Phys 449: Real-Time Weather Simulator Notebook
Winter 2020

Written by: Noah LeFrancois
Project Partner: Filipe Rodrigues
Supervisor: Prof. Hong Guo

This document contains:

1) links to the web pages and documentation necessary for downloading, installing, and running each of the softwares used in this project;
2) a description of the files I have compiled for running each step of my project;
3) a flow chart depicting an overview of our simulation pipeline;
4) some additional information about how I put these files together and how to use them
5) my daily notes recorded throughout the semester, which contain details of the various approaches we attempted, what worked, what failed, what we learned from those failures, and ideas for next steps.
6) some concluding remarks are made about my efforts in this notebook to provide background info which will be useful to the next student who attempts to extend this project.

List of Important Links:

For installation, documentation, and tutorials

Gmsh (Mesh Generation Software):

http://gmsh.info
http://gmsh.info/doc/texinfo/gmsh.html
https://gitlab.onelab.info/gmsh/gmsh/-/tree/master/
https://www.youtube.com/watch?v=ZK8_RxVKuUE&feature=youtu.be&t=548

SU2 (Simulation Software):

https://su2code.github.io
https://su2code.github.io/docs_v7/home/
https://github.com/su2code

ParaView (Visualization Software):

https://www.paraview.org

List of Important Files:

Gmsh:

t1LayersConnected: My initial adaptation of tutorial t1 to generate layers of rectangles.

BallTest_Filled/t5-BallTest.py, BallTest_Filled/t5-BallTest.msh, BallTest_Filled/t5-BallTest.su2:

1) Adaptation of t5 to generate layered spherical shells. This only works for n<=2 layers, but that's adequate for our purposes. This isn't quite what we want yet as the central cavity (i.e. the actual planet) is still filled with mesh, and we want only the atmosphere volume to be meshed.
2) The second two files are the resultant .msh file and the .su2 obtained by exporting from the .msh.

UnityTerrain/terrain.stl, UnityTerrain/terrain.geo, UnityTerrain/nestedSpheres.geo:

1) Filipe's planet terrain generated in Unity and converted to a .stl file.
2) The .geo obtained by opening the .stl in gmsh

3) An example of the technique I’m planning to use with our terrain. By clicking Geometry>Edit script in this .geo file, you can see the steps performed. Two concentric spheres are generated, their Boolean Difference is applied by clicking Geometry>Elementary entities>Boolean>Difference (Cut). Finally the inner and outer surfaces are labelled to use as boundary markers for SU2 by clicking Geometry>Physical groups>Add>Surface. To apply this with terrain.geo, the plan would be to convert the terrain 2D surface to a 3D volume (this is the step we have not yet figured out), add a larger sphere around it for the outer atmosphere, and then proceed as described for the construction of nestedSpheres.geo

SU2:

config_template.cfg:
This file is provided with the SU2 software download (see “List of Important Links”), and contains a list of all of the simulation options available in SU2. This includes different solvers for convection, turbulence, time-stepping, etc as well as options such as multi-grid methods and convergence monitoring. The function which is called to use each feature is accompanied by a list of the acceptable input arguments. Further documentation for many of these features are found on the SU2 website (see “List of Important Links”).

