AME 535a Introduction to Computational Fluid Dynamics
University of Southern California – Fall 2018

Project 1: Finite Difference Method
Flow in a Non-Rectangular Channel

Handed Out: 01 Sept. 2018 *Due: 15 Sept. 2018*

Problem Statement

Consider the problem of optimizing the cross section of a channel. The cross section, shown in Figure 1, is assumed to have a *constant* perimeter

$$l \;=\; 2b + 2\sqrt{(c-b)^2 + h^2}\;, \tag{1}$$

which is directly proportional to the amount of material required. Since l is constant, the shape of the channel can be described in terms of two parameters: the height of the cross section, h, and the half width of the base, b. Given b and h, the geometrical parameter c can be calculated as

$$c \;=\; b + \sqrt{\frac{(l-2b)^2}{4} - h^2}\;. \tag{2}$$

We are interested in two outputs: the flow rate, Q, defined as the integral over the cross section Φ of the velocity u normal to the cross section given by

$$Q \;=\; \iint_{\Phi} u\, dx\, dy\;, \tag{3}$$

and the moment of inertia of the cross section I. The latter can be related to the deflection of the channel, due to the loading of the flowing fluid. The trapezoidal cross section has a thickness $t = 0.05$, and its moment of inertia I with respect to the neutral axis can be calculated from

$$I \;=\; \frac{bt^3}{6} + \frac{bth^2}{2} + \frac{t\sqrt{(c-b)^2+h^2}}{6}\left[h^2 + t^2\frac{(c-b)^2}{(c-b)^2+h^2}\right] \tag{4}$$

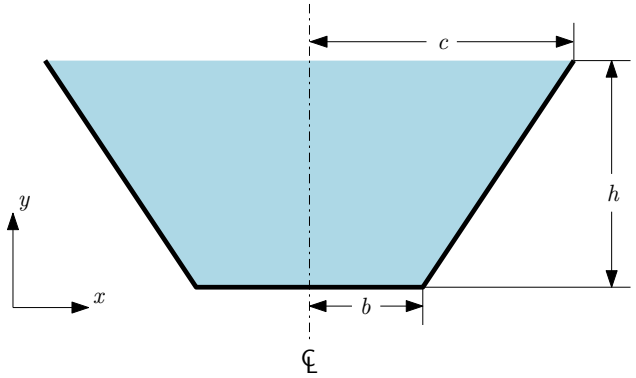

Figure 1: Channel cross section.

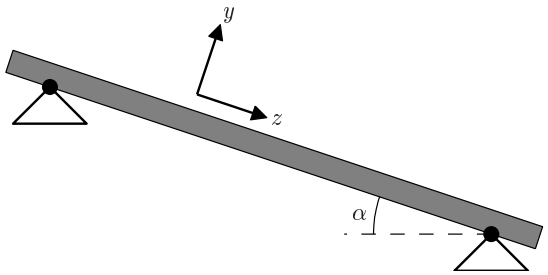

Figure 2: Flow channel.

The channel is at a slope of angle α with respect to the horizontal direction, as shown in Figure 2, and the horizontal component of the gravitational force creates the pressure gradient for the downward flow.

The governing equations are the steady Navier-Stokes equations,

$$\mathbf{u} \cdot \nabla \mathbf{u} \, + \, \nabla p \, = \, \mathbf{f} \, + \, \nu \nabla^2 \mathbf{u} \, . \tag{5}$$

With the assumption of fully developed flow, these can be reduced to Poisson's equation for the velocity component u normal to the channel cross section,

$$-\nabla^2 u \, = \, \frac{g \sin \alpha}{\nu} \, . \tag{6}$$

Here, g is the gravitational constant, and ν is the kinematic viscosity of the fluid. For simplicity, we assume that

$$\frac{g \sin \alpha}{\nu} \, = \, 1. \tag{7}$$

Along the walls of the channel the velocity is zero (no-slip condition), and on the free surface a zero-stress condition ($\partial u / \partial n = 0$) is assumed.

Numerical Procedure

We will solve Poisson's equation (6) using a finite difference procedure. Given the symmetry of the geometry, we only consider half the channel section, as shown in Figure 3. The following boundary conditions are applied on the original domain:

$$\begin{cases} \frac{\partial u}{\partial n} = 0 \, , & \text{on } AB \text{ and } AD \\ u = 0 \, , & \text{on } BC \text{ and } CD. \end{cases} \tag{8}$$

To solve the problem, we map the half domain Ω (with coordinates x, y) to a rectangular domain $\hat{\Omega}$ (with coordinates ξ, η), and solve the equations numerically using the finite difference method on $\hat{\Omega}$. We use a regular grid with $N \times N$ points in the computational domain, giving square cells of size $\Delta\xi = \Delta\eta = \dfrac{1}{N-1}$.

