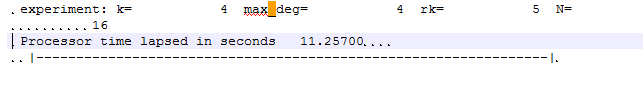
Steps. Week 14 Performing Simulations of 1D3V Normal Shock Wave, Collecting and Processing Results

*Steps to perform various tasks using RKDGV1D3V code.*

**1. Steps to measure effectiveness of the MPI parallelization.**

1. Set up the **RKDGV1D3V** code to perform simulation of normal shock wave with Mach number 3. In addition,
   1. In **DGVparameters.dat**, pick some reasonable values of number of velocity cells M and the number of nodes per cell s. M=10 and s=5 should be sufficient.  
        
      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
   2. In **DGVparameters.dat**, specify the total number of processors that will be used  
        
      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.sh**. It should be equal to (number of nodes)\*(number of processors per node).( See 1.g and 1.i)
   3. In **parameters.dat**, select some reasonable number of the spatial cells, N=16 (with the provided by default k=4) should be sufficient.  
        
      scheme order in x = 4 ! k\_list - degree of the polynomials  
      left endpoint in x = 0.0 ! x\_left   
      right endpoint in x = 2.0 ! x\_right  
      uniform mesh in x = yes ! mesh\_x\_uniform  
      number of cells in x = 16,11,16 ! N\_list
   4. In **parameters.dat** specify the time step small enough so that the code guaranteed would not go unstable. It can be a very small time step. For this part, going with a smaller time step is good.  
        
      time step = 0.00005 ! value of the time step (depends on dx! and rk!)
   5. In **parameters.dat** specify the final time to be equal 100 time steps. The code will reach this time in 100 steps and will stop  
        
      final time = 0.005 ! final value of time
   6. In **parameters.dat** specify the parameter  
        
      instances to save solution = 1 ! num\_save\_sol = how many times the solution (and error and other quantities) is saved during the evolution  
        
      ask for a single save so that saving of the file does not contaminate much the time measurements.
   7. In **run\_slurm.sh**, request the number of nodes to be used in computations. The nodes are requested in the line  
        
      #SBATCH -N 1  
        
      Request 1 node (is done above) for the first simulation. In the consecutive simulations ask for 2, 4, 8, 16, and 32 nodes, respectively.   
        
      Make sure the number of processors specified in **DGVparameters.da**t (see 1.b) is consistent with the number of processors requested in **run\_slurm.sh**
   8. Keep the line   
        
      #SBATCH --ntasks-per-node 28  
        
      unchanged. You are requesting that twenty 28 instances of the code be started on each node. Since the node has 28 processors, it will be one task per processor and the hardware of the node will make sure that each copy runs on a unique processor.
   9. Provide an appropriate name of the output file   
        
      mpirun -np $SLURM\_NTASKS ./rkdg1d3vMPI.a >test\_MPI\_time\_nodes1\_M10s5N16.txt
2. Send the program into execution.
3. When the program concludes with probably take a minute, copy the output file to a separate folder on a local computer.
4. Repeat steps 1.g, 1.i and 2), 3) for numbers of nodes 2, 4, 8, 16, and 32.
5. Collect the measured CPU wall times. For that open the output file using a suitable text editor and go to the end of the file. Look for the following line:   
   This line states time in second on the wall clock that it took to complete the simulation. Please note that this time also includes time spent on communications between processes and time to write the solution on the hard drive.
6. Use the collected time measurements to fill out the table.  
   For this table speedup = (“time measured for 1 node” )/ (“time measured for N nodes”)

|  |  |  |  |
| --- | --- | --- | --- |
| Number of nodes | Number of processors | CPU Wall Time | Speedup |
| 1 |  |  |  |
| 2 |  |  |  |
| 4 |  |  |  |
| 8 |  |  |  |
| 16 |  |  |  |
| 32 |  |  |  |

