Finite Volume code for solving convection/diffusion equations

Eivind Fonn

August 30, 2007

Abstract

This document details code written for the LehrFEM library during the summer of 2007 by Eivind Fonn, for solving convection/diffusion equations using the Finite Volumes approach.

1 Background

The equation in question is

$$-\nabla \cdot (k\nabla u) + \nabla \cdot (cu) + ru = f$$

on some domain $\Omega \subset \mathbb{R}^2$. Here, $k, r, f : \Omega \to \mathbb{R}$, and $c : \Omega \to \mathbb{R}^2$. k is the diffusivity and should be positive everywhere. c is the velocity field.

In addition to the above, one can specify Dirichlet and Neumann boundary conditions on various parts of $\partial\Omega$.

2 Mesh Generation and plotting

The FV mesh generation builds upon FE meshes. Given a FE mesh, use the `add_MidPoints` function to add the data required for the dual mesh:

```
>> mesh = add_MidPoints(mesh, method);
```

`method` is a string specifying which dual mesh method to use. The two options are `barycentric` and `orthogonal`. If `method` is not specified, the `barycentric` method will be used. For the `orthogonal` method, the mesh cannot include any obtuse triangles, and if any such triangle exists, an error will occur.

The full mesh can be plotted using the following call:

```
>> plot_Mesh_LFV(mesh);
```

3 Assembly

Assembly of the stiffness matrices and load vector is done much the same way as in FE, i.e. assemble the stiffness matrices term-by-term, sum them up to get the full matrix. Assemble the load vector in the normal way, then incorporate Neumann boundary data, followed by Dirichlet boundary data.

For the lazy, there is a wrapper function for all of the above:

```
>> [A,U,L,fd] = assemMat_CD_LFV(mesh, k, c, r, d, n, f, cd, out);
```

This function assembles all the matrices and vectors required at once, without any hassle. Here follows a list of the input arguments:

- `mesh` The mesh struct (see last section).
- `k` Function handle for the k-function.
- `c` Function handle for the c-function.
- `r` Function handle for the r-function.
- `d` Function handle for the Dirichlet boundary data.
- `n` Function handle for the Neumann boundary data.
- `f` Function handle for the f-function.
- `cd` If equal to 1, specifies that the problem is convection-dominated, and that appropriate upwinding techniques should be applied. If 0, specifies that the problem is diffusion-dominated. If left out, the program itself will determine whether the problem is convection- or diffusion-dominated.
- `out` If equal to 1, the program will write output to the console. This is useful for large meshes, to keep track of the program.

If any of the function handles are left out or set equal to the empty matrix, it is assumed they are zero and will play no part.

The output are as follows:

- `A` The stiffness matrix.
- `U` The solution vector, containing Dirichlet boundary data.
- `L` The load vector.
- `fd` A vector with the indices of the free nodes.

From the above, the solution can be obtained by:

```
>> U(fd) = A(fd,fd)\L(fd);
```

The solution vector `U` contains the approximate values of the solution u at the nodes given in the mesh.

4 Error analysis

Given a mesh `mesh`, a FV solution vector `U` and a function handle `u` to the exact solution, the error can be measured in three different norms ($\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$) using the following calls:

```
>> L1err = L1Err_LFV(mesh, U, qr, u);
>> L2err = L2Err_LFV(mesh, U, qr, u);
>> Linferr = LInfErr_LFV(mesh, U, u);
```

Here, `qr` is any quadrature rule for the reference element, i.e. `qr = P7O6()`.

5 File-by-file description

Here follows a list of the files which were written as part of this project, and a description of each.

- `Assembly/assemDir_LFV.m`
  `[U,fd] = assemDir_LFV(mesh,bdFlags,fHandle)` incorporates the Dirichlet boundary conditions given by the function `fHandle` in the Finite Volume solution vector `U`. `fd` is a vector giving the indices of the vertices with no Dirichlet boundary data (i.e. the free vertices).
- `Assembly/assemLoad_LFV.m`
  `L = assemLoad_LFV(mesh,fHandle)` assembles the load vector `L` for the data given by the function `fHandle`.
- `Assembly/assemMat_CD_LFV.m`
  Wrapper function for assembly of convection/diffusion problems. See above for description.
- `Assembly/assemMat_LFV.m`
  `A = assemMat_LFV(mesh,fHandle,varargin)` assembles the global stiffness matrix `A` from the local element contributions given by the function `fHandle` and returns it in a sparse format. `fHandle` is passed the extra input arguments given by `varargin`.
- `Assembly/assemNeu_LFV.m`
  `L = assemNeu_LFV(mesh,bdFlags,L,qr,fHandle)` incorporates Neumann boundary conditions in the load vector `L` as given by the function `fHandle`. `qr` is any 1D quadrature rule. Note: In the general diffusion problem $-\nabla \cdot (k\nabla u) = f$, with Neumann boundary data $\frac{\partial u}{\partial n} = g$, the function h you need to pass to the Neumann assembly function is $h(x) = g(x)k(x)$.
- `Element/STIMA_GenGrad_LFV.m`
  `A = STIMA_GenGrad_LFV(v,mp,cp,m,bd,cH,kH,rH,cd)` calculates the local element contribution to the stiffness matrix from the term $\nabla \cdot (cu)$. Here, `v` is a 3-by-2 matrix giving the vertices of the element, `mp` and `cp` give the midpoints on each of the edges as well as the centerpoint (that is, the dual mesh geometry). `m` is the method used to generate the dual mesh (not used at the current time), `bd` are boundary flags for the edges (not used at the current time). `cH` is the c-function, `kH` is the k-function from the diffusion term and `rH` is a proper upwinding function. `cd` is 1 if the problem is convection-dominated and 0 otherwise. `kH` and `rH` are only used if `cd=1`. If not, they need not be specified.
- `Element/STIMA_GenLapl_LFV.m`
  `A = STIMA_GenLapl_LFV(v,mp,cp,m,bd,kH)` calculates the local element contribution to the stiffness matrix from the term $-\nabla \cdot (k\nabla u)$. The arguments are the same as above.
- `Element/STIMA_ZerOrd_LFV.m`
  `A = STIMA_ZerOrd_LFV(v,mp,cp,m,bd,rH)` calculates the local element contribution to the stiffness matrix from the term ru. The arguments are the same as above, except that `rH` is the function handle for the r-function.
- `Errors/L1Err_LFV.m`
  `e = L1Err_LFV(mesh,U,qr,fHandle)` calculates the discretization error in the norm $\|\cdot\|_1$, between the finite volume approximation given

by `U` (the solution vector), and the exact solution given by the function `fHandle`. `qr` is any appropriate quadrature rule on the reference element.

- `Errors/L2Err_LFV.m`
  `e = L2Err_LFV(mesh,U,qr,fHandle)` calculates the discretization error in the norm $\|\cdot\|_2$. All the arguments are as above.
- `Errors/LInfErr_LFV.m`
  `e = LInfErr_LFV(mesh,U,fHandle)` calculates the discretization error in the norm $\|\cdot\|_\infty$. All the arguments are as above.
- `MeshGen/add_MidPoints.m`
  `mesh = add_MidPoints(mesh,method)` incorporates the dual mesh data required for the finite volume method into the mesh `mesh`. `method` may be either `'barycentric'` or `'orthogonal'`. See earlier description.
- `Plots/plot_Mesh_LFV.m`
  `plot_Mesh_LFV(mesh)` plots the base mesh and the dual mesh.