16.930 Advanced Topics in Numerical Methods for PDEs

Massachusetts Institute of Technology – Spring 2015

Version 1.0

Project 34.A – Explicit Solution of First Order Hyperbolic Systems

Handed Out: March 30, 2015 Due (on-line): either April 17 at 12pm or May 11 at 12pm

The objective of this project is to build your first applications. We will start with the simplest case consisting of first order equations and explicit time integration. We will consider three particular applications: the scalar linear convection equation, the wave equation (written as a system of three first order coupled equations) and the Euler equations. All these equations can be written in the form:

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F}^{\text{inv}}(u, \boldsymbol{x}, t) = \quad \boldsymbol{S}(u, \boldsymbol{x}, t),$$

where u is the vector of unknowns, $\boldsymbol{F}^{\text{inv}}(u, \boldsymbol{x}, t)$ are the inviscid fluxes and $\boldsymbol{S}(u, \boldsymbol{x}, t)$ is the source term.

In this assignment, we introduce yet another data structure: the application data structure `app`. A detailed description of the `app` structure can be found in the file 2DG-Structures.pdf. You will see that unlike the other structures considered so far, the `app` structure depends on the application considered.

2DG Directory structure

When you unpack the file 2DG3.zip, it will create a directory called **2DG.3** which contains the subdirectories **master**, **mesh**, **util**, **dgker** and **app**.

The subdirectory **master** should be unchanged.

The subdirectory **mesh** should be the same that you already have, but I have included a new function, `mkmesh_distort.m`, which distorts the meshes created with `mkmesh_square.m` and hence can be used to test curved elements in an easy setting.

The subdirectory **util** contains an additional function `initu.m` which is used to initialize the vector of unkowns. The unknowns can be initialized to constant values, or set to be a function of $\boldsymbol{x}$, in which case the cell array `param` contains the pointer to a function.

The subdirectory **dgker** contains two kernel functions. A simple four stage Runge-Kutta time integrator, `rk4.m`, and the p-coded residual evaluation function `rinvexpl.m`. This version of `rinvexpl.m` will only work for scalar equations. One of the main tasks in this project is to write your own version of the residual evaluation function `rinvexpl.m`. Remember to add the path to this new directory in the function `setup.m`.

Finally, the subdirectory **app** contains some new files. Note that for each application there exists an additional subdirectory which is used to keep the flux functions and the function `mkapp.m` that creates the `app` structure which is specific to that application. So, in our case, we have the subdirectories **convection**, **wave** and **Euler**. These directories should not be included in the path using the `setup.m` function since only one of of these directories will be in the path at a given time. In the **app** subdirectory, you will also find some driver files `convection.m`, `wave_scattering.m` and `euler_channel.m`.

Your Tasks

- The first task consists of performing a convergence study for the linear convection equation. We consider a unit square, with a velocity field that corresponds to a rigid body rotation about the point $(0.5, 0.5)$ with angular velocity of unit magnitude in the counterclockwise direction. The initial condition is a Gaussian hill given by $u = \exp[-120((x - 0.60)^2 + (y - 0.5)^2))]$. At time $T = 2\pi$, the initial distribution will have returned to the original position and hence the difference between the solution at time T and the initial solution will be a measure of the error. We will measure this error using the L_2 norm and numerically perform a convergence analysis test to verify the expected rates of convergence. Since we are using explicit time stepping which is subject to severe stability restrictions, the time integration errors will be small. In order to perform a convergence test for a given p, first we will select the finest mesh we want to run and then we will use the same Δt when solving on coarse meshes. The version of the function `rinvexpl.m` provided, should work for this task.

- The second task consists of writing your own version of the function `rinvexpl.m`. Your function should work for linear and nonlinear fluxes and well as straight edged and curved triangles. Look into the document 2DG-Structures.pdf to see what each of the call back functions is supposed to do. All the callback functions have been provided for your convenience. The callback functions can be called at each Gauss point, but this is quite slow. A much better approach which is not hard to code is to call the callback functions once per element (or edge) and include the values of the unknowns at all the Gauss points.

- You will exercise your residual evaluation routine by solving the wave equation

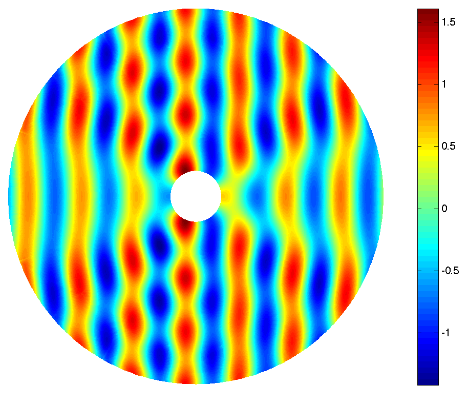
  $$u = \begin{pmatrix} q_x \\ q_y \\ \eta \end{pmatrix}, \quad \boldsymbol{F}^{\text{inv}} = -c \begin{pmatrix} \eta & 0 \\ 0 & \eta \\ q_x & q_y \end{pmatrix}$$

  A particular solution of the wave equation in free space is given by a planar wave of the form

  $$u(\boldsymbol{x}, t) = u_0 \begin{pmatrix} -k_x/k \\ -k_y/k \\ 1 \end{pmatrix} \sin(\boldsymbol{k} \cdot \boldsymbol{x} - ckt)$$

  where the wave number $\boldsymbol{k}$ is given by $\boldsymbol{k} = (k_x, k_y)$ and $k = |\boldsymbol{k}|$.

