PHASTA Challenge

https://github.com/PHASTA/phasta

PHASTA is an open source stabilized finite element computational fluid dynamics (CFD) fluid flow solver. It is an acronym for Parallel Hierarchic Adaptive Stabilized Transient Analysis. It can solve either the compressible and incompressible Navier Stokes equations. The latter are appropriate when the flow speed is everywhere relatively small when compared to the speed that sound travels through the fluid while the former handles cases where the flow speed is more than 1/5th of the speed of sound.

Laminar, transitional, and turbulent flow occurs in either case and PHASTA's primary applications are to turbulent flows. Turbulence can be modeled:

a) in a statistical sense (average equations in time to get equations for average motions-> Reynolds-averaged Navier Stokes (RANS) equations),
b) resolved partially -> large eddy simulation (LES), or
c) resolved completely (at the continuum level) -> direct numerical simulation (DNS).

While laminar flow and RANS can be two dimensional when the geometry is 2D, LES and DNS are always three dimensional by the physics of turbulence. Furthermore, RANS may have steady state solutions but LES and DNS are always unsteady. The unsteady and three-dimensional nature of LES and DNS have driven PHASTA to be developed for extreme scale HPC where it has strongly scaled to 3 million processes and applied on a broad variety of platforms.

PHASTA discretizes space with kth order hierarchical basis polynomials and the finite element method leaving a system of non-linear ordinary differential equations which are further turned into non-linear algebraic equations by applying a time integrator. While explicit time integrators are available, they are often less efficient than implicit time integrators which bring the need for efficient equation solution on each linearization sub-step of a time step. To solve these linearized equations, PHASTA has native implementations of the generalized minimum residual (GMRES) method with a block diagonal preconditioner, but also can be linked to PETSc to gain access to other equation solvers and preconditioners. The native solvers have three options for handling the matrix: sparse, element-by-element, and matrix-free which trade off of storage size, flops-to-solution, and achievable arithmetic intensity. The PETSc implementation only currently supports sparse but could be extended with modest effort.

One key feature of PHASTA is its use of unstructured grids. PHASTA supports hexahedra, wedge, pyramid, and tetrahedral elements. Mesh generation can be done in MATLAB or gmsh whose files can then be imported into SCOREC-core software tools to preprocess the mesh into PHASTA inputs as an alternative to licensed commercial meshes that are typically used for more complex geometry. This pre-processing also splits the mesh into roughly equal sized chunks

using graph partition libraries from (par)Metis or Zoltan. SCOREC-core also provides adaptive meshing procedures that can be tightly linked to evolving PHASTA solutions.

Finally, ParaView and other Kitware tools such as Catalyst support loose (Paraview has a reader for both of PHASTA's solution output formats) or tightly integrated (e.g., in situ) post processing and visualization.

Assignment:

A slightly modified airfoil geometry (**AirfoilBigDom2.geo**) has been provided in the SCC input data location. A compatible **geom.spj**, **adapt.inp**, **run.sh, flow.pht, flowBar.pht,** and **solver.inp** files are also provided.

For this assignment, you should not need to change the **geom.spj** file as the initial and boundary conditions that it sets are fixed across all 3 problems. Some additional background on how to change the **.geo** file to change the mesh resolution will be provided below. Likewise, some additional background on how to change **solver.inp** will also be provided below. Previous videos have already covered the only change to **adapt.inp** (**splitFactor**) and file paths. The ParaView **pht** files will always need to be edited to point at time steps you want to visualize.

Note you will need a **gmsh** source AFTER https://gitlab.onelab.info/gmsh/gmsh/-/commit/3a7c5a0a83ef2dd47e985ee4a065b0e3f1b49b62 or a binary built from the same to get a mesh that can be used in the remainder of the workflow. You will also need to pull a recent version of **PHASTA** and **SCOREC/core** and rebuild them.

