}$ using the following equation, which gives a tridiagonal system:

$$\mathbf{T^{(f)}} \cdot \mathbf{u_{J/2}} = h^2 \mathbf{p_{J/2}^{(f)}} - \mathbf{u_0} - \mathbf{u_J} \tag{5}$$

where we define $T^{(f)}$ and $p_j^{(f)}$ using the following relations, taking $\mathbf{T^{(0)}} = \mathbf{B} - 2\mathbf{I}$ from the discretization in the x and y directions:

$$\mathbf{T^{(n)}} = 2\mathbf{I} - (\mathbf{T^{(n-1)}})^2$$

$$\mathbf{p_j^{(n)}} = h^2(\mathbf{p_{j-1}^{(n-1)}} - \mathbf{T^{(n-1)}}\mathbf{p_j^{(n-1)}} + \mathbf{p_{j+1}^{(n-1)}})$$

After solving for $\mathbf{u_{J/2}}$, we obtain two tridiagonal systems at level (f-1) involving $\mathbf{u_{J/4}}$ and $\mathbf{u_{3J/4}}$. This splitting continues at every stage until we have obtained all of the $\mathbf{u_j}$ vectors. Alternatively, a finite-difference (FD) solver could be combined with this method: CR is used for the first r stages to reduce the number of systems, then the FD solver is applied to fill in the intermediate y-lines.

FACR combines the two methods introduced above in order to accelerate the process; at the rth stage of CR we Fourier analyze Eq. 5 along x. This will give us a tridiagonal system in the y-direction for each x-Fourier mode:

$$\hat{u}_{j-2^r}^k + \lambda_k^{(r)} \hat{u}_j^k + \hat{u}_{j+2^r}^k = h^2 p_j^{(r)k} \tag{6}$$

We solve the tridiagonal system for $\hat{u}_j^k$ at the levels $j = 2^r, 2\times 2^r, ..., J-2^r$ and use the FT method to get the x-values on these y-lines. The simplest implementation uses r=0, which means that each of the systems is solved using this approach. An improved implementation would be as discussed above for the CR solver: FACR is used for the first r stages to reduce the number of systems, then a FD or FT solver is applied to fill in the intermediate y-lines. [4]

By selecting the value of r such that the computational cost is minimized, FACR can theoretically achieve an approximate gain of speed between a factor of two and a factor of four compared to either of the simpler FT or CR methods [4]. For the purposes of this project, we only implemented the simpler r=0 case.

3 Results

Fig. 1a) displays the test case used for the FACR solver: the source term p is given as $p(x,y) = sin(\frac{2\pi}{L_X}x)sin(\frac{2\pi}{L_Y}y)$. The Poisson equation with this source term has an exact solution $u(x,y) = -[(\frac{2\pi}{L_X})^2 + (\frac{2\pi}{L_Y})^2]^{-1}sin(\frac{2\pi}{L_X}x)sin(\frac{2\pi}{L_Y}y)$, which is displayed in Fig. 1b). Fig. 1c) displays the numerical solution obtained using our FACR solver, while Fig. 1d) displays the error of this numerical solution compared to the exact solution.

