2.097J / 6.339J / 16.920J

Numerical Methods for Partial Differential Equations

Massachusetts Institute of Technology – Fall 2017

Project 1 – Finite Differences Methods

Handed Out: Sept. 11, 2017 *Due: Sept. 25, 2017*

---

1 Governing equations

Consider flow in a 2-D channel governed by the Euler equation. If the channel is rectangular, the flow would be uniform. Now assume that the shape at the bottom changes and a bump appears. If the bump is small, we could solve for the new flow by a perturbed Euler equation, which is linear. This linearized Euler equation has error of order $O(p'^2)$. It reads:

$$\frac{Dp'}{Dt} + \rho_0 c_0^2 \nabla \cdot \mathbf{u}' = 0$$
$$\rho_0 \frac{D\mathbf{u}'}{Dt} + \nabla p' = 0$$

For any variable s, we use s' to denote perturbation on s, s_0 the unperturbed property, and the actual value $s = s_0 + s'$. The variables could be $p, \mathbf{u}, \rho, c$, which are the pressure, velocity, density and sound speed. $\frac{D}{Dt}$ is defined as

$$\frac{D}{Dt} := \frac{\partial}{\partial t} + \mathbf{u}_0 \cdot \nabla$$

Taking the D/Dt of the first equation and the $\nabla\cdot$ of the second equation leads to governing equation in the flow field:

$$\frac{D^2 p'}{Dt^2} = c_0^2 \nabla \cdot \nabla p'$$

Let

$$q' = \frac{Dp'}{Dt}$$

we get

$$\begin{aligned} \frac{\partial p'}{\partial t} + \mathbf{u}_0 \cdot \nabla p' &= q' \\ \frac{\partial q'}{\partial t} + \mathbf{u}_0 \cdot \nabla q' &= c_0^2 \nabla \cdot \nabla p' \end{aligned} \tag{1}$$

The boundary conditions are also part of the governing equation. In terms of the actual solution, they read:

$$
\begin{aligned}
&p = p_0, \quad x = 0 \\
&\frac{dp}{dx} = 0, \quad x = L \\
&\nabla p \cdot \mathbf{n} = -\rho(F_{tt} + 2uF_{tx} + u^2 F_{xx}), \quad y = 0, H
\end{aligned}
$$

this boundary condition is analytical, means that it has no approximation in it. Here $y = F(x,t)$ is the geometry for the upper and lower wall. $\mathbf{n} = (\pm F_x, \mp 1)$ is the normal vector pointing into the flow field.

For q', the boundary condition is induced by that of p', by the definition of q'. The simplest form is:

$$
\begin{aligned}
&q' = 0, \quad x = 0 \\
&\frac{dq'}{dx} = 0, \quad x = L
\end{aligned}
\tag{2}
$$

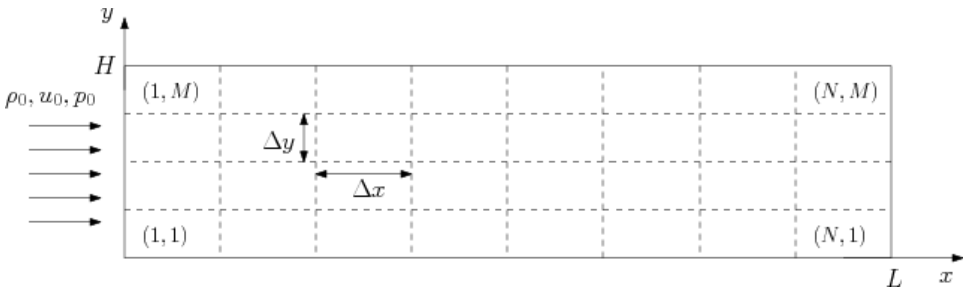
Although this boundary condition gives some reflection at the right boundary, it is still acceptable for this project.

2 Geometry, mesh and parameters

2.1 Uniform flow

The unperturbed problem is the uniform flow in a 2-D rectangular channel. The top and bottom sides are solid walls.

The domain is $[0, L] \times [0, H]$. It is discretized with N + 1 grid points in the x direction, M + 1 grid points in the y direction. The horizontal length of a cell is Δx, the vertical length is Δy.

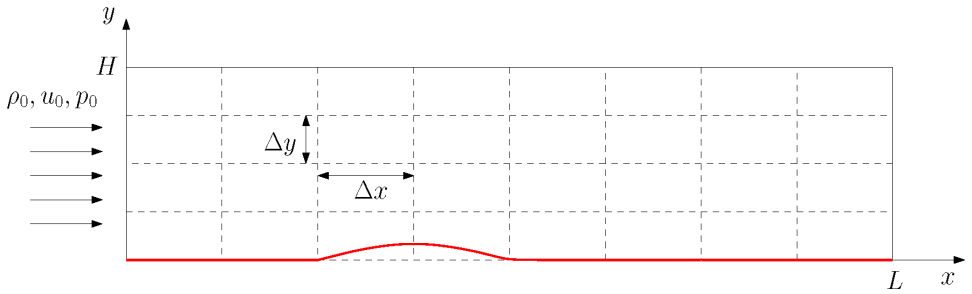


2.2 Flow perturbed by bump

Then assuming that the geometry at the bottom wall is perturbed and the shape function is:

$$y = \begin{cases} F(x), & x \in [\frac{L}{4}, \frac{L}{2}] \\ 0, & otherwise \end{cases}$$

