PH30110 Computational Astrophysics, Fluid Dynamics 2022-2023

# HAND-IN EXERCISE FLUID DYNAMICS COMPLETE

You will need to hand in two files for me to mark: One C source code file(s) (i.e. .c, which I can compile and run) and one text file (pdf) with brief answers and code output. Please include your source code as an appendix to the pdf file as well.

**Question 1**: *A functional fluid solver*

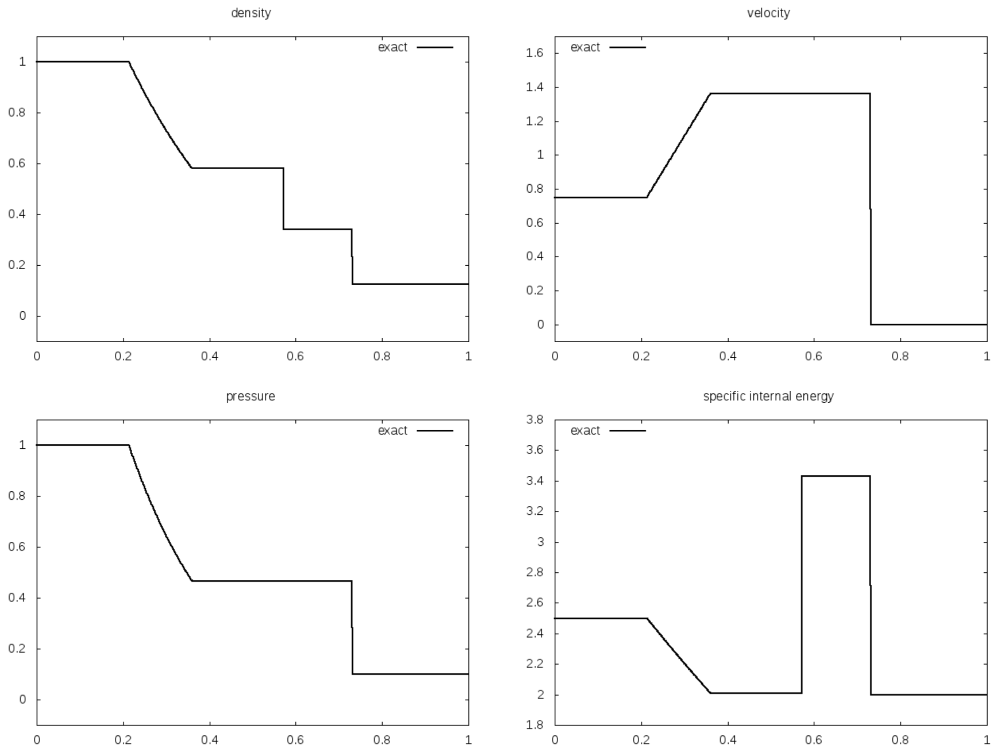

Figure 1: Exact shock tube test result for problem A. The initial conditions are that the left state ($x < 0.3$ at $t = 0$) has $\rho = 1$, $p = 1$ and $v = 0.75$, whereas the right state has $\rho = 0.125$, $p = 0.1$ and $v = 0$. The snapshots are taken at $t = 0.2$.

Your task is to write a C code that solves Euler's equations (without forces, heating or other source terms) and is capable of (approximately) reproducing Figs. 1 and 2. These are a standard shock tube tests, using a polytropic index $\hat{\gamma} = 1.4$. You

