In this homework assignment you will compile and run the Fortran code **burgers-stdnt18.f** available at the course website

http//courses.engr.uky.edu

The data file which must be read by the code is **inputfle.d**, also available at the course website. You will introduce a few minor modifications to this file to provide the range of data needed for the runs of this assignment, one of the main tasks of which is to solve a Navier–Stokes-like equation, the 1-D Burgers' equation,

$$u_t + \frac{1}{2}(u^2)_x = -p_x + \frac{1}{Re}u_{xx}\,, \qquad x \in (0,1)\,, \tag{1a}$$

with initial data

$$u(x,0) = u_0(x) = x, \tag{1b}$$

and boundary conditions

$$u(0,t) = 0 \quad \text{and} \quad u(1,t) = 1\,. \tag{1c}$$

In (1a) $-p_x$ is a forcing function that must, in general, be coded in three subroutines. You will consider a problem having a nearly trivial exact solution $u(x) = x^2$ and thoroughly study grid-function convergence rates in the context of the cell-Re problem. In particular, you will run the problem for several different values of Re with a range of grid spacings using default centered differencing of advective terms, first-order upwinding of these terms, and mollification (filtering) applied to the centered-difference approximate solution as it is advanced in pseudo time during integration to a final steady state.

1. Substitute the given exact solution into Eq. (1a) to find the function $-p_x$. Note that this is how exact solutions to differential equations are often produced: choose a solution with desired physical and/or mathematical features, substitute it into the DE, and find the forcing function needed to yield the given solution. Engineers call this the "method of manufactured solutions," and its use by engineers is relatively recent; but it has been employed by numerical analysts since the beginnings of machine computation.

2. In this problem you will demonstrate how the code performs when solutions are classical ($u(x) = x^2$)—in fact, in this case (trivially) C^∞. Use $u_0(x) = x$ for initial data, and compile the code; in the input file set $Re = 10^1$. Starting with grid spacing $h = 0.1$, that is, 11 spatial points, and a time-step size $k = 0.1$, integrate for 100 time steps. Then decrease both grid spacing and time-step size together—which means keep the Courant number constant—successively by factors of two making sure to double the number of time steps for each new run so you always integrate to the same final time. Continue this until you find a spatial step size sufficiently small that centered differencing provides a good solution—say, uniform four decimal place accuracy. Then refine the grid (and time-step size) one additional time to permit testing grid-function convergence rates, both in the L^2 and maximum norms—output printed on the screen by the code.

   Next, go back to the 11-point grid, set *ifltr* = 1 and *fprm* between about 100 and 1000 in the input file. Via numerical experiments, determine the best value (say, to two or three significant digits) of *fprm* with respect to the observed error norms and comparisons of plots of the exact and computed solutions. Then refine the grid (possibly several times) to permit grid-function convergence tests, and do these, as done above, using the same sequence of spatial and time step sizes.

Finally, repeat this process using 1^{st}-order upwinding with *ifltr*= 0 and $iupwnd = 1$ entered in the input deck. Again, perform grid refinement to permit conducting grid-function convergence tests as done in the preceding two parts. Make a simple table summarizing your results for order of accuracy for the three methods, and for each method provide a single plot showing $u(x)$ *vs.* x for the exact solution and for solutions on the three finest grids employed for your convergence tests. Discuss these results. Are they what you would expect? Why, or why not?

3. Estimate the number of grid points needed to satisfy the center-difference cell-Re restriction with $Re = 10^3$. Then perform calculations on 11-, 21- and 41-point grids with a fixed time step size $tk = 0.005$ on all three grids, using each of the three methods: centered differencing, centered differencing with mollification, and 1^{st}-order upwinding, as done in Prob. 2. Do any of these grids satisfy the restriction? In cases where mollification is employed, again attempt to find the optimal parameter *fprm* to within two or three significant digits. Note whether these values change with grid spacing—and discuss (why they should, or should not?)—and whether solution accuracy is sensitive to values of this parameter. Be sure to use a sufficient number of time steps to produce a steady solution; you should look at (plot!) the contents of the file **time-series.out** to assess whether steady state has been reached.

4. Repeat the above for $Re = 10^5$ and $Re = 10^7$. Determine the grid spacing needed to essentially remove cell-Re difficulties for centered differencing, being sure that solutions have reached steady state. Then determine the coarsest grids on which first-order upwinding and filtered centered differencing produce reasonably accurate solutions. Compare the run times of these three approaches. (No grid-function convergence required for this problem.)

With the results obtained by solving the above problems, prepare a "lab report" (or a journal-paper-like write up) to document your findings. This should consist of the following parts. First, there should be a short (no more than one paragraph) **abstract** which briefly summarizes what you did, and what were your most important results. The body of the report should begin with an **introduction** in which you discuss the problem and its importance, and cite references (*e.g.*, some of those given in the lecture notes). This should be followed by an **analysis** section where you present a well-posed mathematical representation of your problem and describe how you will solve it. Provide a centered-difference discretization of Eq. (1a), quasilinearization of non-linear terms, and modifications needed for upwinding of the advective term; then show (and discuss—explain why it should work fairly well) the formula used for mollification. Next demonstrate the method for finding p_x carried out in Prob. 1. (You may wish to divide the analysis section into several subsections.) The next section contains **results**. The final section presents your **conclusions**—*i.e.*, what did you learn from this exercise? In this section provide a short (one- or two-sentence) summary of what you did; then state what you learned regarding the three methods employed in this exercise.