MA4245 Mathematical Principles of Galerkin Methods
Project 5: 2D Wave Equation

Prof. Frank Giraldo
Department of Applied Mathematics
Naval Postgraduate School
Monterey, CA 93943-5216

Due on Monday June 11

1 Continuous Problem

The governing partial differential equation (PDE) is

$$\frac{\partial q}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} q = 0 \qquad \forall (x,y) \in [-1,1]^2$$

where $q = q(x,y,t)$ and $\boldsymbol{u} = \boldsymbol{u}(x,y)$ with $\boldsymbol{u} = (u,v)^T$. Let the velocity field be

$$u(x,y) = y \qquad \text{and} \qquad v(x,y) = -x$$

which forms a velocity field that rotates fluid particles in a clockwise direction. Note that this velocity field is divergence free, that is, that the following condition is satisfied

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0.$$

The reason why this condition is important is that the identity

$$\boldsymbol{\nabla} \cdot (q\boldsymbol{u}) = \boldsymbol{u} \cdot \boldsymbol{\nabla} q + q \boldsymbol{\nabla} \cdot \boldsymbol{u}$$

simplifies to

$$\boldsymbol{\nabla} \cdot (q\boldsymbol{u}) = \boldsymbol{u} \cdot \boldsymbol{\nabla} q$$

which means that we can rewrite the initial problem statement in the conservation form

$$\frac{\partial q}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f} = 0 \qquad \forall (x,y) \in [-1,1]^2$$

where $\boldsymbol{f} = q\boldsymbol{u}$ is the flux. This form is now ready for use with the DG method.

Clearly, this problem represents a 2D wave equation that is hyperbolic and is thereby an initial value problem that requires an initial condition. Let that initial condition be the Gaussian

$$q(x, y, 0) = e^{-\left[(x-x_c)^2+(y-y_c)^2\right]/(2\sigma_c^2)}$$

where $(x_c, y_c) = (-0.5, 0)$ is the initial center of the Gaussian and $\sigma_c = \frac{1}{8}$ controls the shape (steepness) of the Gaussian wave. Use periodic boundary conditions for all four sides (i.e., you are solving flow on the surface of a torus; the iperiodic array will take care of this for you if you are using CG, for DG you have to do it via the fluxes).

2 Simulations

Use both CG AND DG methods to solve this equation. As usual, I recommend you try CG first and then move on to DG. For DG you will need additional maps for the edges. You need to write the code to handle arbitrarily-sized elements, polynomial orders, and integration formulas. What I mean by arbitrarily-sized elements is that you should not assume that each element is of the same size. You can, however, assume that each element uses the same polynomial and integration orders. As a test to make sure that you are doing this properly rotate your grid by 45 degrees to see if the code still works. Show me a plot at the initial and final time with the grid rotated.

2.1 Results You Need to Show

You must show results for $N = 1, 2, 4, 8, 16$ with increasing number of elements $N_e = nel^2$ where nel denotes the number of quadrilaterals in the x and y directions. Plot broken L^2 error norms (defined below) versus number of points (N_P) and show all 5 curves on one plot. Remember that you must use a log plot for the error to capture the spectral convergence.

For the following simulations, you must turn in four plots (one set of four for CG and another set of four for DG): two plots showing convergence rates (for exact and inexact integration) and another two plots showing 2-norm errors versus wallclock time. Write a discussion on your findings. Depending on how you write your code, you may not see much difference in the wallclock time between exact and inexact.

Also, give me a complexity analysis of your code (operation count) to see exactly how many operations your code is performing.

N=1 Simulations For $N = 1$ use $nel = 8, 16, 24, 32, 40, 48$ elements.

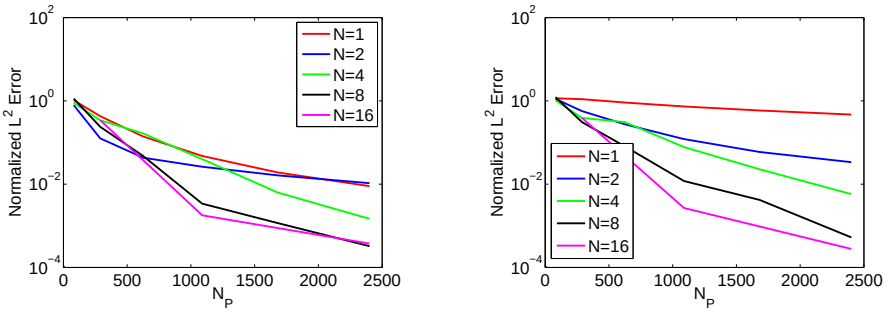
N=2 Simulations For $N = 2$ use $nel = 4, 8, 12, 16, 20, 24$ elements.

