Project 1: Flow over a square

AE 523, Computational Fluid Dynamics, Fall 2020

Due: October 9, 11:55pm, electronically via Canvas

1 Problem Description

A flow that is incompressible and irrotational can be modeled by Laplace's equation for a scalar velocity potential. In two dimensions, an equivalent formulation is obtained using a stream function, $\psi(x, y)$, from which the velocity components are given by

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x}. \tag{1}$$

With this definition, continuity is automatically satisfied. Requiring the flow to be irrotational leads to Laplace's equation for ψ,

$$\nabla^2 \psi = 0. \tag{2}$$

In this project, you will be solving for two-dimensional potential flow around a square, as illustrated in Figure 1.

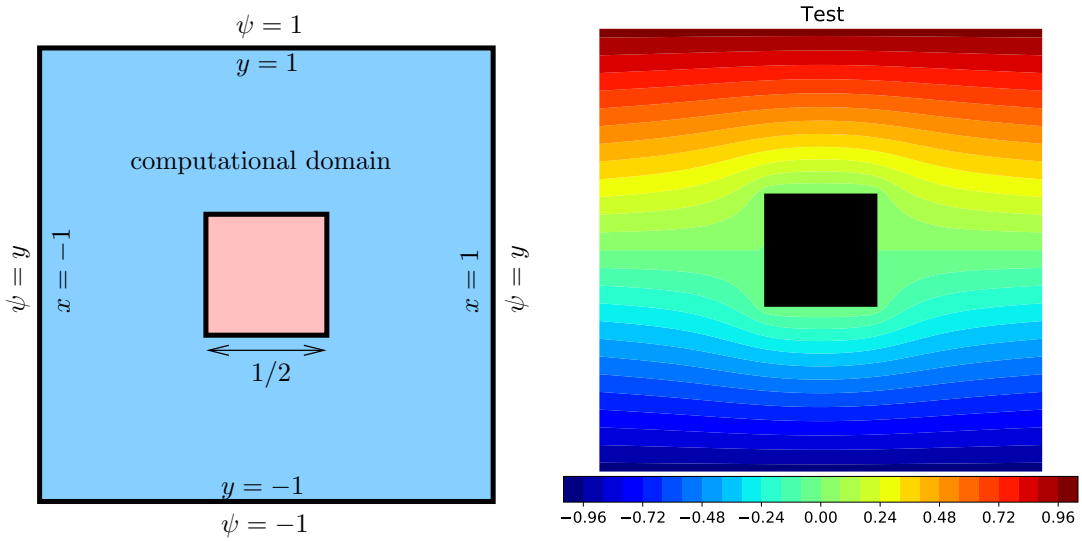

Figure 1: Solution of potential flow around a square, using a stream function, $\psi(x, y)$.

The computational domain is a square, $(x, y) \in [-1, 1]^2$, and the boundary conditions are Dirichlet. On the bottom and top walls, ψ is set to -1 and +1, respectively, and on the sides, ψ is set to y. The inner square, of side length $\frac{1}{2}$ and placed in the center of the outer square,presents an obstacle to the flow, and by symmetry it corresponds to a streamline on which $\psi = 0$.

2 Discretization

You will use a finite-difference method to solve Equation 2 on the computational domain. The grid consists of a lattice of $(N + 1)^2$ points, as shown in Figure 2. Some of these points are on the boundary, and some are inside the inner square, hence outside the computational domain. It is up

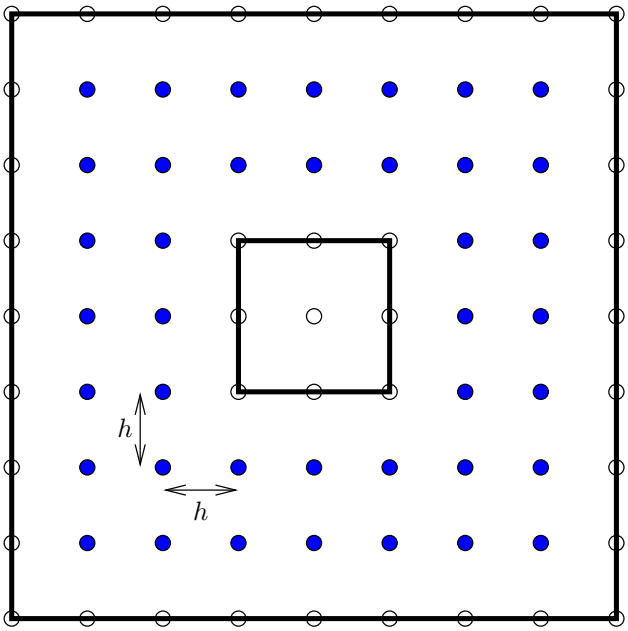

Figure 2: Finite difference grid for $p = 0 \Rightarrow N = 8$.

to you to decide how best to deal with these points. N is the number of intervals across the entire domain, so that the spacing is $\Delta x = \Delta y = h = 2/N$.

To ensures that the grid conforms to the boundary of the inner square, use

$$N = 2^{p+3}, \qquad p = [0, 1, 2, \ldots]. \tag{3}$$

At each interior node, use a standard second-order five-point stencil to discretize the Laplacian in Equation 2. Note that the given Dirichlet conditions fully specify ψ on the boundaries.

