Alan Abraham

Myles Mwathe

Yao Xiao

Finite Element Model Proposal

1. Integrate Triangle into FEGrid

Homework had the **.node** and **.ele** data provided as input files. These files were originally generated by the **Triangle** program **http://www.cs.cmu.edu/~quake/triangle.html**. FEGrid will be improved to also allow for a constructor that accepts a **Triangle** polygonal input which is selected if a user gives a **.poly** command-line argument and a floating-point value for maximum triangle area.

2. Template FEPoissonOperator by data type

This will also necessitate templating other classes and upgrading the main program. Have this template type be controllable from the GNUmakefile. Demonstrate this feature with datatypes of **[float, double, complex<float>, complex<double>]**

3. Non-zero Dirichlet boundary conditions

FEPoissonOperator and **FEMain** will be upgraded to allow for a boundary condition class to be specified which has a virtual function that specifies a function which takes a Node as input and returns a templated type value

4. Demonstrate Piecewise Linear elements converge at 2nd order accuracy

Pick a known analytic function as your exact solution Φ. From this derive an exact representation for $f = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2}$. Use this f in your program, and set the boundary conditions at the boundary nodes to be Φ. Compute a sequence of solutions with finer grids where we can relate mesh-spacing h to the element area as $h = 2\sqrt{A}$. Compute a norm of the error $\Phi_n - \phi_h$ for each grid resolution and solve for the leading term in the error expansion.

5. Implement a time-dependent FEM solver

Given that you have a spatial discretization which produces a **SparseMatrix L** and a Right-hand side vector b for steady problem, you can formulate the semi-discrete problem (discrete in space, analytic in space. Generally referred to as the Method of Lines).

$$\frac{\partial \phi_h}{\partial t} = L_h(\phi_h) - f_h$$

This is a system of ordinary differential equations. We can evolve this forward in time if given $\phi_h(t=0)$ and boundary conditions over time $\phi_h(x=d\Omega,t)$ and a suitable integrator. A common method is Backward Euler.

So we can see that we have a new sparse matrix and a new right-hand side that can be solved for $\phi_h(t+\Delta t)$. This process is repeated for each time step to create a time sequence of the solution.

6. Verify correct time-dependent behavior

Decide on an analytic exact solution $\Phi(x,t)$. Derive analytically $f(x,t) = L\left(\Phi(x,t) - \frac{\partial\Phi(x,t)}{\partial t}\right)$. Use these in your time-dependent code for your initial values and boundary conditions and fh and calculate the error at the final time $E_h = \phi_h - \Phi_h$.

7. Create time-dependent animations of interesting source terms and boundary conditions

Given a sequence of **.vtk** files with names that have sequential integer names you can create a VisIt database on which you can show different animations.

8. Extrude the 2D elements into an extrusion into 3D

For a different set of command line arguments the input elements, or poly representation and creates a 3D extrusion of this mesh to a specified size which preserves the property h. ie. the 3D tetrahedron volumes have a cube root equivalent to the square root of the original 2D elements.

9. Solve Poisson Equation in 3D

Everything we have learned about 2D applies to a 3D problem. The 3D versions of our classes look mostly like the 2D code, but with an extra dimension.

10. Verify 3D steady solutions

Pick a known analytic function as your exact solution Φ. From this derive an exact representation for $f(x,y,z) = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2}$. Use this f in your program, and set the boundary conditions at the boundary nodes to be Φ. Compute a sequence of solutions with finer grids where we can relate mesh-spacing h to the element area as $h = 2\sqrt{A}$. Compute a norm of the error $\Phi_n - \phi_h$ for each grid resolution and solve for the leading term in the error expansion.