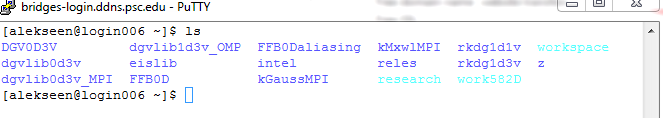
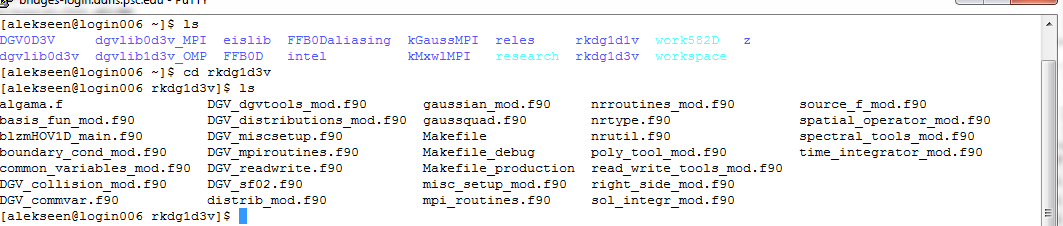
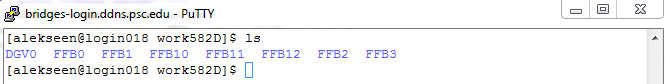
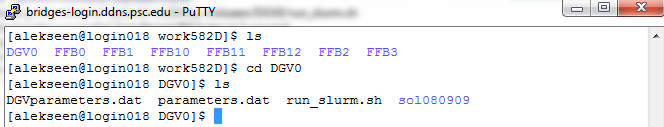
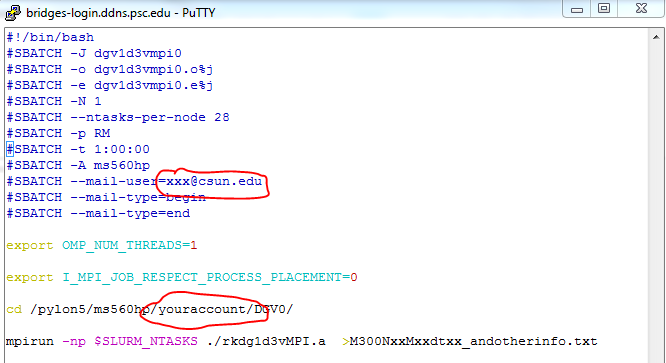
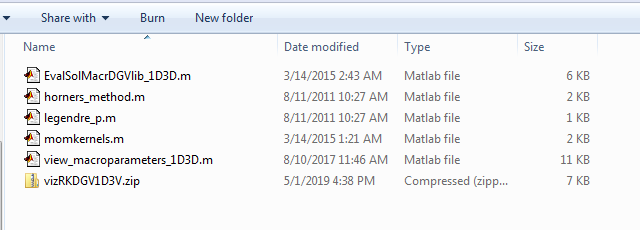
Steps. Week 13 Setting up Simulations of 1D3V Normal Shock Wave

*Steps to perform various tasks using RKDGV1D3V code.*

**1. Codes/files that we will use in this Project**

* Code **RKDGV1D3V.** This is a version of the DGV library that is merged with 1D spatial driver. The parameter files and the code will look very familiar. The file to download from Canvas is **RKDGV1D3V.zip**.
  + Download the archive file. On Bridges, create a directory **rkdg1d3v.** 
  + Copy the code to the **rkdg1d3v** directory.  
    
  + Do not compile just yet: you will need to modify **DGV\_sf02.f90**
* Working Directory Archive **RKDGV1D3V\_workingdirectory.zip**.
  + Download the archive file. On Bridges, go to the Pylon5 work file space. Create a folder **DGV0**.   
    
  + Copy contents of the archive into the folder.  
      
      
    you may notice that you do not have as many subfolder as you had in the zero dimensional simulations using **DGV0D3V** code. This is not a mistake. It just happened that the 1D driver will place all output in sol080909.
  + Modify the file **run\_slurm.sh** so that is uses the correct e-mail to send updates on the status of the code and   
    
* Visualization of the solution moments. **Matlab** code archive **vizRKDGV1D3V.zip**. Download the archive file. Copy all files from the archive to a directory where you will store the solutions to the 1D problem.  
    