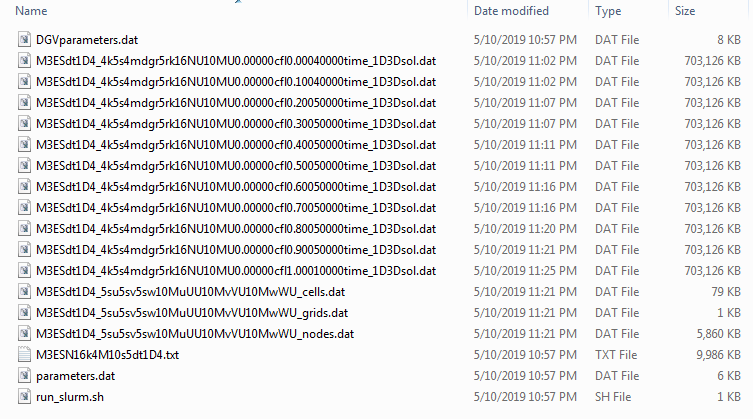
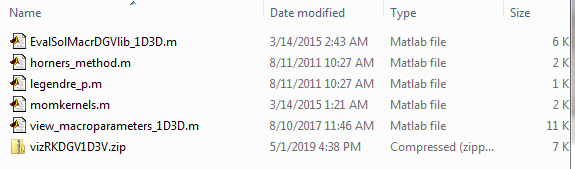
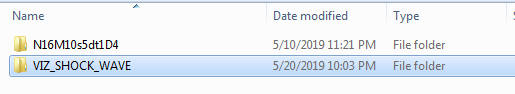
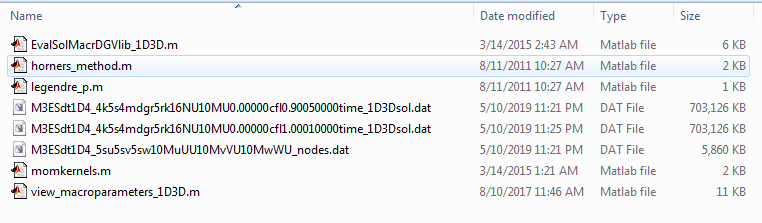
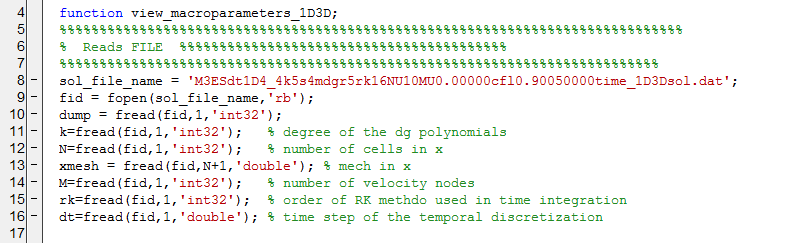
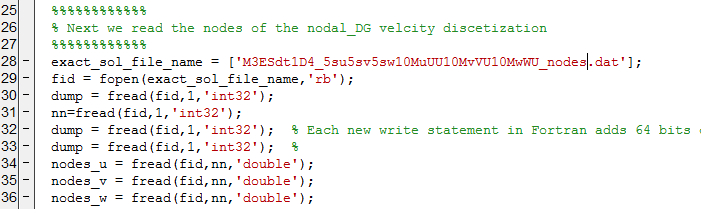
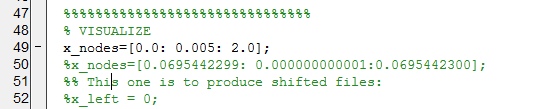
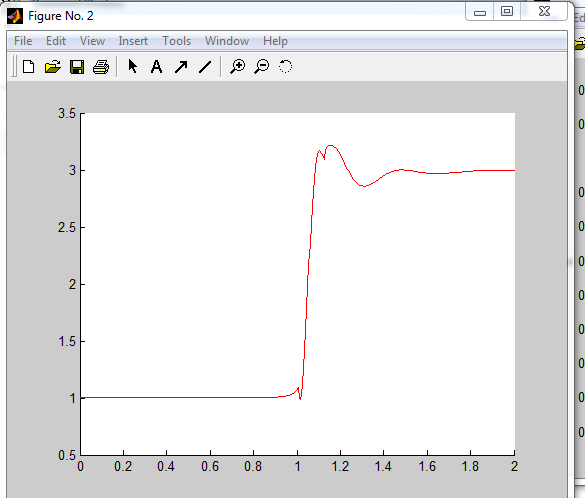
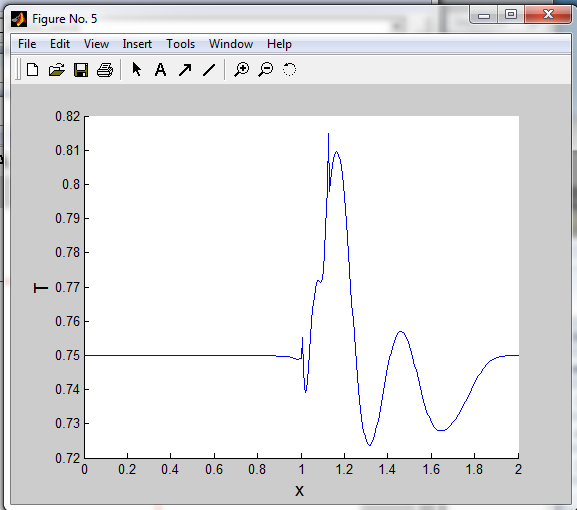