There are three tasks with rapidly escalating difficulty. As noted in the writeup above, while no CFD problem is easy, as the Reynolds number rises, steady flow solutions cease to exist. Similarly, two dimensional solutions also cease to exist. Furthermore, boundary layers become much thinner and broader ranges of scales appear. These exercises explore that rapid rise in complexity.

Throughout this document <inp> should be seen as a request to choose the appropriate number or string for inp in place of <inp>. For example, where you see **<#part>-procs_case** if you are doing an 8 part case, #part=8 and thus you should view this as **8-procs_case**.

As your results will be auto-graded, it is critical that your directory and file names match the requested form.

Problem 1

$Re_c = \rho * U * c / \mu = 200$. Here ρ is the density (set by the inflow Temperature (T=288),

pressure (p=101300 pa) through the ideal gas law ρ= p/(RT) and R=288.29438), U (34) is the free stream speed, c (0.5) is a reference length taken here to be the approximate chord (length of the airfoil), and μ is viscosity which we will vary to change the Reynolds number (Re). Since density evaluates to 1.22, we achieve this Reynolds number with μ=0.1037.

The provided **.geo** and **solver.inp** will be a good starting point for this flow but you should provide convincing evidence that the solution you obtain is accurate (hint: the method is second order accurate in space and either first or second order accurate in time depending on time integrator you choose—more on that below).

When run, PHASTA will produce a **forces.dat** file. Column 1 is the step number, column 7 is the force on the airfoil in the x direction, and column 8 is the force on the airfoil in y direction.

Convergence of the nonlinear residual can be checked/monitored by what the code writes into **histor.dat**. Column 1 is the step number, 2 is the wall time, 3 is the non-linear residual, and 4 is (dB) residual reduction relative to the first step of the run.

Please provide only your final/best/submitted results matching the following names: **mesh.geo**, **forces.dat**, **histor.dat**, two line plot files (described below), **solver.inp** file, **geombc.dat.<1..#part>**, **restart.<last_step#>.<1..#part>**, and the **flow.pht** file that loads it as your submission to problem 1 all in a directory named **Re200** (so yes geombc and restarts should be in **<#part>-procs_case** sub-directory). These outputs must start from step 0 with the initial condition provided by **chef**. The **forces.dat** and **histor.dat** files are written into the **<#>-procs_case** directory with every run. Thus they will be clobbered by each new run if you don't do something to maintain provenance like that provided by the **run.sh** file. Below you will find an explanation of some of the **solver.inp** parameters you might explore in this and the other two problems.

By default, PHASTA will restart from the last run. This is controlled by the file **numstart.dat** which is written into the **<#part>-procs_case directory.** If you want to restart from step zero (the initial condition) in for example an 8 part directory:
echo 0 > 8-procs_case/numstart.dat
from the run directory where your **solver.inp** sits.

Problem 2

Re_c=2000 is achieved by scaling viscosity down by a factor of 10.

Please provide only your final/best/submitted results matching the following names: **mesh.geo**, **forces.dat**, **histor.dat**, two line plot files (described below), **solver.inp** file, **geombc.dat.<1..#part>**, **restart.<last_step#>.<1..#part>**, and the **flow.pht** file that loads it as

your submission to problem 2 all in a directory named **Re2000**.

Problem 3

Re_c=20000 is achieved by scaling viscosity down by a factor of 10 again or 100 from the original.

Please provide only your final/best/submitted results matching the following names: **mesh.geo**, **forces.dat**, **histor.dat**, two line plot files (described below), **solver.inp** file, **geombc.dat.<1..#part>**, **restart.<last_step#>.<1..#part>**, and the **flow.pht** file that loads it as your submission to problem 3 all in a directory named **Re20000_2p5D.** That directory should hold 2.5D results (one element in the z direction). This case should then be repeated as a full 3D case as noted below. Thus, the results from this repeated case should be put in a 4th directory named **Re20000_3D**.

Additional requirements

The above are the base submissions.