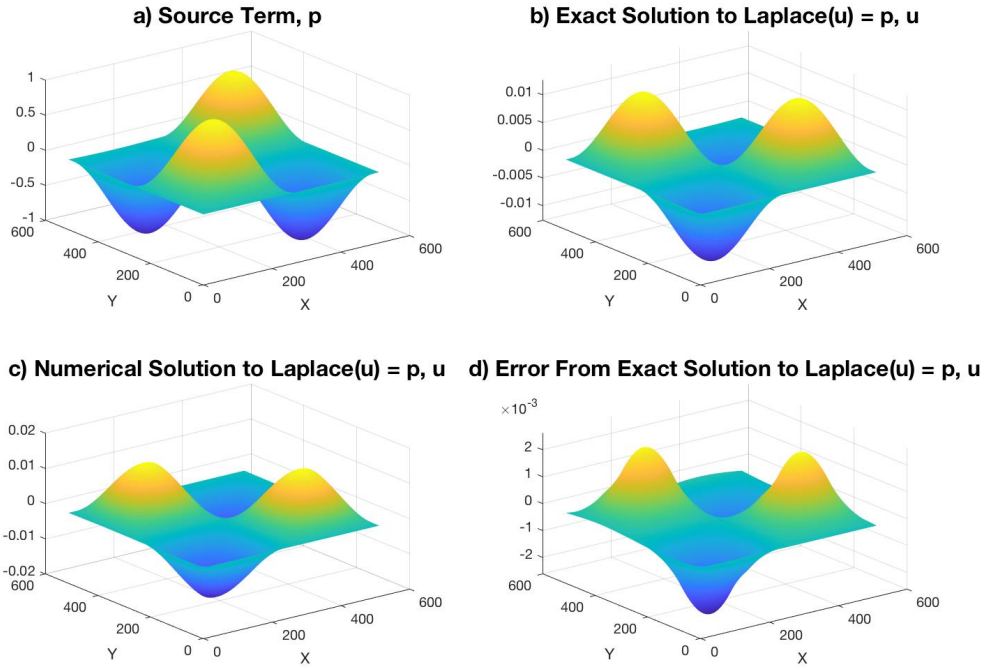

Figure 1: Results from the FACR solver run on a grid size of 513x513. **a)** Source term for the Poisson equation. **b)** Exact solution to the Poisson equation. **c)** Our numerical solution using FACR solver. **d)** The error of our numerical solution compared to the exact solution.

This solver was able to run on grid sizes up to 513x513, corresponding to $f = 9$; above this size there was an issue of NAN values being produced. Due to the time constraints of this project this bug has not yet been resolved. We have nonetheless demonstrated that the solver is capable of performing on non-trivial grid sizes, and have produced convergence plots for the solution error as well as algorithm run-time.

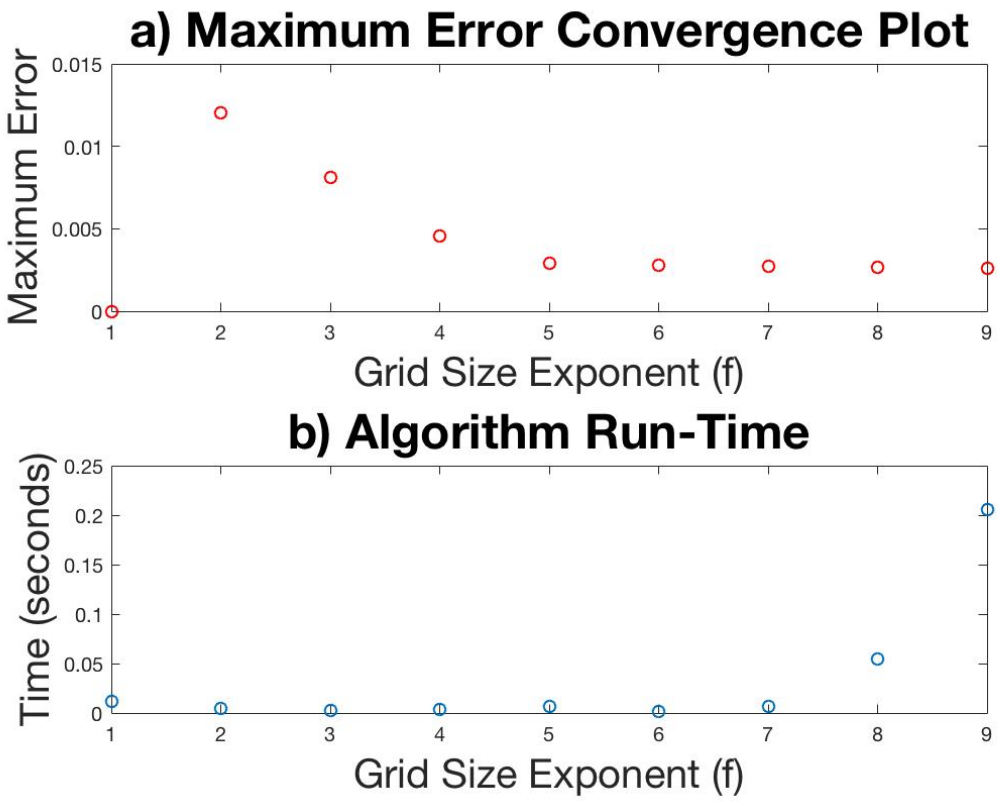

Figure 2: Analysis of FACR solver performance as a function of grid size, which goes as $(2^f + 1)$. **a)** Solution error compared to exact solution. This error value was computed by taking the error at each location on the grid and reporting the maximum value. **b)** Execution time to solve the Poisson equation.

The error plot 2a) shows a convergence with increasing grid resolution over the first few values, however it then remains approximately constant over the remaining values. The cause of this bug is unknown, as the error should converge to zero as the grid resolution grows in order for the numerical solution to be consistent with the exact solution.

Similarly, the run time is approximately constant until $f = 8$, when it starts to increase exponentially. Although grid sizes $f > 10$ fail to produce a correct solution due to the unresolved NAN bug discussed earlier, the run time continues to follow this trend of exponential growth as seen in Fig. 3.