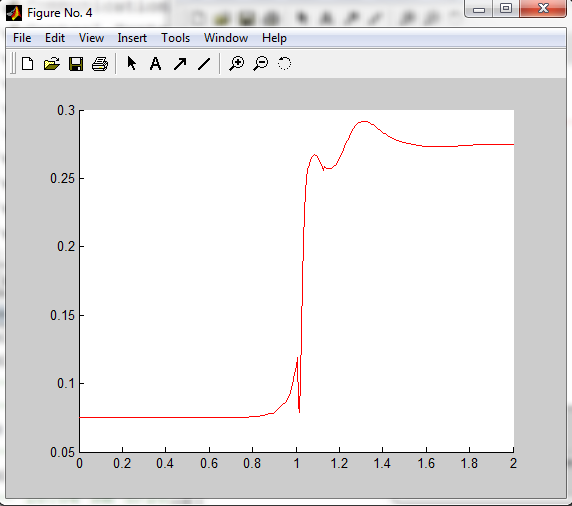
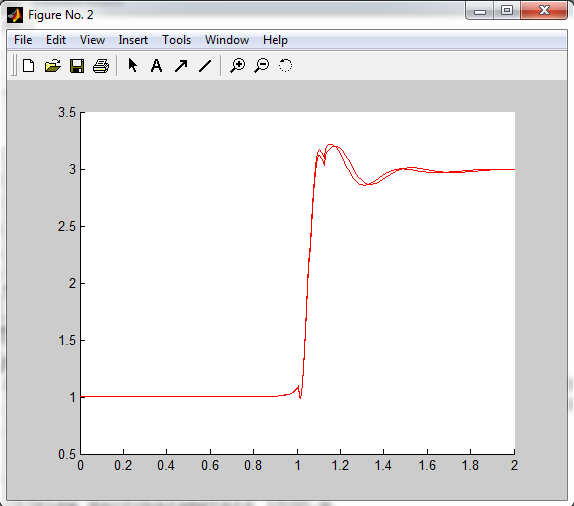
1. Prepare a graph of Speedup versus the number of nodes. Make a graph of the line with slope one passing through the origin. This line corresponds to the ideal speedup. (Can you explain why?) Propose explanations to why the observed speedup is less than ideal as the number of processors increases.