For the case(s) that have a steady solution that should be sufficient. For the case(s) that yield periodic or at least close to periodic solutions, you should also create and fill in the following table in a file named **force.stats** (2 by 3 table of numbers related to the force in x (fx) and the force in y (fy) which are the 7th and 8th column of **forces.dat** (first column in time step number which you can multiply by your time step to get time))

mean(fx)	rms(fx)	dominate frequency (fx)
mean(fy)	rms(fy)	dominate frequency (fy)

For case(s) that are not steady the line plots should be extracted from time-averaged fields. PHASTA has an input to trigger the collection of time averaged fields that are written into the restart files which will be described below. Note however that time averages are over a single run. Your time averaged fields should be from a run that is AFTER the initial transient and long enough (or as long as you have time for) to converge the time average. Thus, for these cases, you should submit two sets of the output. The initial transient portion should be as described above. The second set of the requested files that you are averaging over should be in a sub-directory named **Averaged**. For example**, ReXXX/Averaged** should have the set of files used to study averaged flow after the transient that is documented in **ReXXX** cases that don't have a steady state. Note further that ANY changed parameters may cause a transient so it is important that the results in your Averaged directory have the solver.inp (except turning on averaging) and continue from that run. The easiest way to do this is:

1) run the transient from step 0 and for sufficient steps to get through the transient noting the time step that is past the transient as N_step_transient_end.
2) `mkdir Average; cd Average`
3) `cp ../solver.inp ../run.sh .`
4) `mkdir <#procs>-procs_case; cd <#procs>-procs_case`
5) `ln -s ../../<#procs>-procs_case/g*.`
6) `ln -s ../../<#procs>-procs_case/restart.<N_step_transient_end>.* .`
7) `echo < N_step_transient_end>  > numstart.dat`
8) `cd ..`
9) edit the solver.inp file to turn on time averaging and set the steps to be long enough to cover several periods of the lowest frequency in the unsteadiness. If you have a doubt about what is long enough, make it twice as long as you think might be good enough and then you can compare the time average of the first half (in a ParaView pht file that loads the restart from half the interval which will contain a time average from N_step_transient_end to that step) and a time average from the full run (second pht file that loads the restart from the full interval).
10) submit the run.sh script

The starting and ending points for the two line plots are
Line 1 (use ParaView *PlotOverLine* and File-> Save Data and name this **line1.dat** in each case)
Start 0.35 -0.125 Z/2
End 0.35 0.125 Z/2

Line 2 (use ParaView *PlotOverLine* and File-> Save Data and name this **line2.dat** in each case)
Start 0.65 -0.125 Z/2
End 0.65 0.125 Z/2

Where Z is the domain width. Please keep PlotOverline defaults (1000 pts) in csv format. When you save the data be sure to set Precision to 12 and check Use Scientific Notation

The third case should be run as 2.5D (one element in the z direction) and as full 3D with at least 32 layers in the extrusion and a domain width of Z=0.2.

TIPS

Here are some important inputs that could affect both the accuracy and the time to solution (both are scored). As there are MANY input parameters, only a small set needed to perform this challenge are described here (e.g., you should be able to leave the rest of the parameters in the **solver.inp** alone).

```
Number of Timesteps: 2000   # sets the number of time steps for the current run
Time Step Size: 2.0e-4      # larger values get to steady state faster but if unsteady flow,
                            # smaller time steps are needed for accuracy
Viscosity:  1e-2   # μ in the writeup above.  This is is how you change the Reynolds number
Number of Timesteps between Restarts: 50  # pretty movies with low numbers but fills disk
```

```
Print ybar: False          # toggle to True to collect time averaged fields mentioned above
Solver Type: GMRES sparse  #  default that you will get if you enter this or choose nothing
                           #  alternatives include Solver Type: PETSc but must build that
                           #    Solver Type: GMRES EBE  #    Solver Type: GMRES Matrix Free

# the following control how much "work" goes into each solve
     Number of GMRES Sweeps per Solve: 1               # replaces nGMRES
     Number of Krylov Vectors per GMRES Sweep: 200     # replaces Kspace
     Tolerance on Momentum Equations: 0.05
     Number of Solves per Left-hand-side Formation: 1   # nupdat/LHSupd(1)

# the following control accuracy vs stability (at larger time time steps seeking steady states
# of the time integrator.  First Order is fastest to steady states but is less time accurate
# than Second Order
     Time Integration Rule: Second Order                # Second Order sets rinf next
# and this parameter controls temporal damping if Second Order (ignored for first order)
     Time Integration Rho Infinity: 0.25                # rinf(1) Only used for 2nd order
 Number of Elements Per Block: 128  # can influence cache performance and thus cost
 Step Construction  : 0 1 0 1 0 1.  # controls non-linear iterations
                                    # 0 1 is a single solve so this is 3 per time step
```

