Michael D. Walker (mw6136), Philip Satterthwaite (ps1639), Michael Schroeder (ms2774)
APC523: Numerical Algorithms for Scientific Computing
March 26, 2024

Final Project Proposal: Comparing Numerical Methods on Cylindrical Wave Equation Solver Performance

1. Introduction

The ability to understand and predict surface deformation is of great real-world importance in applications ranging from predicting imminent volcanic eruptions to understanding how the surface of a fluid will move under external forces [1]. Understanding these deformations often requires solving high-order partial differential equations, a task that is sometimes impossible to do by hand (e.g., attaining an analytical solution to the Navier-Stokes equation). As a result, scientists and engineers are often left to solving PDEs numerically. This task can quickly become computationally expensive, and it is thus necessary to implement various numerical algorithms to optimize the solutions' speed, cost, accuracy, and stability.

In this project, we proposed to investigate the trade-offs between various numerical methods' speed, cost, accuracy, and stability, as they solve the wave equation in cylindrical coordinates. We will use iterative methods such as Jacobi, Gauss-Seidel, Successive Over-Relaxation, and Crank-Nicolson. Additionally, as time permits, we will also compare Forward and Backward Euler methods, spectral methods, and various Runge-Kutta schemes. Further, we would like to compare the performance of these methods when a damping term is included in the equation, at the resonance frequency, and as the timestep varies.

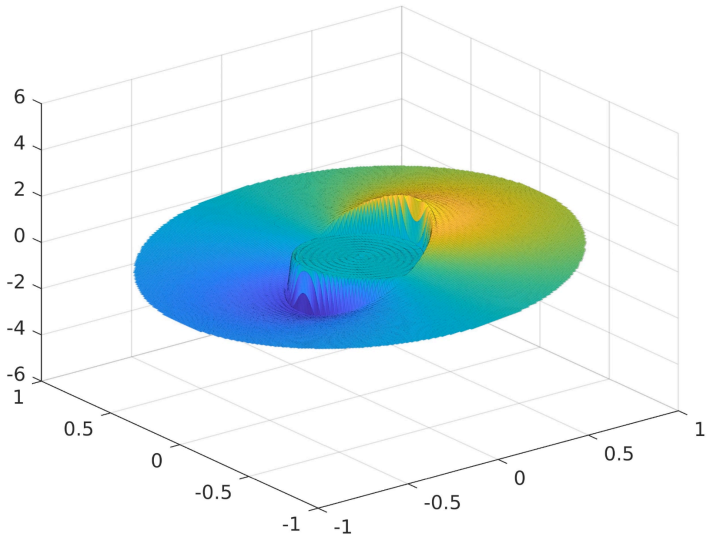

FIGURE 1. Example solution to the cylindrical wave equation.

2. The Wave Equation in Cylindrical Coordinates

For a small deformation of any planar circular elastic material, the surface deformation, φ, satisfies the wave equation in plane-polar coordinates

$$\frac{\partial^2 \varphi}{\partial t^2} = c^2 \left[\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial \varphi}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 \varphi}{\partial \theta^2} \right]$$

where the driving force of the wave motion for a material initially at rest can be specified through the boundary and initial conditions. Even with inhomogeneous boundary conditions, such an equation can be solved analytically using methods of separation of variables and eigenfunction expansion. Seeking to model waves on the surface of the fluid generated by the motion of the walls, we choose the following boundary and initial conditions.

$\varphi(r = R, \theta, t) = A\cos(\omega t)\cos(\theta)$
$\varphi(r = 0, \theta, t) \to$ finite
$\varphi(r, \theta + 2\pi, t) = \varphi(r, \theta, t)$
$\varphi(r, \theta, t = 0) = 0$
$\partial\varphi/\partial t\,(r, \theta, t = 0) = 0$

These conditions lead to the analytical solution

$$\varphi(r, \theta, t) = A\left(\frac{r}{R}\right)^2 \cos(\omega t)\cos(\theta) + \tilde{\varphi}(r, \theta, t)$$

$\tilde{\varphi}$ is composed as follows, where $J_\alpha(\cdot)$ is the Bessel function of the 1st kind and order α, η is a variable of integration, and λ are the eigenfunctions for each of the modes.