**2. Steps to compute a solution to the normal shock wave**

1. Set up the **RKDGV1D3V** code to perform simulation of normal shock wave with Mach number 3. In particular, The following parameters need to be updated in the **DGVparameters.dat**
   1. 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.
   2. current solution base name = M380\_ES\_  
        
      use some suitable name here.
   3. 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).
   4. 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.
   5. 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.
2. Update **parameters.dat**  
   The following parameters need to be updated
   1. 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. A good place to start is 120 MFPs. The numbers should be consistent with the choices of x\_1 and x\_2 parameters used in DGV\_sf02.f90. The size of the domain is given in the dimensionless variables.
   2. 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.
   3. current solution base name = M380\_ES\_ !  
        
      Provide a meaningful file name to be used with the simulation outputs.
   4. 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.
   5. 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.
3. Update **run\_slurm.sh**The following lines need to be updated:
   1. #SBATCH -N 1   
        
      Select the number of computer nodes that you will request for the run
   2. #SBATCH --ntasks-per-node 28  
        
      do not change this one – we will always request all processorson a node.
   3. #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.
   4. 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.
4. Submit the job into execution.
5. When the process is completed copy the following results to a **local computer** OR e**xternal hard drive.** Please note that the solution files can be quite large and you may will have to use additional storage. The following files need to be saved:
   1. DGVparameters.dat, parameters.dat, run\_slurm.sh, and the output \*.txt file
   2. all output files in sol080909. Solution files will be large.
6. If storage become an issue, you may choose to reduce the total number of velocity points changing M=10, s=5 to M=11, s=3 will make solutions about 3.5 times smaller. However it is not recommended to go below 21 points per dimension for this simulation. You can also choose to keep less than 10 saves of the solution. However you would like to review the saves to decide which ones can be deleted. If you have about 64 GB of space, you will probably have enough space to complete this simulation. You can also store your files on Bridges and download one file at a time to save on the local memory.
7. Once the solutions are saved on a local computer, you would like to review them. To review the solutions, we visualize them using the provided matlab files. This will be explained in detail in the next section. You will visualize the solution corresponding to different time saves. A special attention will be paid to the last several time saves.
   1. If the solution changing shape? If so, it may have not reached the steady state and the final time needs to be extended. Check step XXX for directions on how to restart a solution from a save.
   2. Sometimes changing shape of the solution for later times suggests problems with parameters. Consult your professor about it.
   3. If the solution does not change between several saves, it reached the steady state. If the density of the solution looks like the figure below, you have accomplished a successful simulation. Otherwise, contact the professor.
8. Use directions in Steps 2.b) and 2.d) to increase the number of spatial cells and to decrease the size of the time step, keeping the ratio dt/dx about the same. The new simulation will take longer. Use directions of step3.c) to request ore time. OR use 3.a) to request more processors or both. Repeat the simulation and compare the results. If the shapes of the graphs coincide, your solution have converged. Otherwise increase the number of spatial cells and repeat.

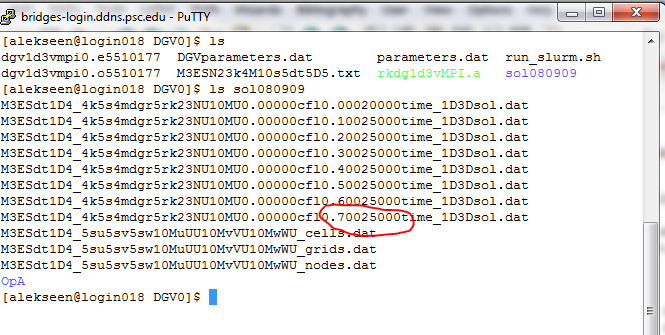
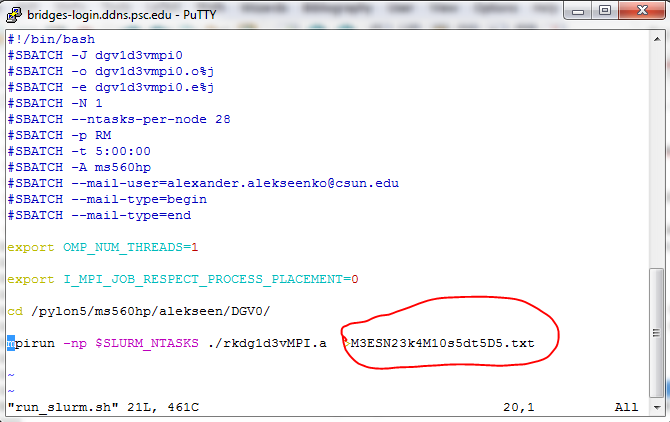
**3. Steps to visualize a solution to the normal shock wave**

Once a simulation of a shock wave is completed, we need to visualize it. A set of matlab subroutines is supplemented that can be used to plot the solution. The following steps will guide you through the process of visualizing the solution

1. Once a simulation is completed, the results of simulation should be moved to a local storage. It can be your computer hard drive or an external hard drive. **.** Please note that the solution files can be quite large and you may will have to use additional storage. The following files need to be saved:
   1. DGVparameters.dat, parameters.dat, run\_slurm.sh, and the output \*.txt file
   2. all output files in sol080909. Solution files will be large.   
        
