cylinderFlow - Documentation Manual

Nicholas C Crabb

May 26, 2017

Contents

Chapter 1

Introduction

1.1 Introduction

This project started as an evening/weekend project for me to kill time without wasting it. The initial goal was to build a two-dimensional finite element compressible/viscous Euler solver on a rectangle grid capable of potentially-transonic flow simulations for studying pressure gradients and distributions given localized pressure shocks in the flow domain. The reason for considering viscous effects is for numerical stability and because the prototype example on which this simplification is based considers highly-complicated flow geometries and exceptional air densities, so that viscous effects could potentially restrict the flow propagation.

1.2 Options

1.3 TODO's

Chapter 2

Graph Implementation

Since the meshes produced are Delaunay triangulations (see subsequent chapter), and these are sparse graphs, the graph is implemented using the list representation. More specifically, for better memory management, the `Mesh` class contains a STL vector of nodes and facets, and each node contains a STL vector of nodes as it's adjacency list.

2.1 `Mesh` class

The class `Mesh` is the main owner of the graph built from the input data grid points and corresponding facet list. The main tasks handled by `Mesh` include

1. input data parsing from the specified grid data file
2. performs divide and conquer Delaunay triangulation on grid points
3. builds facet list from triangulated graph
4. identifies which nodes are boundary nodes based on total angle of associated facets
5. writing formatted output of the resulting mesh to specified output file

2.2 `Node` class

This is the basic vertex object of the graph constructed in `Mesh`. It contains it's own adjacency list as well as it's $xy-$location and a unique ID integer, as well as many helper methods used by `Mesh` in building the Delaunay triangulation. The functions we will be evaluating on the mesh will hold values at these points, and as such our basis functions will be the piece-wise linear functions evaluating to 1 at a specific node and 0 on all other nodes (see Solver chapter for further detail).

2.3 `Facet` class

This is the basic triangle object that is used for partitioning the domain for finite-element expansion of the model functions.

2.4 Formatted I/O

The graph constructor and post-processor assumes a specific formatted input file. Shown is an example input file and corresponding output.

```
ID X Y
1 -1 0
2 -0.923879533 -0.382683432
3 -0.923879533 0.382683432
4 -0.707106781 -0.707106781
5 -0.707106781 0.707106781
6 -0.5 0
7 -0.382683432 -0.923879533
8 -0.382683432 0.923879533
9 -0.25 -0.433012702
10 -0.25 0.433012702
11 0 -1
12 0 0
13 0 1
14 0.25 -0.433012702
15 0.25 0.433012702
16 0.382683432 -0.923879533
17 0.382683432 0.923879533
18 0.5 0
19 0.707106781 -0.707106781
20 0.707106781 0.707106781
21 0.923879533 -0.382683432
22 0.923879533 0.382683432
23 1.0 0.0
```

Input file example. Format is single space delimited; first row includes descriptive header; first column a unique integer ID for each node, second column is $x-$location of node, third column is $y-$location of node.

```
NODE DATA:
ID ISBORD X Y ADJ
1 1 -1 0 2 3 6
2 1 -0.92388 -0.382683 1 4 9 6
3 1 -0.92388 0.382683 1 5 6 10
4 1 -0.707107 -0.707107 2 7 9
        .
        .
        .

BOUNDARY NODES:
1 2 3 4 5 7 8 11 13 16 17 19 20 21 22 23

FACET DATA:
ID AREA CENTROID VERTICES
1 0.0956709 (-0.80796,-0.127561) 1 2 6
2 0.0956709 (-0.80796,0.127561) 1 3 6
3 0.103856 (-0.626995,-0.507601) 2 4 9
4 0.103856 (-0.626995,-0.507601) 2 9 4
        .
        .
        .

NODE VERTEX DATA:
NODE_ID FACET_IDS
1 1 2
2 1 3 4 5
3 2 6 7 8
4 3 4 11 12
        .
        .
        .
```