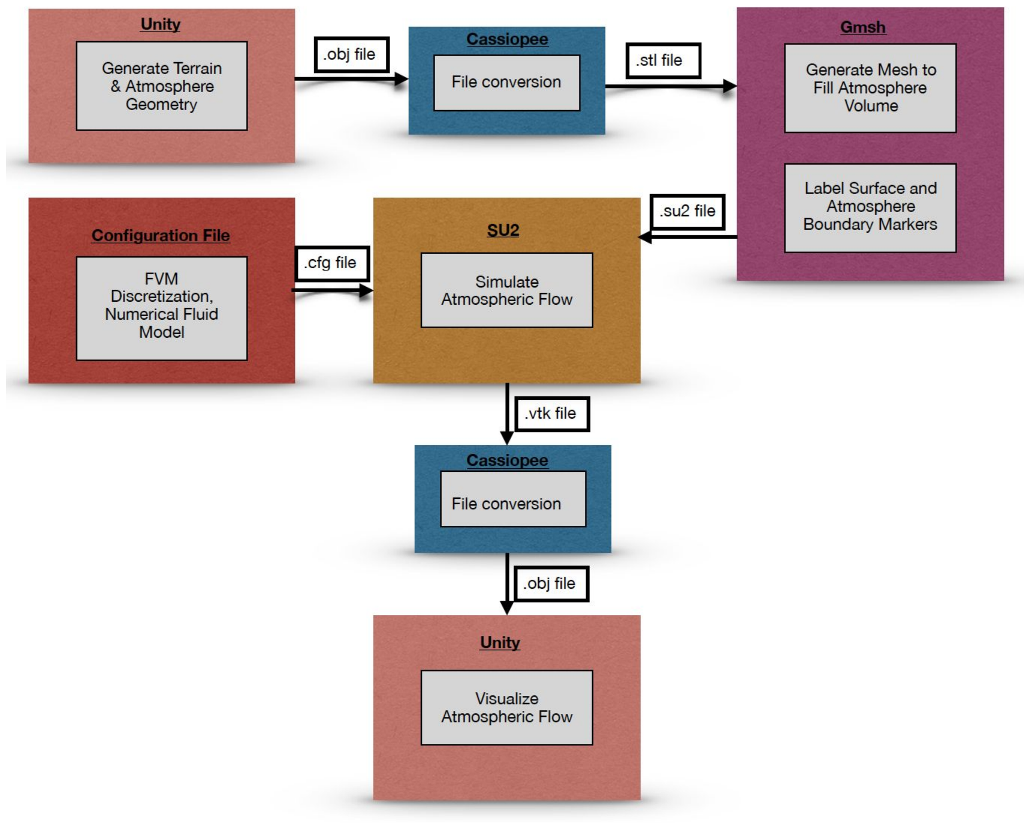
BallTest_Filled/t5-BallTest.su2, BallTest_Filled/TimeStepBall.cfg, BallTest_Filled/BallTest_flow_00001.vtk:

1) Same .su2 file as is contained in the Gmsh folder for BallTest_Filled
2) Configuration file for the SU2 simulation. This controls the physical governing equations, time marching, boundary conditions, convection method, and input/output information as described in my final report.
3) Flow output file for this simulation.

NestedSphereTest/nestedSpheres.su2, NestedSphereTest/TimeStepBall.cfg, NestedSphereTest/NestedSpheres_flow_00001.vtk:

1) .su2 file output from UnityTerrain/nestedSpheres.geo.
2) Configuration file with input changed to take the .su2 from this folder.
3) Flow output file for this simulation.

Procedure Outline:


Unity
Generate Terrain & Atmosphere Geometry
.obj file
Cassiopee
File conversion
.stl file
Gmsh
Generate Mesh to Fill Atmosphere Volume
Label Surface and Atmosphere Boundary Markers
.su2 file
SU2
Simulate Atmospheric Flow
Configuration File
FVM Discretization, Numerical Fluid Model
.cfg file
.vtk file
Cassiopee
File conversion
.obj file
Unity
Visualize Atmospheric Flow

This pipeline diagram is explained in more depth in my final report. The configuration file I used for my ball test and nested spheres test was put together by adapting the configuration file of one of the provided examples in the SU2 github folder download (see “List of Important Links”). I modified the necessary lines to implement my chosen features and input arguments drawn from config_template.cfg (see “List of Important Files”), and deleted the features which were not necessary for our model.

I used the ParaView software for visualizing my .vtk simulation data outputs; this won’t be necessary once the visualization component of our project in Unity is completed, but it is a good tool for quickly/easily checking SU2 results. A number of other visualization softwares will work with .vtk files if you want to use something else, but it doesn’t really matter what you choose because it won’t be integrated back into the pipeline for anything.
Gmsh can be operated through a few different techniques: using the Gmsh GUI, editing the .geo script which can be accessed by clicking Modules>Geometry>Edit script (click Reload script to apply your changes), and running a separate Python script (other languages can be used, this is described in more detail in the Gmsh documentation). My use of each of these three techniques is documented in my daily notebook below, as well as in the Gmsh website & documentation (see “List of Important Links”). An example of my use of a Python script is shown in t5-BallTest.py, which is built upon the t5.py Gmsh tutorial.