      This is a typical look of a directory where the solution is saved  
        
        
      as you can see, the solution files are indeed pretty large.
2. Matlab subroutines are provided to visualize the computed solutions. The subroutines are available on the Canvas webpage in the archive file **vizRKDGV1D3V.zip**. The archive file contains the following Matlab files:  
     
   
3. It is recommended to create a directory where we will place the Matlab files and the solution files that we will visualize. For example, let this directory be called **VIZ\_SHOCK\_WAVE**:  
     
   inside this directory we can keep copies of the solution files that we want to visualize.  
     
   Please note that to visualize the solution you will need the **\_1D3Vsol.dat** file and the **\_nodes.dat** file that came with it.
4. Open the file **view\_macroparameters.dat** in Matlab. To visualize the moments of the solution, you will need to provide names of the appropriate files to the code and set values of a few parameters. Below is the guidance to this step:
   1. In **line 8** provide the name of the solution file that you would like to visualize, e.g.,   
      
   2. In **line 28** provide the name of the \_nodes.dat file that corresponds to the solution file appearing in line 8.  
        
      
   3. In line 49 you need to specify the array of points in variable x at which the moments will be computed. We use the the Matlab construct “Begin Value : Step Size: End Value” to create an array of points. It is important that no points fall outside the solution’s domain in variable x. So all values of x in the array x\_nodes should be between the left and right domain boundaries specified in **parameters.dat.** However, the array x\_nodes can only use a portion of the solution domain.   
        
         
        
      Last but not least, It is recommended to use between 200 and 400 points for plotting. 200 should be enough to produce a smooth graph, while using more than 400 points will slow the process down while not providing much visal benefit.
   4. Once the above parameters are set you can run the code. It will create 4 figures. In Figure 2, the graph of number density as a function of variable x is plotted. In Figure 3 the u component of the bulk velocity is plotted, in Figure 4, the graph temperature is plotted. In Figure 5, the graph of mass flux is plotted. The mass flux is constant in x in the true solution. In numerical solution, it will not be. This is a manifestation of violation of conservation laws due to truncation errors in the solution. It is important, however the mass flux is constant within 3 to 4 digits. This will guarantee sufficient accuracy of the computed solutions.
   5. **ANALYSIS of the figures:**  
      Below you can see the Figures that were computed from the solution. These figures are not good and suggest two things. First, the sharp oscillations in the solution suggest that we under resolve in variable x. So, it is recommended that we increase the number of cells in x and repeat the simulation. Second, we note that the solution is appears to be constant on the left boundary, but not so much at the right boundary. This can be cause by two things – large round off errors and strong loss of conservation (this can be due to insufficient number of cells in x). Or the solution did not reach steady state. In this case, we need to state a later final time value. You can either re-ran the solution from the initial data, that is you can either make a whole new simulation. Or you can restart the solution. The steps to restart the solution will be given in the next section.   
        
      One way to see if there were problems with conservation of mass, is to look at the Figure 5 where the mass flux is plotted. We can see that the mass flux is not constant in first digit. This is a significant violation of the conservation laws. The mass flux of the true solution is constant. It is possible that if we increase the number of cells in variable x, we will improve accuracy of discretization in x and will improve the accuracy of the conservation laws and the mass flux.   
        
      Last, but not least. To check if the final time was sufficient for the solution to reach the steady state, we can plot the closest time save of the solution and to check if the graphs of the moments are still changing. If they are still changing, the chances are the solution has not reached the steady state. If they are not changing , then it most likely has. A small shift of the moments profile indicates of violation of conservation laws.   
        