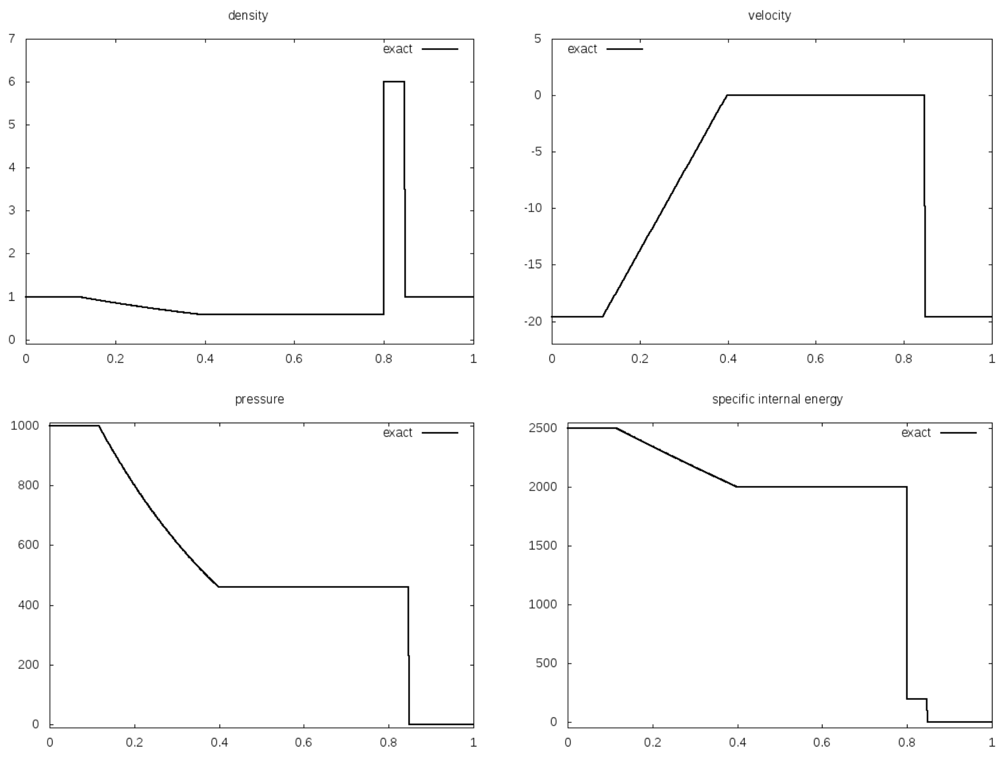

Figure 2: Exact shock tube test result for problem B. The initial conditions are that the left state ($x < 0.8$ at $t = 0$) has $\rho = 1$, $p = 1000$ and $v = -19.59745$, whereas the right state has $\rho = 1$, $p = 0.01$ and $v = -19.59745$. The snapshots are taken at $t = 0.012$.

can use the figures from the computational fluid dynamics notes for illustrations of how your method of choice performs against the exact solution. Fig. 2 is the more challenging of the two for a fluid dynamics solver, because the conditions are set to be more extreme. Make sure to use boundary conditions that will lead to the expected behaviour at the grid boundaries.

## **Question 2**: *Spherical explosion - more challenging problem*

For this exercise, we will stay with one dimension, but move from Cartesian grid to spherically symmetric coordinates. If you recast Euler's equations in spherical form, you will find that you should be able to recycle your methods from the Cartesian grid approach as long as you capture the extra terms in *geometrical source terms*, that is terms without a derivative that you might well treat as any other source terms. A simple but actually quite reasonable way you would treat a source term in your solver is by having it act during your state update along with the flux