Now the channel looks like:

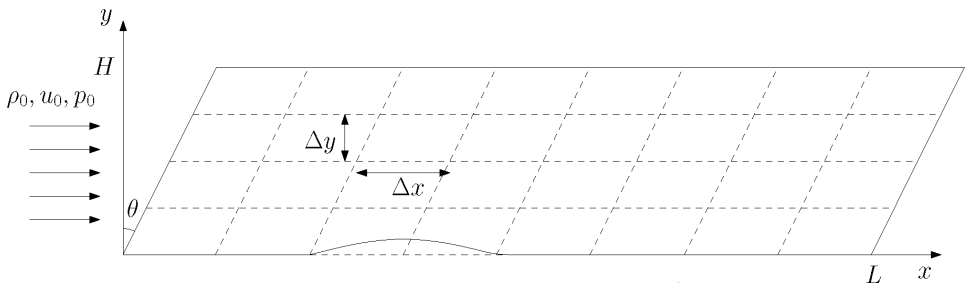


We could view the new flow as generated from the uniform flow, where the perturbation in the flow is induced by the small bump. Hence the equations for perturbed flow in above section could be used.

2.3 Oblique mesh

The vertical lines of the mesh could be rotated by an angle θ to create an oblique mesh, so that it lines up with the features of the solution. This type of mesh may improve the quality of the solution.

The oblique mesh looks like:



2.4 Values of parameters

Parameters of the incoming flow:

$$\rho_0 = 1, c_0 = 1, u_0 = \sqrt{2}$$

Parameters for the geometry:

$$L = 6, H = 2$$

$F(x), M, N$ and Initial state p', q' are input parameters.

3 Questions

1) a. (10 pts) Derive the governing boundary condition for p' on boundary. The error in the boundary condition you derived should be the same as that in the linearized Euler equation.

b. (20 pts) Derive a numerical scheme for the governing equation, using second-order finite-difference in space.

For this question, DO NOT substitute actual values into parameters in the final answer.

2) Write a code using the scheme derived in last question. You are allowed to use ode45 to do time integration.

a. (10 pts) On a bump described by function:

$$F(x) = 0.01\left(1 - \cos\left(\frac{8\pi}{L}(x - L/4)\right)\right)^2$$

Plot the contour for p' at time = 2 in your report (no figure generation in submitted code). Include the color-bar in the figure.

b. (10 pts) Your code will be tested by an automatic grader with certain boundary conditions and initial conditions for which the solution is known. Please program your solver so that arbitrary boundary and initial conditions are accepted.

3) (30 pts) Assume that now the oblique mesh as described in 2.3 is used.

a. (5 pts) Estimate what the twist angle θ should be, so that the mesh aligns with the features of the solution.

b. (10 pts) Derive the governing equation on this new mesh.

c. (15 pts) Derive a numerical scheme for the governing equation, using second-order finite-difference in space.

For this question, DO NOT substitute actual values into parameters in the final answer.

4) (20 pts) Same as question 2.

4 Submit requirement and input/output

- The same submission requirement still works.
- For question 2, write a MATLABfunction of the following signature:

  ```
  function [pp, x, y] = q2(dt, nstep, M, F, ppinitial, qpinitial)
  ```

  where the inputs and outputs are

  1) dt: scalar. A positive number Δt specifying the time spacing.

  2) nstep: scalar. Number of time steps calculated. The initial state is numbered 1, and nstep is the last time step, at which the pressure is outputted.

  3) M: scalar. Number of cells in y direction, as shown in the section 2.

  4) F: of size ($\frac{N}{4}$+1, 1). The hight of the bump, at uniformly distributed points in $[\frac{N}{4}, \frac{2N}{2}]$. You can use the size of this function to determine N.

  5) ppinitial: of size (N+1, M+1), initial value of p'.

  6) qpinitial: of size (N+1, M+1), initial value of q'.

  7) pp: of size (N+1, M+1). The output, p' at step nstep.

  8) x: of size (N+1, M+1). The output, x coordinate corresponding to p.

  9) y: of size (N+1, M+1). The output, y coordinate.

- For question 4, write a MATLABfunction of the following signature:

  ```
  function [pp, x, y] = q2(dt, nstep, M, F, ppinitial, qpinitial)
  ```

  The parameters have same definition as above.

A Knowledge on MATLAB

- 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, 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 M by N uniform grid with can be created with

  ```
  [xi, eta] = meshgrid(linspace(0,1,M), linspace(0,1,N));
  ```

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

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

  ```
  plot(x, y, 'k');
  hold on;
  plot(x', y', 'k');
  axis equal;
  ```

- Remember to make A a *sparse matrix* (or you will run out of memory): `A=sparse(N*M, N*M);` in MATLAB.

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

- In MATLAB, 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.

- The $N^2 \times 1$ solution vector `U` can be reshaped to an $N \times N$ array with `u=reshape(U,N,N);` (`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.

- A contour plot can be created with

```
plot(x(end,:), y(end,:), 'k');
hold on;
plot(x(:,1), y(:,1), 'k');
plot(x(:,end), y(:,end), 'k');
scale = linspace(0,max(u(:)),50);
[cc,hh]=contour(x,y,u,scale);
clabel(cc,hh);
axis equal;
```

First, the geometry is drawn. Then contour curves are made for the solution `u` on the grid `x`, `y`, all which are $M \times N$ arrays.