Questions

1) (10 pts) Introduce an analytical mapping T to transform the original domain Ω to the unit square computational domain $\hat{\Omega}$. Write the specific equation(s) you will solve and the corresponding boundary conditions in the computational domain.

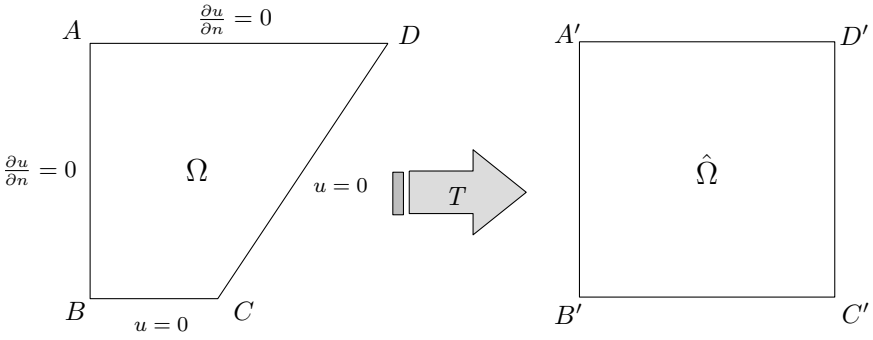

Figure 3: Geometrical transformation from the physical domain to the computational domain.

2) (20 pts) Use either Taylor series expansions, the method of undetermined coefficients, or Lagrange's interpolant method to derive second-order accurate finite-difference schemes for the discretization of *all* the derivative terms in the interior of the domain, *as well as* for the boundary points. You do not have to derive special schemes for the corner points: use the top boundary scheme at corner A, and use $u = 0$ at the other three corners. Also show how you will numerically evaluate the integral for the approximate flow rate $\hat{Q}$.

3) (15 pts) In a finite difference approximation, weighted combinations of the solution value at surrounding nodes are used to represent the governing differential equation at a given node. For node N, these weights are 'stamped' into appropriate positions in row N of the matrix. For a node N_{ij}, present the appropriate weights, $w_{i,j}$, $w_{i+1,j}$, $w_{i,j+1}$, *etc.*, which scale the solution, combining to form the approximate Laplacian operator in the interior. Present the weights for *not only* the interior region nodes *but also* the nodes in the boundary region of the domain.

4) (30 pts) Write a program that solves the problem. A skeleton code for this problem has been provided in `ChannelSkeleton.m` to help you get started, but its use is not mandatory. In addition, one may decide to use any programming language to complete this assignment. You are not required to create a code to solve the linear system $AU = F$: instead, use a MATLAB built-in solver (see hints) or a linear algebra library for the language you've chosen. Using the program you have developed, calculate the solution and $\hat{Q}$ for a grid size $N = 21$, with $l = 6.5$, $b = 1.0$, and $h = 2.0$. Compare your results with the sample solution in Figure 4. Please show your results and the comparison.

5) (5 pts) Comment briefly (several bullets) on how the program you wrote to solve the channel flow problem would be extended to form a general Poisson solver for an arbitrary 4-sided solution domain.

6) (10 pts) Calculate the convergence rate for the error in the output $\hat{Q}$, and for the L_∞ and the L_2 norms of the solution. Do the calculation for $l = 6.5$, $h = 2.0$, and $b = 0.5, 0.7$ (a total of 6 convergence plots), and verify that you obtain second-order convergence. Use the grid sizes $N = 11, 21, 41, 81$. Since we don't know the exact solution, use the solution for $N = 81$ as a reference. To calculate the L_∞ and the L_2 norms, restrict the solution obtained at the