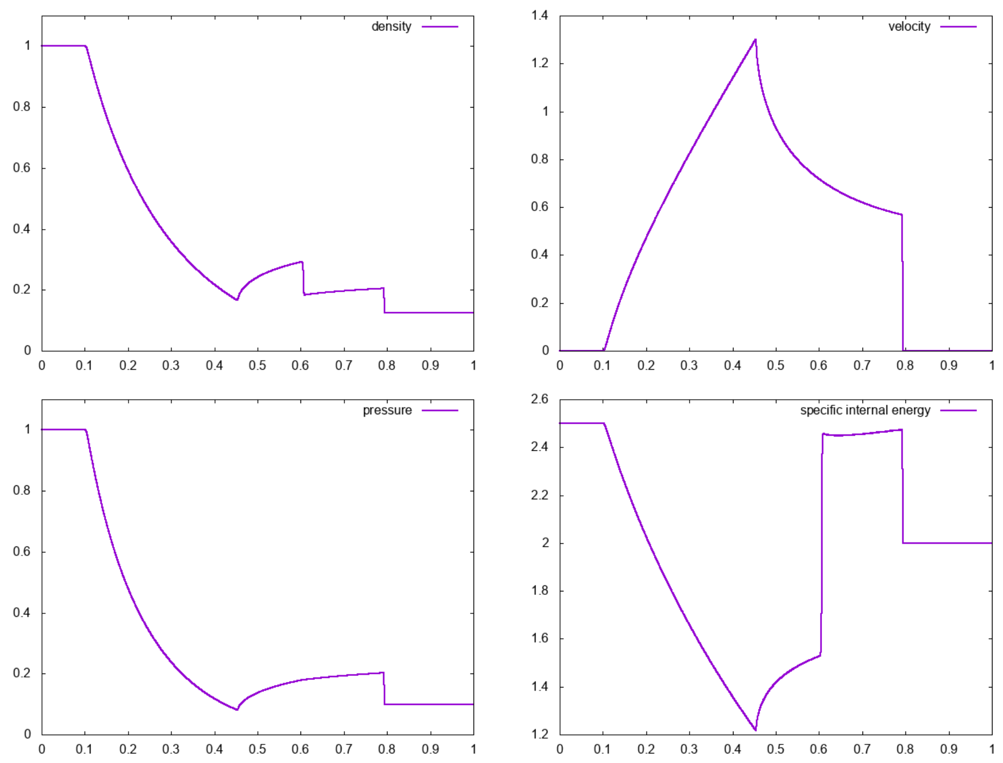

Figure 3: High-resolution PLM results for spherical explosion. The initial conditions are that the left state ($r < 0.4$ at $t = 0$) has $\rho = 1$, $p = 1$ and $v = 0$, whereas the right state has $\rho = 0.125$, $p = 0.1$ and $v = 0$. The snapshots are taken at $t = 0.25$.

update without worrying whether these two update operations should somehow have been mixed together: the changes to the state due to the flux update and due to the source term just get added separately.
Try to reproduce the general features of Fig. 3, which uses a polytropic index $\hat{\gamma} = 1.4$ and is set in spherical coordinates. Note that, while the set-up here is still necessarily an abstraction, this is indeed the expected pattern of a supernova blast wave and subsequent supernova remnant once the outflow has left the vicinity of the exploding star. Eventually, the forward shock / reverse shock pattern is a source of synchrotron emission and a background for cosmic ray models.

## Hand-in

HAND-IN: 1. Your version of Figs 1, 2 and 3 with your solution (using 100 zones) *overplotted* on the exact solution. The data for the exact solutions from the figures is available on Moodle, for the spherical set-up I have added a high resolution simulation result. Use any of the solvers described in the notes, with higher points awarded for more powerful solvers that get closer to the solution. If you went for a basic solver and wish to emphasize that it converges to the exact solution at $> 100$ zones, you are allowed to include a higher resolution version of your result as well. Again, you can use any program for plotting (python, gnuplot...) but *all* computation should be done by your C code. Comparison solution data is available on Moodle.

2. A brief text (not more than one page, more concise is allowed) motivating your approach to the problem, your choice of solver, time step and flux reconstruction. Explain the workings of your code as if teaching the topic to a fellow student (ie *don't* just read out the lines of your code, I can see that you declare a variable or a for loop)

3. Your source code, well annotated and with clear layout.

## Some remarks on marking

For this exercise, I will not necessarily award additional marks for advanced coding concepts (eg pointers to functions) if they do not serve an obvious purpose in your implementation. You can earn up to 15 points for your motivation paragraph. A basic solver (lowest-order Lax-Friedrichs) will allow you to earn up to 55 points (indicative: 15 for a well-written/annotated code; 25 for a conceptually correct code allowing for bugs that do not reveal a lack of understanding; 15 for a successful output). A further 20 points can be earned for implementing a higher-order time-step, a more complex flux reconstruction algorithm and parallelization of the code. If you go beyond the lowest-order Lax-Friedrichs, you will need to provide a demonstration that your algorithm is indeed superior. A final 10 points maximum are set aside for those of you who fully solved question 2 as well.