Fri 17Jan2020

Setup to run .cfg

Run these 3 lines to setup the SU2_RUN command needed to run the file. First line is the path to the folder containing SU2_CFD and similar files.

```
export SU2_RUN=/Users/Documents/SU2/bin
export PATH=$SU2_RUN:$PATH
export PYTHONPATH=$SU2_RUN:$PYTHONPATH
```

Navigate to the directory in which the .cfg is stored, then run it:

```
cd SU2/SU2-master/TestCases/unsteady/square_cylinder/
SU2_CFD turb_square.cfg
```

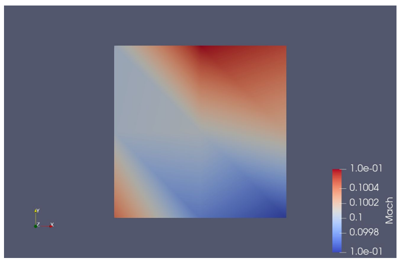
Configuration Template

The file config_template.cfg in the SU2-master folder contains all of the allowed .cfg commands and their possible inputs.

Sat 18Jan2020

Basic square mesh test

/Users/Documents/SU2/su2code.github.io-master/SquareTest
Used the example of a small 3x3 2D square mesh from the Users Guide/Mesh File page. Repurposed inviscid_bump.cfg from the first tutorial. Had to change the marker names for boundary conditions, but then it ran fine. Next step is to play with making a larger 2D grid and then a 3D grid using the square.py script provided for making meshes.



Tue 21Jan2020

Larger square mesh test

Can change xNodes and yNodes to change grid size. Tried to add a Z coordinate option but I'm not sure what needs to be done in the nested loops.

Built computer w/ Filipe, still need to do start-up and network connection before installing the SU2, ParaView, and mesh generation software. Need to document these installations and write the procedure so Hong can also get it set up on his laptop.

Thurs 23Jan2020

3rd-party mesh generation

Met w/Hong and agreed that finding 3rd-party mesh generation software will be much faster and more adaptable than developing the mesh generation script myself from the simple square example. Various options are presented in this slideshow:
https://www.nianet.org/wp-content/uploads/2019/10/08-09-19-NIA-SU2-workshop-Making-Your-Own-Mesh-min.pdf
2 main candidates are gmsh open-source software and the two example files included in the slideshow for a sphere and a hemisphere. Will continue reading gmsh documentation and try to download; Hong has experience with this software and it looks fairly powerful based on example screenshots.

Fri 24Jan2020

Gmsh setup

Downloaded and tried to set up some points and lines manually. Not very intuitive, will need to do a bunch of tutorials. Confirmed that the mesh can be exported as a .SU2 file. Downloaded the tutorial files in python format from the gmsh git. Need to get the "binary SDKs" from http://gmsh.info/bin/ to import gmsh in python and run the tutorials.

Mon 27Jan2020

Gmsh install procedure:

Download software development kit (SDK) from http://gmsh.info/#Screenshots (not sure if we also need the folder from the first option, "download gmsh", in order to get the user interface program and place it in the Applications folder. Will check this when I install on the workstation.

cd into the directory gmsh-4.5.1-MacOSX-sdk (or equivalent for Windows) is stored

Run the terminal command to add the lib directory from the SDK to PYTHONPATH:

```
export PYTHONPATH=${PYTHONPATH}:${PWD}/lib
```

Run the terminal command to run the desired example or script:

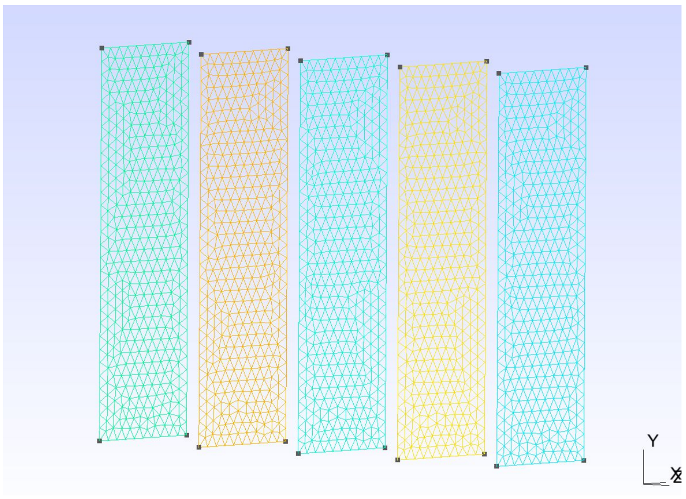
```
python share/doc/gmsh/demos/api/t1.py
```