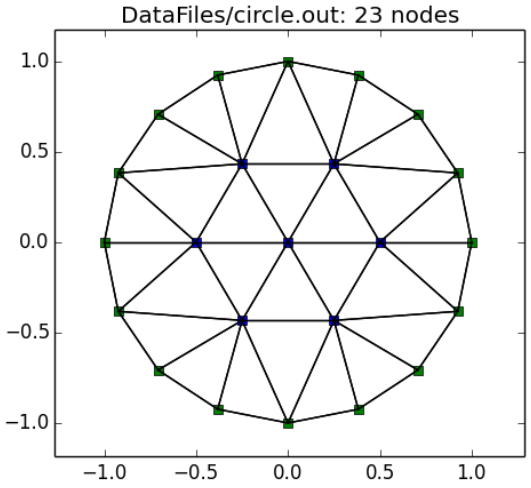
Output file example (data blocks have been truncated with ellipses for aesthetics). Format is single space delimited, in four blocks.

First block echoes input along with a bool identifier if the node is a boundary node (in the second column), as well as prints the unique ID of the adjacent nodes in the last (variable number of-) columns.

Second block is just a list (by ID) of which nodes were identified as boundary nodes.

Third block is the facet data dump. First column is a unique identifying integer for each facet; second column is the computed area of the facet; third column is the $xy-$coordinates of the facet's centroid; the last three columns are the unique IDs of the nodes that make up the vertices of the facet.

The last block is a reiteration of the third block, but from the node's perspective. The first column is the node ID, the remaining columns are the unique facet IDs for which the node is a vertex.



Resulting mesh visualization of given example input file.

Chapter 3

Meshing

3.1 Delaunay triangulation - divide and conquer

The divide and conquer approach to triangulation, due to *[ref]*, is implemented here.

3.2 Building the facet list from triangulated graph

This is essentially an implementation of finding all unique 3-cycles within the triangulated graph.

Chapter 4

Solver

4.1 Motivating model

The original motivation for building this software was to provide an efficient solver on general domains Ω for the conservation equations of mass, momentum and total energy density in a flow field $\mathbf{u} = (u, v)$

$$\mathbf{U}_t + \nabla \cdot \mathbf{F} = \mathbf{0},$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p - (\nu + \lambda)u_x - \lambda v_y & \rho uv - \nu u_y \\ \rho uv - \nu v_x & \rho v^2 + p - (\nu + \lambda)v_y - \lambda u_x \\ (E + p)u & (E + p)v \end{bmatrix},$$

closing the system with the ideal gas law $p = \frac{R}{c_v}(E - \frac{1}{2}\rho|\mathbf{u}|^2)$. We note the following definitions for clarity:

TODO: put in definitions here...

Note that for $\nu = 0$ we have the compressible Euler equations, otherwise we have the conservation equations in the presence of fluid viscosity and linear elasticity.

4.2 Setting up the piece-wise linear basis

For each node $N_i = (x_i, y_i)$ in the mesh, we define a piece-wise linear function $\varphi_i(x, y)$ such that $\varphi_i(N_i) = 1$, and $\varphi_i(N_j) = 0$ for all other nodes $N_j \neq N_i$. That is, supp φ_i is the union of all facets F_k for which N_i is a vertex, and to compute φ_i, we must then compute for each facet F_k the equation of the plane through its vertices, say $N_i, N_{i'}, N_{i''}$, that is at $z = 1$ at N_i and $z = 0$ at the other two. This can be done directly with use of the plane's normal vector $n = (n_1, n_2, n_3) = \overline{N_i N_{i'}} \times \overline{N_i N_{i''}}$:

$$\varphi_i\Big|_{F_k} = \varphi_{i,k}(x, y) = 1 - \frac{n_1}{n_3}(x - x_i) - \frac{n_2}{n_3}(y - y_i).$$

We then also immediately have the gradient for φ_i on this facet

$$\nabla\varphi_i\Big|_{F_k} = \nabla\varphi_{i,k} = \left(-\frac{n_1}{n_3}, -\frac{n_2}{n_3}\right).$$