N=4 Simulations For $N = 4$ use $nel = 2, 4, 6, 8, 10, 12$ elements.

N=8 Simulations For $N = 8$ use $nel = 1, 2, 3, 4, 5, 6$ elements.

N=16 Simulations For $N = 16$ use $nel = 1, 2, 3$ elements.

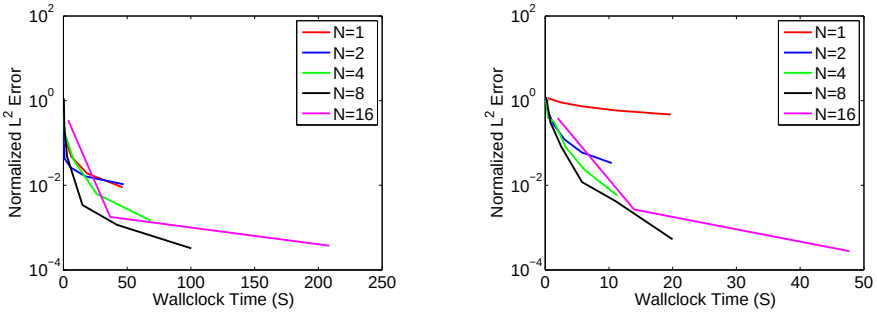
Here is an example of the Convergence rates plot you should show me:



a) Exact Integration b) Inexact Integration

Figure 1: The Convergence Rates for CG using a) exact and b) inexact integration using $C = 0.25$ with RK3.

Here is an example of the Computational Cost plot you should show me:



a) Exact Integration b) Inexact Integration

Figure 2: The Computational Cost for CG using a) exact and b) inexact integration using $C = 0.25$ with RK3.

3 Extra Credit

For extra credit, streamline your inexact integration codes to make them as fast as possible. Do not make overly simple assumptions (generality is still good) but make any assumption that is allowed by virtue of cardinality to make your codes as fast as possible. Give me timings of your code. If you wish to enter into

the "Fastest Code" Prize you must also send me a zip file of your code ready for me to run it on my computer. If I have to debug your code to run it, it is an immediate disqualification from the contest.

4 Helpful Relations

Error Norm The normalized broken L2 error norm that you should use is:

$$||error||_{L^2} = \sqrt{\frac{\sum_{e=1}^{N_e}\sum_{i=1}^{M_N}\left(q_{(e,i)}^{numerical} - q_{(e,i)}^{exact}\right)^2}{\sum_{e=1}^{N_e}\sum_{i=1}^{M_N}\left(q_{(e,i)}^{exact}\right)^2}} \tag{1}$$

where $e = 1, ..., N_e$ are $N_e = nel^2$ quadrilateral elements and $q^{numerical}$ and q^{exact} are the numerical and exact solutions at the M_N grid points inside each element. This L^2 error is valid for both CG and DG.

Time-Integrators To solve the time-dependent portion of the problem use the 3rd order SSP (Strong-Stability-Preserving) RK method: for $\frac{\partial q}{\partial t} = R(q)$ let $q^{(0)} = q^n$ and

$$q^{(1)} = q^{(0)} + \Delta t R(q^{(0)}),$$
$$q^{(2)} = \frac{3}{4}q^{(0)} + \frac{1}{4}q^{(1)} + \frac{\Delta t}{4}R(q^{(1)})$$
$$q^{(3)} = \frac{1}{3}q^{(0)} + \frac{2}{3}q^{(2)} + \frac{2\Delta t}{3}R(q^{(2)})$$

where $q^{n+1} = q^{(3)}$, or a better time-integrator of your choice. Make sure that your time-step Δt is small enough to ensure stability. Recall that the Courant number

$$C = u\frac{\Delta t}{\Delta x}$$

must be within a certain value for stability. For the 3rd order RK method I gave you, I used $C = 0.25$ for all the simulations; however, for stability purposes, the time-step can be much bigger.