  

**2. What is new in RKDG1D3V.**

**RKDG1D3V** is closely related to the code **DGV0D3V** we used earlier. Both codes use **DGVlib** library and both codes use the same parameters files. As a result handing the **RKDG1D3V** code will be in many ways similar to **DGV0D3V** code. Albeit, there will be differences. First important difference is that the code is 1D. As a results more parameters need to be set in **parameters.dat.** Second important difference is that the code is MPI parallel. As a result, additional parameters need to be specified in DGVparameters.dat and additional parameters need to be specified in **run\_slums.sh.** The third difference is visualization. We will not have an easy way to visualize the solution. Instead, we will plot moments of the solutions.

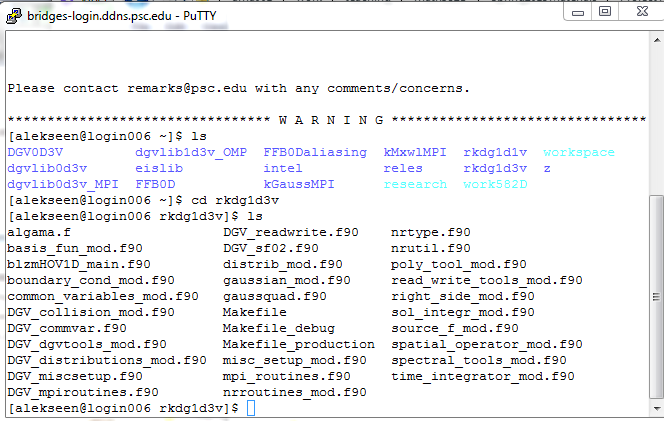
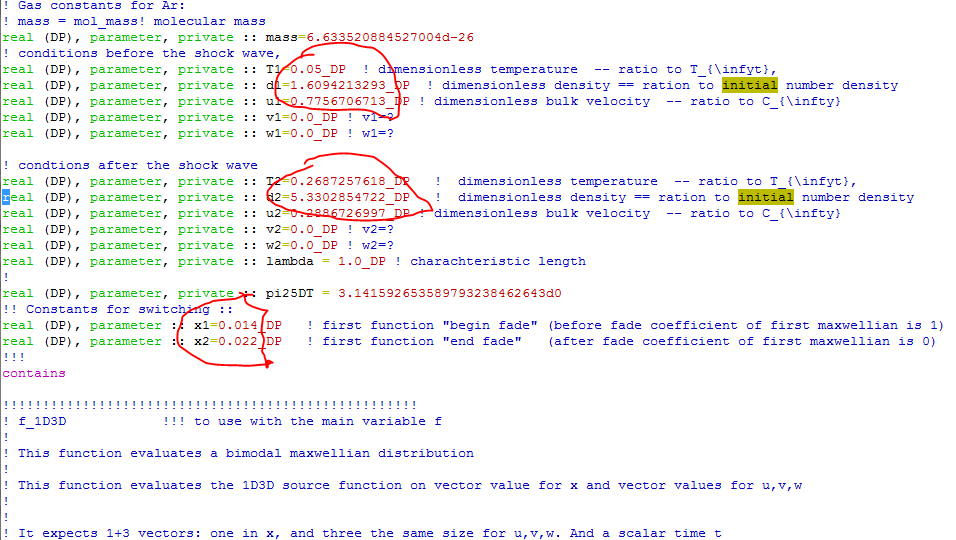
**2. Steps to figure out dimensionless values for the normal shock wave.**

The RKDGV uses dimensionless variables. This aspect of the code is identical to DGV0D3V. The steps we take are almost identical to steps we used when setting up spatially homogeneous relaxation runs for DGV0D3V. However, there will be one difference: we will need to figure the size of the spatial domain and the length of time for simulations. Let us get started:

* Use the spreadsheet file you prepared in Project 3 to compute dimensionless values for Mach 3 conditions. You can use the same values you had in simulations of spatially homogeneous relaxation as long as your reference temperature was above 2500 K.
* Read the introductory information on the shock wave simulation in file shock\_wave\_setup.pdf. Follow the directions given in the file to estimate a few numbers
  + Size of the domain in variable x, dimensionless (about 120 mean free times)
  + Dimensionless values x\_1 and x\_2 to designate the transition zone for initial data in formula 7. The distance between x\_1 and x\_2 is about 20 MFPs location of x\_1 and x\_2 roughly about the middle of the computational domain, perhaps shifted a bit to the right.
  + The dimensionless value of the final time (start from 200 MFTs).

**3. Steps to prepare the executable.**

We will attempt to simulate a normal shock wave using RKDG1D3V. The first step is to set up the executable. To prepare the executable, we only need to set up DGV\_sf02.f90 to have the appropriate values of the streams. Use the spreadsheet file you prepared in Project 3 to compute dimensionless values for Mach 3 conditions. You can use the same values you had in simulations of spatially homogeneous relaxation as long as your reference temperature was above 2500 K.