$$\tilde{\varphi}(r, \theta, t) = \sum_{m=1}^{\infty} \left(\frac{1}{c^2\lambda_{m,n} - \omega^2} \frac{2A}{R^2} \frac{\int_0^R \left(\omega^2\eta^2 + 3c^2\right) \eta J_1\left(\sqrt{\lambda_{1,m}}R\eta\right) d\eta}{J_2^2\left(\sqrt{\lambda_{1,m}}R\right)} \cos(\omega t) \ \ldots \right.$$

$$- \frac{2A}{J_2^2\left(\sqrt{\lambda_{1,m}}R\right)} \left(\int_0^R \eta^3 J_1\left(\sqrt{\lambda_{1,m}}R\eta\right) d\eta \ \ldots \right.$$

$$\left. \left. + \frac{1}{c^2\lambda_{m,n} - \omega^2} \frac{1}{R^2} \int_0^R \left(\omega^2\eta^2 + 3c^2\right) \eta J_1\left(\sqrt{\lambda_{1,m}}R\eta\right) d\eta \right) \cos\left(c\sqrt{\lambda_{1,m}}t\right) \right) \cos(\theta) J_1\left(\sqrt{\lambda_{1,m}}r\right).$$

3. **Iterative Methods**

3.1. **Jacobi Iteration.** To implement a Jacobi iterative scheme, a new displacement array is created. The difference between the new displacement and old displacement at each location is calculated, and the maximum of this difference over the entire grid is taken to be the error of the iteration. The arrays are then swapped and the process is repeated until the error is less than the defined tolerance.

3.2. **Gauss-Seidel Iteration.** For a Gauss-Seidel iterative scheme, the process is very similar, however values to the left of the cell being updated are evaluated with the already updated value.

3.3. **Successive Over-Relaxation.** For Successive Over-Relaxation (SOR), the initial process is similar to the Gauss-Seidel iteration, however there is an additional step where an over-relaxation parameter Ω is defined and varied by the user to determine the optimal value for a particular set of equations.

4. **Time-Stepping Methods**

These methods discretize the time domain into small intervals and approximate the solution at each time step. They iteratively advance the solution from an initial condition to a desired final time, updating the solution at each time step based on the governing equations.

4.1. **Forward Euler.** The Forward Euler method is a numerical technique for solving ordinary differential equations by approximating the solution at the next time step based on the derivative at the current time step, often used for its simplicity but limited by its first-order accuracy and potential stability issues.

4.2. **Backward Euler.** This method is implicit, so the solution at the next time step is approximated based on the derivative at the future time step. Unlike Forward Euler, it is unconditionally stable but is more computationally expensive.

4.3. **Runge-Kutta.** This method iteratively approximates the solution at each time step using a weighted average of multiple intermediate slope estimates. It is significantly more accurate than other similar methods. We plan implement and compare RK2, RK3, and RK4.

4.4. **Crank-Nicolson.** Crank-Nicolson is used for time-dependent partial differential equations. It solves the equations by using the average of the implicit and explicit Euler methods, making it stable and second-order accurate in time and space. It is especially accurate for parabolic PDEs.

5. Spectral Methods

These methods are employed for solving differential equations by representing the solution as a sum of basis functions. They offer high accuracy and convergence rates, making them particularly suited for problems with smooth solutions or periodic boundary conditions.

5.1. **Fourier Series.** Fourier series are used in spectral methods to represent periodic functions as an infinite sum of sinusoidal functions. By expanding the solution in terms of Fourier modes, spectral methods can efficiently capture periodic behavior and rapidly converge to the solution, especially for problems with periodic boundary conditions such as ours.

5.2. **Chebyshev Polynomials.** Chebyshev polynomials are orthogonal polynomials defined on the interval [-1, 1]. They are usually employed in spectral methods to approximate non-periodic functions or to handle problems with non-uniform spatial grids. The use of Chebyshev polynomials allows for high accuracy and rapid convergence.

6. Possible Project Expansions

Given enough time, we will seek to expand the project in three ways. First, it would be interesting to investigate the impact of time step sizes on stability, convergence, and precision. Second, we would like to add a damping term to the equation to determine how different fluids (different viscosities) may impact the resulting profile. Third, we could observe the effect of forcing the outside wall at the system's resonant frequency.

7. Personal Benefits

This project will allow us to develop further proficiency multiple aspects of the numerical method topics of this course. We work in a computational group (the Computational Turbulent Reacting Flow Laboratory, CTRFL), and will seek to incorporate any lessons learned. While not relevant to this project, the course content in massively parallel architectures was also extremely relevant to our research. The flow and combustion solvers used in CTRFL are almost entirely written in `Fortran`. When appropriate, we will use `Fortran` to write our numerical solver functions, giving us more exposure to its syntax and code format that will be greatly useful to us in our future research.

References

[1] Volcano Hazards Program. Movement on the surface provides information about the subsurface. https://www.usgs.gov/programs/VHP/movement-surface-provides-information-about-subsurface.