NOTE it is very easy to change inputs to a level that the code diverges/crashes (usually when the temperature goes negative). Most common cause of this is too large of a time step and/or too low of convergence on a given time step (too few of iterations). It is part of the challenge to find combinations of the above parameters that get acceptably accurate simulations in a minimum time. For the first two problems in this challenge, answers within 5% of the fully converged value will get full points and then wall clock time to reach that solution will be scored separately. As the third problem will likely be a stretch for the available resources, progress towards a converged solution will be scored.

Note however, that there is a dimension beyond changes you make to the solver.inp. Changes to your .geo file, after run through gmsh will produce varying mesh resolution that govern the spatial accuracy of your simulation. Below is a quick review of how the .geo file controls the element sizes. Here too there is a tradeoff. A fine mesh will be more accurate but this raises the cost of the simulation in a few ways. First, more elements and nodes takes more time for the computer to process. Second, finer meshes often require smaller time steps for stability. Finally, on each non-linear iteration, the equations are often more stiff taking more Krylov vectors in GMRES and/or more non-linear iterations.

Adding a few comments to selected lines from the provided .geo file (this is NOT a complete geo file but this has more descriptive comments).

```
//+
SetFactory("OpenCASCADE");
lc = 5.0e-1;               // set a global sizing variable

Point(1) = {0,0,0,lc/100}; // create a point at 0, 0, 0 with a size that is 100 x smaller than lc

// gmsh will attempt to size elements around points based on that 4th entry so you can control
// your element sizes this way.

// you can also control your mesh sizes with fields like this  see gmsh documentation and
// tutorials and or just tweak the example provided

Field[3] = Distance;
Field[3].PointsList = {16};
Field[3].CurvesList = {11};
Field[3].Sampling = 100;

Field[4] = Threshold;
```

```
Field[4].InField = 3;
Field[4].SizeMin = lc / 12.5;
Field[4].SizeMax = 10*lc;
Field[4].DistMin = 0.30;
Field[4].DistMax = 5;

//  IT IS VERY important to use boundary layer meshes in this challenge.  The documentation in
// gmsh is not very good for this.  That said, you should be able to tweak the example provide
// with the commented “tips” below about what the parameters change
Field[11] = BoundaryLayer;
//+
Field[11].CurvesList = {1, 3, 33, 2, 4,5};
//+
Field[11].Size = 3.0e-4;
// For Higher Re 3.0e-4 will likely need to be made smaller. How much smaller? That is part
// of the challenge BUT it is suggested that you visualize your solution after each trial
// and confirm that you have “good” resolution of the boundary layer near the wall (decrease
// until you do)

//+
Field[11].SizeFar = 0.01;   // governs how large the wall normal spacing grows by “end” of BL
Field[11].Thickness = 0.05; // governs how thick the BL grows


Mesh.SaveWithoutOrphans=1;  // This is critical and why you need a recent version of gmsh.
// without this our tool chain cannot process the mesh created by gmsh as it contains the
// the points and elements we use for sizing that are not actually part of our mesh.
```