* Go to the folder containing RKDG1D3V code.   
  
* Open DGV-sf02.f90 for editing. Modify parameters of the streams and the values x\_1 and x\_2 that determine the transition zone of the initial data.  
    
  Notice that in 1D code densities are NOT multiplied by 2.
* Compile the code.

**4. Steps to prepare the working directory.**

Before we can run a simulation, we need to prepare the working directory. We have already started on this process creating a new directory DGV0 and placing the parameter files and the run\_slurm.sh file there.

Then we need to copy the correct executable to the folder.

The next steps explain the rest of the process.

* Update **DGVparameters.dat**  
  The following parameters need to be updated in the DGVparameters.dat
  + number of cells in u = 10,21,4 !! Mu\_list   
    degree of local Lagrange basis in u = 5,1,1 !! su\_list   
    number of cells in v = 10,21,4 !! Mv\_list  
    degree of local Lagrange basis in v = 5,1,1 !! sv\_list  
    number of cells in w = 10,21,4 !! Mw\_list  
    degree of local Lagrange basis in w = 5,1,1 !! sw\_list  
      
    Use your combination of M and s, s<=8. Pay attention that the total number of points per dimension Ms. If this number is too big, you will have issues with solution memory size.
  + current solution base name = M380\_ES\_  
      
    use some suitable name here.
  + number of procs for linear problem = 4 ! num\_lin\_proc --  
    The number that goes here is the total number of the MPI processes that you will request in run\_slurm. It should be equal to (number of nodes)\*(number of processors per node).
  + error maxwell ESBGK = 10.0  
    error maxwell Vel ESBGK = 200.9 ! vel\_lev -- velocity dependent regime  
    error maxwell linearization = 300.0 ! linear\_lev,   
    error maxwell decomposition = 400.0 ! decomp\_lev  
      
    In this set of parameters the threshold values are specified to switch between five different models of collision, numbered from 0 to 4. We are interested in using only the last one. Therefore we need to adjust the numbers so that the indicator stayed below the last threshold. The values provided above will guarantee it.
  + ref termal velocity = 1580.375122325165 ! C\_inf m/s  
    ref characteristic length = 1.0 ! L\_inf m  
    ref number of molecules = 1.0E+21 ! N\_inf  
      
    Use the parameters of the dimensionless reduction here.
* Update **parameters.dat**  
  The following parameters need to be updated
  + left endpoint in x = 0.0 ! x\_left   
    right endpoint in x = 0.032 ! x\_right  
      
    Set up the left and right boundary of the computational domain. The numbers should be consistent with the x\_1 and x\_2 parameters used in DGV\_sf02.f90. The size of the domain is given in the dimensionless variables.
  + number of cells in x = 16,11,16 ! N\_list  
      
    Select the number of uniform cells in variable x to run the simulation. You can start from 16 cells and then increase. Note that time step generally needs to be decreased proportionally to maintain solution stability.
  + current solution base name = M380\_ES\_ !  
      
    Provide a meaningful file name to be used with the simulation outputs.
  + final time = 0.1 ! final value of time (USC Mach 10: 0.00025  
    time step = 0.00005 ! value of the time step (depends on dx! and rk!)  
      
    Initial time should be left to be zero. Compute a value of the dimensional time to stop simulation (after about 200 MFTs) and a suitable time step. You can start by making it to be equal to 1/105 of the 200 MFT. Stability of the simulations depends on the ratio of temporal step and the spatial cell size, dt/dx. If this ratio is larger than certain critical value (generally unknown), simulations will crash. So, if you are increasing number of cells in x, you should also decrease the time step.
  + instances to save solution = 10 ! num\_save\_sol = how many times the solution (and error and other quantities) is saved during the evolution  
      
    use a reasonable number 10-20 to produce interim saves. There is a way to restart the solution from an interim save if the solution is timed out. Too many saves will use space on the disk.
* Update **run\_slurm.sh**The following lines need to be updated:
  + #SBATCH -N 1   
      
    Select the number of computer nodes that you will request for the run
  + #SBATCH --ntasks-per-node 28  
      
    do not change this one – we will always request all processor on a node.
  + #SBATCH -t 1:00:00  
      
    You will have to figure how much time to request for the run. Record times used in your simulations to estimate time for future simulations.
  + mpirun -np $SLURM\_NTASKS ./rkdg1d3vMPI.a >M300NxxMxxdtxx\_andotherinfo.txt  
      
    Make sure you are calling a correct executable and provide a meaningful name for the output file.