With these functions $\{\varphi_i\}$, we can now express any function's $f(x, y)$ piece-wise linear approximation

$$f(x, y) \approx \sum_i a_i \varphi_i.$$

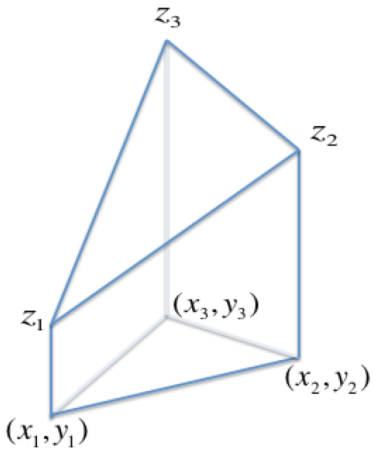
4.3 Weak formulation of motivating model

The first hurdle in formulating the weak form of the equations is how to integrating the basis functions. The challenge arises because if we want to use an arbitrary mesh, then we cannot assume any special structure of the facets. That is, we want to be able compute directly

$$\langle \varphi_i, \varphi_j \rangle = \int_\Omega \varphi_i \varphi_j dA$$

for any pair i, j. Of course a motivating factor for choosing the finite element basis as described above is that this quantity is only nonzero when N_i, N_j are adjacent in the mesh. Thus, as with computing the formulae for φ_i above, let's focus our attention on a single facet, and the same will apply to each facet for which both N_i, N_j are vertices.

In general, since we are assuming that all our functions are approximated as piece-wise linear, let's discuss how we might integrate a general such function over a general facet.



Integrating a general element.

First, to simplify the computation we can perform a change of variables transformation to the triangle T with vertices $(0,0),(0,1),(1,0)$. To ensure the transformation is orientation-preserving, let's first order the facet's vertices in a counter-clockwise fashion. This can be accomplished by first finding the facet centroid $c = (c_x, c_y)$, and order by angle using the dot product: define $e_1 = (1,0)$, then

$$v_i \leftarrow (x_i - c_x, y_i - c_x) \longrightarrow \theta_i = \begin{cases} \arccos(v_i \cdot e_1), & \text{if } y_i \geq c_y \\ 2\pi - \arccos(v_i \cdot e_1), & \text{otherwise} \end{cases},$$

then ordering by θ_i. We can then evaluate the transformed integral:

$$\int_{F_k} f(x,y)dxdy = \int_T f(u,v)\left|\frac{\partial(x,y)}{\partial(u,v)}\right|dudv,$$

where we have

$$\begin{bmatrix} u \\ v \end{bmatrix} = b + A(x,y) = \begin{bmatrix} (y_1 - y_2)x_1 + (x_2 - x_1)y_1 \\ (y_2 - y_3)x_2 + (x_3 - x_2)y_2 \end{bmatrix} + \begin{bmatrix} y_1 - y_2 & x_2 - x_1 \\ y_2 - y_3 & x_3 - x_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

so that $\left|\frac{\partial(x,y)}{\partial(u,v)}\right| = \det(A^{-1})$. This transformation takes $(x_1, y_1) \mapsto (0,1), (x_2, y_2) \mapsto (0,0), (x_3, y_3) \mapsto (1,0)$. Thus we can compute u, v in terms of the (x_i, y_i), compute the tranformed function values from b, A, compute A^{-1} directly. Since A is independent of x, y, we can pull its determinant out of the integral, so that without loss of generality we can assume that $F_k = T$, then

just scale the result by $\det(A^{-1})$. By direct calculation we can then express the integral as an "exact" $O(1)$ computation

$$\int_T p(x,y)dA = \frac{1}{2}\left(z_3 - \frac{n_1}{3n_3} - \frac{n_2}{3n_3}\right),$$

when $n_3 \neq 0$ where $n = (n_1, n_2, n_3) = w_1 \times w_3$ is the plane p's normal vector ($w_i = (x_i - x_2, y_i - y_2, z_i - z_2)$). The integral is equal to $z_3/2$ if $n_3 = 0$.

4.4 Boundary conditions enforcement

Chapter 5

Graphical User Interface