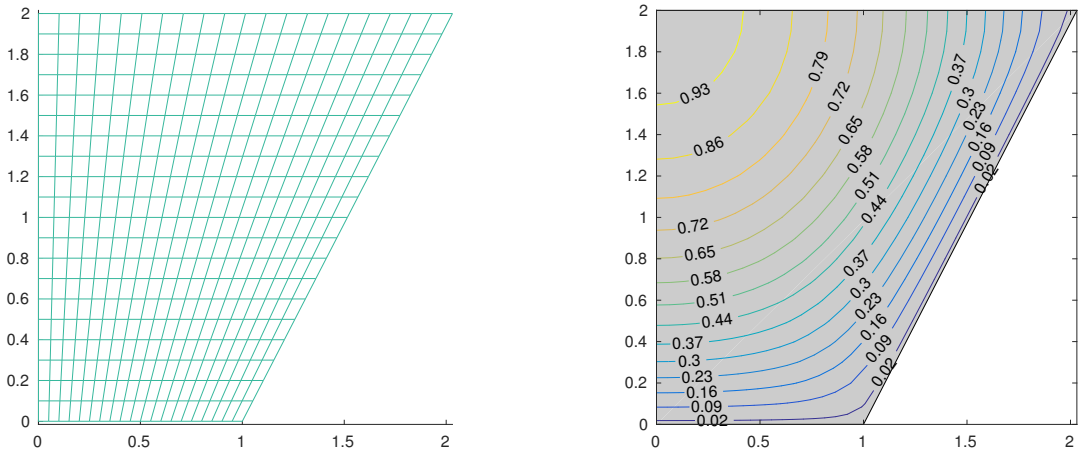

Figure 4: Sample solution, for $l = 6.5$, $h = 2.0$, $b = 1.0$, and $N = 21$. The plots show the grid (left) and contour lines of the solution (right). For these values, the flow rate is $\hat{Q} = 3.631$.

reference mesh to the current coarse mesh. Please plot your solutions on a semi-log or log-log plot where the errors are represented on the log scale. This is common practice, and should be applied to all future error plots even if it is not explicitly stated to do so.

7) (10pts) Keeping $l = 6.5$, vary the two inputs $h = [0.1, 2.0]$ and $b = [0.1, 2.0]$. Choose points on a regular grid in this two dimensional space, with constant interval 0.1 for both h and b. Each point, an (h, b) pair, represents a possible cross section. For each of these configurations, solve the problem using $N = 21$, and calculate the flow rate $\hat{Q}$ and the moment of inertia I. Create a graph, using the two outputs and calculate the convex hull of these points. The convex hull, known as the Pareto optimal frontier, is a trade-off curve: for a specific choice of one output, it determines the optimal choice for the other output. Assuming we are interested in maximizing the flow rate, what can we say about the resulting geometry? Explain the results you obtain for the Pareto front.

MATLAB **Hints**

- It is usually necessary to vectorize MATLAB code in order to get high performance. This means, for example, that for-loops should be avoided and replaced by higher-level constructs. We do not require you to do this, so feel free to use for-loops when building up matrices and vectors. This will make the code easier to read and understand, and these (relatively) small problems are fast enough anyway.

- A uniform grid with spacings `dxi`, `deta` can be created with

  ```
  [xi,eta]=ndgrid(0:dxi:1,0:deta:1);
  ```

  This gives two $N \times N$ arrays `xi`, `eta`.

- A grid with x, y coordinates in the $N \times N$ arrays `x`, `y` can be visualized using

  ```
  mesh(x,y,0*x,0*x)
  view(2), axis equal
  ```

- Remember to make A a sparse matrix (or you will run out of memory): `A=sparse(N^2,N^2);`

- When filling in the A matrix, it is convenient to use a mapping function: `map=reshape(1:N^2,N,N);` The position of the grid point `i`, `j` in the linear system of equations is then `map(i,j)`.

- The solution U to the linear system of equations $AU = F$ can be computed with `U=A\F`. If `A` is a sparse matrix, this will use the direct sparse solver in MATLAB. If you're using another programming language, you can use a linear algebra library to solve $AU = F$.

- The $N^2 \times 1$ solution vector `U` can be reshaped to an $N \times N$ array with `u=reshape(U,N,N);` This format is sometimes easier to work with, for example when computing the integral for $\hat{Q}$ and when plotting the solution.

- The contour plot in Figure 4 was created with

  ```
  patch([0,b,t,0],[0,0,h,h],-ones(1,4),0,'facecolor',[.8,.8,.8])
  [ccc,fff]=contour(x,y,u,0.02:0.07:max(u(:)));
  clabel(ccc,fff);
  axis equal
  ```

  First, the geometry is drawn, using the parameters `b`, `c`, and `h`. Then contour curves are made for the solution `u` on the grid `x`, `y`, all which are $N \times N$ arrays.

- Alternatively, you can create a color plot:

  ```
  surf(x,y,0*x,u)
  view(2),axis equal
  set(gcf,'renderer','zbuffer');
  shading interp, colorbar
  ```

- To plot the convex hull for the points in the vectors `Qs` and `Is`, type

  ```
  k=convhull(Qs(:),Is(:));
  plot(Qs(:),Is(:),'.',Qs(k),Is(k))
  ```