3 Solvers

You will implement three types of solvers:

1. **Direct**: Build a *sparse* linear system of equations to solve for the nodal states. The system will take the form

   $$\mathbf{A}\boldsymbol{\Psi} = \mathbf{F},$$

   where $\boldsymbol{\Psi}$ is the unrolled state vector of unknowns. Solve this system using a sparse direct solver, such as the backslash operator in Matlab or `scipy.sparse.linalg.spsolve()`.

2. **Iterative smoothers**: Implement two iterative smoothers: under-relaxed Jacobi and over-relaxed Gauss-Seidel. For Gauss-Seidel, you will use the "red-black" ordering, in which the nodes are colored in checkerboard fashion and the smoother is applied first to the red nodes and then to the black nodes. The presence of the inner square does not change the red-black ordering: imagine a checkerboard with the center cut out. You will use under-relaxation for Jacobi and over-relaxation for Gauss-Seidel. *Do not implement these smoothers in matrix form. That is, you should not have any matrices, full or sparse, of size that scales with the number of unknowns.*

3. **Multigrid**: Implement a V-cycle multigrid solver, in which successively finer grids are obtained by increasing p in Equation 3. Use full-weighting for the restriction operator, I_{2h}^{h}, and interpolation for the prolongation operator, I_{h}^{2h}. On the down/up sweep of the V-cycle perform $\nu_1 = \nu_2 = 2$ pre/post smoothing iterations. On the coarsest grid, perform $\nu_c = 50$ smoothing iterations. For the initial conditions in all iterative runs, use $\psi = 0$ at the interior nodes. For the smoother, use Gauss-Seidel with an over-relaxation factor of $\omega = 1.5$. *Again, implement these without any matrices whose size scales with the number of unknowns.*

4 Post-processing

To visualize the flowfield, plot contours of the stream function, as shown in Figure 1. These contours are streamlines of the flow. The top and bottom of the domain represent impenetrable walls, since ψ is set to a constant there. Acceleration of the fluid around the square changes the pressure of the fluid, and this affects the force on the walls. Of interest will be the pressure coefficient distribution on the bottom wall,

$$c_p(x) = 1 - \frac{u^2}{U_\infty^2}, \tag{4}$$

where the free-stream speed (speed of the flow without the inner square present) is $U_\infty = 1$ based on the boundary conditions. Note that u is obtained by differentiating ψ, according to Equation 1. Use a second-order one-sided finite difference of the ψ data to obtain u at the bottom boundary nodes.

Integrating and normalizing the pressure coefficient gives the lift coefficient on the bottom wall,

$$c_\ell = \frac{1}{2}\int_0^2 c_p(x)\,dx. \tag{5}$$

Use the trapezoidal method to perform this integration.

5 Questions and Tasks

1. **[20%]** Write a program to set up and solve, in sparse matrix form, the finite-difference system for this problem for a given p. The result of the solve is the value of ψ at the grid points. Print out, as a 9×9 matrix (columns corresponding to different x locations), the resulting nodal ψ values for a run with $p = 0$.
2. **[20%]** Write post-processing functions/programs to generate plots of the pressure coefficient along the bottom wall and to compute the lift coefficient. On a single figure, plot $-c_p(x)$ for runs with $p = 0, 2, 4$. Next, compute and report the lift coefficients for $p = 0, 1, 2, 3, 4, 5$. Using c_ℓ at $p = 7$ as the exact value (report this too), make a "semi-log-y" plot of the lift coefficient error versus p. What convergence rate do you observe? Discuss this result.
3. **[20%]** Implement the Jacobi and Gauss-Seidel smoothers. Plot the L_2 residual norm convergence history of the Jacobi smoother for $p = 3$, using under-relaxation factors of $\omega = 0.3, 0.6, 1.0$. Do this for the first few hundred iterations and discuss the effect of under-relaxation. Next, make a separate plot of the L_2 residual norm convergence history of Gauss-Seidel using $\omega = 0.5, 1, 1.5$. Start from zero interior-node initial conditions for all of your runs. Discuss how the two histories compare and the effects of under-relaxation.
4. **[15%]** Implement a V-cycle multigrid method that will work starting from an arbitrary grid

level p, with two or more grid levels. Describe your implementation, including the restriction and prolongation operators.

a. **[7%]** Plot the L_2 residual norm convergence of the multigrid method (norm versus multigrid iteration) with a fixed coarse grid of $p = 1$, for $2, 3, 4$, and 5 levels. How does the convergence rate behave as the number of levels is increased?

b. **[8%]** For $p = 0$ as the coarsest level and $p = 4$ as the finest, investigate the behavior of multigrid by varying pre/post/coarse smoothing iterations. Try to determine the most efficient setting of smoothing iteration numbers for multigrid, and make one or more plots with several convergence histories to demonstrate your investigation. For a fair comparison, plot the L_2 residual norm convergence against *work units* (WU), where 1 WU corresponds to the cost of one smoothing iteration on the finest grid. Note that in this two-dimensional problem, going down to a coarser level decreases the cost of smoothing by a factor of 4. Also include (overlay) a convergence plot of pure Gauss-Seidel ($\omega = 1.5$) on the finest level for comparison.

6 Deliverables

You should turn in, electronically via Canvas,

1. A technical report as a `.pdf` file that describes your methods and results, and that addresses all of the above tasks. The report should be professional, complete, and concise. **10%** of your grade will be determined by the professionalism (neatness, labeling of axes, spelling, etc.) of your report.
2. Documented source files of all codes you wrote for this project, as one `.zip` archive.

This is an individual assignment. You can discuss the project at a high level with each other, but you must turn in your own work.