Tues 28Jan2020

Generating 3D rectangular mesh from t1 tutorial

Edited t1.py to make 2 layers, needed to list the desired corners, lines, curveloops, planesurface, and physicalgroups. Did this manually and then wrote a bunch of for loops to do it for arbitrary number of layers. Have not yet connected the layers so it is just a stack of 2D rectangles with adjustable separation, *dz*. Next steps are generating the .SU2 file to check if marker tags are present and whether the layers need to be connected in order to run an actual simulation.

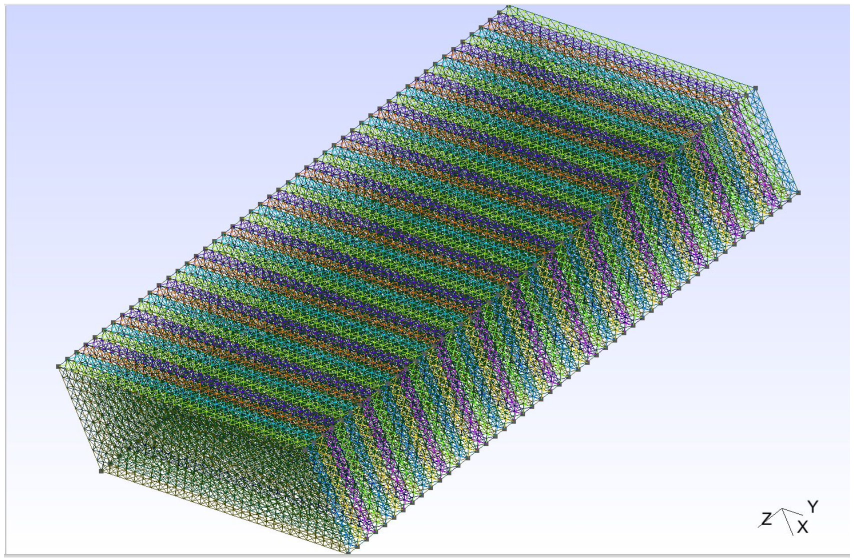
Update: doesn't look like exporting to .SU2 file produces marker tags. Will need to figure out how to add them either manually or automate it in the script/GUI.



Wed 29Jan2020

Connect layers of 3D rectangular mesh

Added curveloops and planesurfaces for the sides of the rectangular mesh cubes, then added them to the physicalgroup necessary for including these surfaces in the mesh. Result is a series of connected layers with spaces in between them; this can be made to approximate a continuous vertical mesh by making dz very small and the mesh density can be adjusted by changing lc, the characteristic length of the mesh triangles.



Fri 31Jan2020

Continuing Tutorials, Checking marker format

Messed with t3 tutorial, used factory.revolve to rotate a rectangular base through 360 degrees in order to make a rectangular torus. Not sure if I could make a spherical shell out of this by manipulating the base and using a few combined base shapes, but this offers nice connectivity vertically and horizontally when the final argument (whether to recombine the mesh) is set to 'False'.

In order to specify boundary markers, need to list the boundary points under the heading:

```
% Boundary elements
%
NMARK= 5
MARKER_TAG= inlet
MARKER_ELEMS= 75
```

The boundary points should be listed under the corresponding boundary marker, and not listed in the main list at the beginning of the file, which is a list of 'Inner element connectivity'.

The node coordinates are listed after 'Inner element connectivity', under this heading:

% Node coordinates
%
NPOIN= 18780

Will need to figure out how to specify the node coordinates and boundary elements. For a concise example of the necessary format, see: /Users/Documents/SU2/su2code.github.io-master/SquareTest

Mon 3Feb2020

Continuing Tutorials, Circle Arcs

Messed with t4 tutorial, used factory.addCircleArc to make a semi-circle then revolved this around 2pi in order to make a sphere. Need to explore inclusion of holes in the domain, might be able to put a hole in the shape of a smaller semi-circle inside in order to make the planet, although this might not allow us to create realistic terrain.