  We want to calculate the wave scattered by a unit cylinder when it is illuminated by a planar wave of the form shown above. A suitable mesh to study this problem can be obtained using the `mkmesh_trefftz.m` function as shown in the driver file `wave_scattering.m`. The scattered field can be obtained subtracting from the total field from the incident field. For $\boldsymbol{k} = (3, 0)$, and after the initial transient has passed, these fields will look like this:



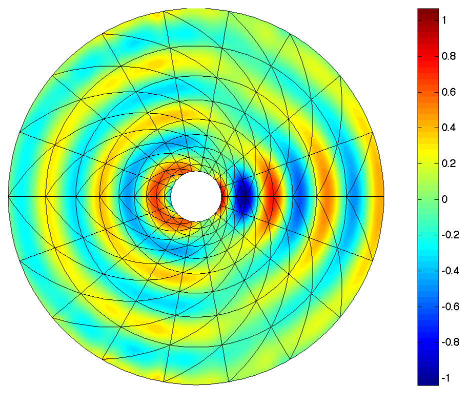


Figure 1: Total Field Scattered Field

A common way to report the strength of the scattered field is to plot the magnitude of the scattered filed, $|\eta^s|$ as a function of the azimuthal angle θ at a constant radius. For convenience, we will choose a circle which is at a radius $R = e$ and consider the azimuthal angle θ to be zero for $(e, 0)$ and run anticlockwise. Note that if the number of subdivisions, `m` in `mkmesh_trefftz` is taken to be an odd number (e.g. 11), then there will always be a grid line at $R = e$.

- Finally, you will test your residual evaluation function for the Euler equations. A good test is to consider the propagation of small perturbations. The driver file `euler_channel_entropy.m` has been set up to solve a subsonic $M = 0.3$ a uniform flow with a small amplitude Gaussian hill perturbation for the entropy placed in the middle of the domain (Note that if we perturb the entropy and keep the pressure constant, this is equivalent to modifying the density while keeping the pressure constant). This perturbation propagates with the flow velocity u_x to the right.

  You are asked to test your code with two additional waves (perturbations) that will propagate with speeds u_x+c and u_x-c, respectively, where c is the speed of sound. For these sound wave test cases, determine a smooth, compactly supported initial condition which would produce planar waves (i.e. no variation in y) that propagate at $u_x + c$, and, similarly, for the $u_x - c$ propagation speed.

Project deliverables

- You should submit your **2DG** directory on the stellar website before coming to class on the due date.
- Produce a table, or graph, showing the the convergence rates obtained for the linear convection problem for $p = 2, 3, 4$. You should select at least three values of h for each p to guarantee that you are in the asymptotic convergence range. Repeat the same process but now using the function `mkmesh_distort.m` to produce a mesh with curved elements.
- Plot the magnitude of the scattered wave by a unit cylinder for $\boldsymbol{k} = (3, 0)$ on the circle $R = e$ using $p = 1, 3$. First, determine a grid size H for $p = 3$ that is small enough so that you

obtain reasonably grid converged results. Then, for $p = 1$, run a mesh with size h such that this $p = 1$ solution will have the same number of degrees of freedom as the $p = 3$ solution. Discuss (briefly) the quality of these $p = 1$ and $p = 3$ solutions. (Note: use the full upwind flux function available in `wavei_roe.m` for these simulations).

- Determine the $u_x \pm c$ sound wave soutions for the Euler equations. For each wave set up a smooth, compact initial perturbation in the center of the domain and show the computed solution just before the perturbations exit the computational domain. Use $M = 0.3$ and $\gamma = 1.4$ for your simulations. For these results, again show and discuss a $p = 1$ and a $p = 3$ solution with the same number of degrees of freedom, using the same approach as for the wave scattering problem. (Note: use the full upwind flux function available in `euleri_roe.m` for these simulations).
- Solve the $M = 0.3$ and $\gamma = 1.4$ Euler flow in a channel for the mesh specified in `euler_channel.m` but setting the channel bottom obstacle to a height of `db=0.2`. Show the computed pressures and Mach numbers. As before, show and discuss a $p = 1$ and a $p = 3$ solution with the same number of degrees of freedom. (Note: use the full upwind flux function available in `euleri_roe.m` for these simulations).
- **Optional:** Using the exact same meshes for the same simulations as above, compare the performance of the Lax-Friedrichs flux to the full upwind flux for the wave scattering and compressible Euler flow problems, for both $p = 1$ and $p = 3$.