UC Berkeley Math 228B, Spring 2024: Problem Set 7

Prof. Per-Olof Persson (persson@berkeley.edu)

Due May 3

1. (a) The `dgconvect0` function on the course web page has several shortcomings. Write a new version named `dgconvect` which incorporates the following improvements:
    1. Replace the equidistant node positions in an element by the Chebyshev nodes $s_i = \cos(\pi i/p)$, $i = 0, \dots, p$, scaled and translated to $[0, h]$ and in increasing order.
    2. Implement support for arbitrary polynomial degrees p, by computing the mass matrix `Mel` and the stiffness matrix `Kel` using Gaussian quadrature of degree $2p$ (see function `gauss_quad` on the course web page). Form the nodal basis functions using Legendre polynomials (see function `legendre_poly` on the course web page).
    3. The original version plots the solution using straight lines between each nodal value. Improve this by evaluating the function (that is, the polynomials in each elements) at a grid with $3p$ equidistant nodes, and draw straight lines between those points.
    4. Replace the discrete max-norm in the computation of the error by the continuous L_2-norm $\|u\|_2 = \left(\int_0^1 u(x)^2\,dx\right)^{1/2}$.

   (b) Write a function with the syntax `errors, slopes = dgconvect_convergence()` which runs your function `dgconvect` using $p = 1, 2, 4, 8, 16$, $\Delta t = 2 \cdot 10^{-4}$, $T = 1$, and number of elements n chosen such that the total number of nodes $n \cdot p$ equals $16, 32, 64, 128, 256$. Return the corresponding errors in the 5-by-5 array `errors`, and estimate 5 slopes in the array `slopes` making sure to exclude points that appear to be affected by rounding errors. Also make a log-log plot of the errors vs. the number of nodes $n \cdot p$.

2. (a) Write a function with the syntax `u, error = dgconvdiff(; n=10, p=1, T=1.0, dt=1e-3, k=1e-3)` which is a modification of your `dgconvect` function from the previous problem to solve the convection-diffusion equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} - k\frac{\partial^2 u}{\partial x^2} = 0, \tag{1}$$

   on $x \in [0, 1]$ with the same initial condition as before, $u(x, 0) = \exp\left\{-100(x - 0.5)^2\right\}$, and periodic boundary conditions. Use the LDG method for the second-order derivative with $C_{11} = 0$ and $C_{12} = 1/2$ (pure upwinding/downwinding). For the error computation, use the exact solution

$$u(x, t) = \sum_{i=-N}^{N} \frac{1}{\sqrt{1 + 400kt}} \exp\left\{-100\frac{(x - 0.5 - t + i)^2}{1 + 400kt}\right\} \tag{2}$$

   where N should be infinity but $N = 2$ is sufficient here.

   (b) Write a function with the syntax

   ```
   errors, slopes = dgconvdiff_convergence()
   ```

   that performs a convergence study for your `dgconvdiff` function exactly as in problem 1, using a diffusion coefficient of $k = 10^{-3}$.

Code Submission: Your Julia file needs to define the four requested functions, with exactly the requested names and input/output arguments, as well as any other supporting functions and variables that are required for your functions to run correctly.