      By default the solution are plotted on top of each other. This is accomplished by the commad “hold on” that you can see after each figure. To reset the figures either use “hold off” or just close the figure window before running the graphing subroutine.  
        
      After plotting two consecutive saves of the solution, it seems that the profiles are not exactly the same, but there is no clear development. By the look of the graph, it seems that we are looking at the effects of the truncation errors caused by an insufficient resolution in variable x.   
        
      We conclude that we need to both increase the number of cells in x (twice) and reduce dt (twice) to maintain the CFL condition, and re-run the simulation.  
        
        
        
        
        
        
      

**3. Steps to restart a solution from a saved solution.**

It happens more often than not that the solution either timed out before completing or the solution completes, but the final time was not sufficient to reach the steady state. It also happens, although very rare, that the entire cluster crashes for unbeknown reasons and the solution could not be completed. In such cases it would be desired then to re-start the solution form the last saved file. This way, we do not have spend time to evolve the solution from the initial time. The code provides such option. The steps to restart the solution are provided below

1. First of all, most of the entries in DGVparameters.dat and parameters.dat has to stay exactly the same in order for the restart to work. In particular we can not chance values of velocity cells, M, s, and N. We can not also chance the values of time step dt. We should not change the names of the files. Naturally, we should not change other solution parameters, since it probably constitutes a different simulation.  
     
   We can change
   * 1. the final time
     2. the initial time (if you want to affect the intervals at which the solutions are saved)
     3. the number of times the solution is saved.
     4. the number of processors that is used to parallelized the simulation. Say you want to use more cores to run the simulation faster.
2. To tell the code to restart the solution we need to modify the following entries in **parameters.dat**
   * 1. Restart is not desired:   
          
        solution restart = no ! need\_to\_restart yes to restart  
        restart time = 0.90030000 ! restart\_time\_txt restart/last saved solution time in text format --- use the exact numbers from the filename!
     2. Restart is requested:  
          
        solution restart = yes ! need\_to\_restart yes to restart  
        restart time = 0.70025000 ! restart\_time\_txt restart/last saved solution time in text format --- use the exact numbers from the filename!  
          
        note that the value that you provide for the restart time is taken from the file name of the saved solution. In the example below the last save of the solution happened at time 0.70025000. This value of time becomes part of the file name.   
        Similarly to how these long file name are automatically generated when the solution is saved. Long names are automatically generated when solution is restarted. If you accidentally change values of N, M or S, or values of the solution name, restart will fail to generate the correct file name and will fail. So, make sure not to change other values than those listed in the previous bullet.
     3. **IMPORTANT: DO NOT FORGET to cancel the** restart request when you are setting up a new simulation. Otherwise, your code will look for a saved solution that is not there. Or, even worse, it will find a wrong saved solution, if you are not careful with naming them differently.
3. Another important parameter that you can change during a restart is the final time that is provided in **parameters.dat**:   
     
   final time = 1.0 ! final value of time  
     
   simply provide the value that is desired.   
   1. The last parameter to check in **parameters.dat** is the number of times the solution is saved. You can keep it the same, or provide a new values.   
        
      instances to save solution = 10 ! num\_save\_sol = how many times the solution (and error and other quantities) is saved during the evolution  
        
      The intervals at which saves are computed as (“final time” – “initial time”)/”instances to save solution”. So if you desired, you can also change the initial time  
        
      initial time = 0.0 ! beginning value of (time)  
        
      This will only affect the time intervals at which the solution is saved and nothing else.
4. You need to modify **run\_slurm.sh**
   * 1. You may choose to request more computational time:   
          
        #SBATCH -t 1:00:00
     2. You may change the number of processors. In this case we specify the number of nodes to use:  
          
        #SBATCH -N 1  
          
        Please follow the directions provided previously to modify the **DGVparameters.dat** so that a proper number of processors appears there. (see Part 1. 1).b))
     3. **IMPORTANT:** Modify the name of the output file by adding “\_1”, “\_restart” or similar to its name. Or use some other modification. This way you will not lose the earlier file.   
        
5. **Submit the solution to the scheduler.**