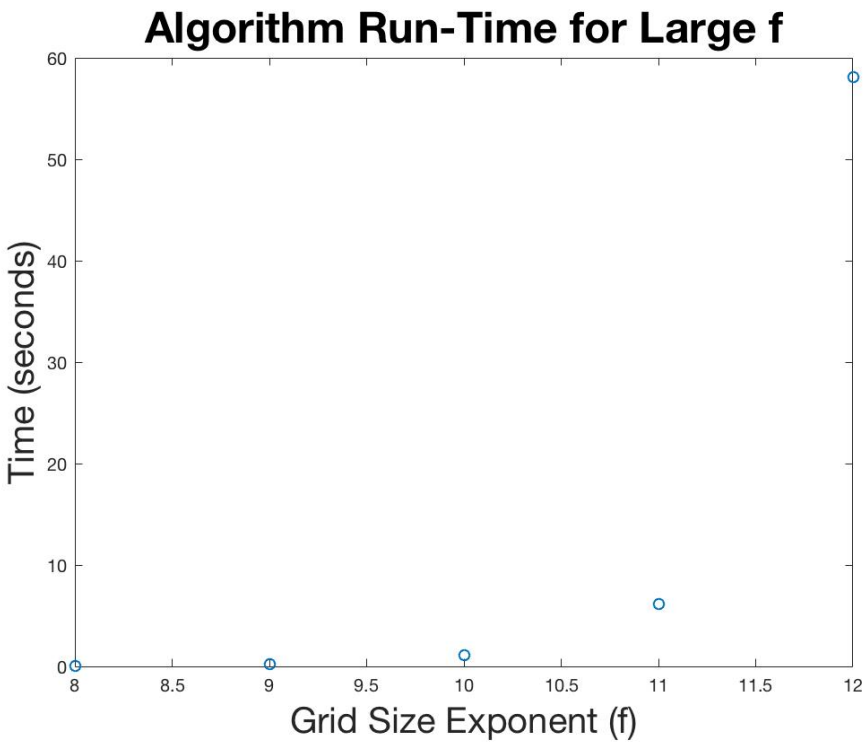

Figure 3: Analysis of FACR solver execution time as a function of grid size, which goes as $(2^f + 1)$. Plotted over the region of interest, where exponential growth begins.

4 Conclusions

In this project we implemented and tested a FACR algorithm for solving the 2D Poisson equation with Dirichlet boundary conditions in the y-direction and periodic boundary conditions in the x-direction. While this algorithm seems to still have some bugs which affect the convergence behavior as well as applicability to grids larger than 513x513, it provides reasonable run-time results and appears to have an accuracy of roughly $10^{-3} - 10^{-4}$ over the tested range of grid sizes.

It is important to note that, while efficiency was taken into consideration in the implementation of this FACR solver, a rigorous effort to maximize this efficiency has not yet been undertaken. A few components of the code could likely be refined to decrease run-time. The method used to find the stencil points needed to set up the tridiagonal system of Eq. 6 has one such component: a number of for-loops are used to iterate through the list of central y-coordinates for the reduced systems at each level. It may be possible to replace this imple-

mentation with a vectorized approach which is more efficient than for-loops. Alternatively, when using this solver in a larger simulation code which will need to solve the Poisson equation numerous times on the same grid, the stencil points could be computed once during the initial call and then stored for easy access during future calls. This would greatly reduce the contribution of this process to the run-time as the number of calls necessary for a simulation increases. A for-loop is also used to generate $\mathbf{T^{(n)}}$ and $\mathbf{p_j^{(n)}}$ for all of the $n = 1 : f$ levels, and this too could likely be made more efficient although it could not be pre-computed since it is dependent on the source term which will change each time the solver is called.

Additionally, this implementation uses the simplest case of FACR, with $r = 0$ such that all of the rows of the grid are computed using FACR. Stopping before all of the rows have been filled in and then using a method such as a finite-difference or Fourier Transform solver for the intermediate rows could further improve the algorithm speed.

References

[1] Olivier Mercier and Jean-Christophe Nave, *The Characteristic Mapping Method for the Linear Advection of Arbitrary Sets* (September 2013) 1

[2] Xi-Yuan Yin, Olivier Mercier, Badal Yadav, Kai Schneider, Jean-Christophe Nave, *A Characteristic Mapping Method for the Two-Dimensional Incompressible Euler Equations* (2019) 1, 2

[3] Noah LeFrancois, Xi-Yuan Yin, Jean-Christophe Nave, *Characteristic Mapping Method for Smoke Simulation with the Euler Equations* (2019) 1, 2

[4] William H. Press et. al. *Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition*. Cambridge University Press. pp. 814-815 (1992) 2, 3

[5] Paul N. Swarztrauber *The Methods of Cyclic Reduction, Fourier Analysis and the FACR Algorithm for the Discrete Solution of Poisson's Equation on a Rectangle*. SIAM Review, Vol. 19, No. 3 : pp. 490-501 (1977)