Tues 4Feb2020

T5, Holes & Spheres

Idea: could use this definition of a spherical volume to create a ball in the middle with a surrounding spherical atmosphere. The function Cheesehole() in T5 generates a sphere.

T6, T7, T8: Not relevant

T6 just seems to demonstrate transfinite meshes, which have curved grids. This doesn't seem particularly relevant to us.
T7 demonstrates using a background mesh for post-processing: "Characteristic lengths can be specified very accurately by providing a background mesh, i.e., a post-processing view that contains the target mesh sizes." Also doesn't seem very relevant.
T8 demonstrates animations and more post-processing, as well as "scripting" and "options". Also doesn't seem very relevant.

T9: Post-processing Plugins (levelsets, sections, annotations)

Unfortunately the sphere used for this demonstration is just loaded straight from a post-processing (.pos) file, view3.pos, which means we can’t use it to set up our geometry. However, the level-sets visualization could be useful at some point to examine our geometry once it’s been set up.

Marker Format:

Investigated the .SU2 files output from gmsh again, found that they do in fact contain Node Coordinates (NC) and Boundary Markers (BM). BM for the stacked rectangle layers seem to be placed at each layer of rectangle, not on the connective scaffold sides. For the semi-circle they were only placed on the original surface and not on the extruded/revolved layers. Not sure whether this is going to work but my next step is to run a simple rectangle inlet/outlet test and see if the BMs can be placed properly.

Thurs 6Feb2020

Installation on Workstation

Downloaded the necessary files for SU2, Gmsh, Python 3.7, and Anaconda. Some difficulties with the command line installation stuff, gmsh seems to be installed but when I try to run t1.py it doesn’t recognize the module ‘gmsh’ when the script tries to import it. For SU2, I tried to set up the SU2_RUN command and wasn’t given any errors, but when I try to run SU2_CFD inv_channel.cfg, I get the error : the term ‘SU2_CFD’ is not recognized as the name of a cmdlet, function…

Note: used ‘set’ command instead of ‘export’ command as that seems to be the Windows equivalent. Not sure how legit this is.

Installed paraview but it keeps crashing every time I try to open it.

Got gmsh folder without sdk to set up the user interface program; this seems to work but I have to open files through the “open” option in the interface, can’t just open them directly from the folder.

Tues 11Feb2020

Cube domain and boundary markers

Set up the cube domain by rewriting T5; when we add a PhysicalSurface (dim=2) the marker tag seems to be correct but the interior points don't get filled, so I copied the marker tag into the output file without the PhysicalSurface.
EDIT: Realized my physicalVolume (dim=3) had the wrong input index, once this was fixed it seems to output the file properly.

Output file seems to be a text file instead of a .su2 file, need to figure out how to make this work to input for SU2.
EDIT: Trying to just copy the text and overwrite a duplicated existing .su2 file.

Working in /Users/Documents/SU2/su2code.github.io-master/CubeTest ; modified squareTest.cfg by changing the boundary markers and input SU2 file.

Received the following error:

```
Error in TokenizeString(): line in the configuration file with no "=" sign.
Look for: 10 35 39 23 44 0
str.length() = 16
libc++abi.dylib: terminating with uncaught exception of type int
Abort trap: 6
```

That string of numbers is the first line of the NELEM list.

Wed 12Feb2020

Formatting .su2 file, getting Filipe's terrain

Wrote a python reformatting script to correct the spacing formatting of the .su2 output from Gmsh, formatSU2.py.
Discussed my Gmsh script and his terrain generation with Filipe and discussed how to write his geometry as points, lines, loops, and surfaces and send them to me so that I can generate a mesh on it that will have much more controllable and realistic terrain.

Thurs 13Feb2020

Finishing & Testing formatSU2

Added the getLastChunk function to the script so that it can finish the file from the final startIndex; test.su2 file appears to follow the correct format now upon visual inspection re: spacing, headers.

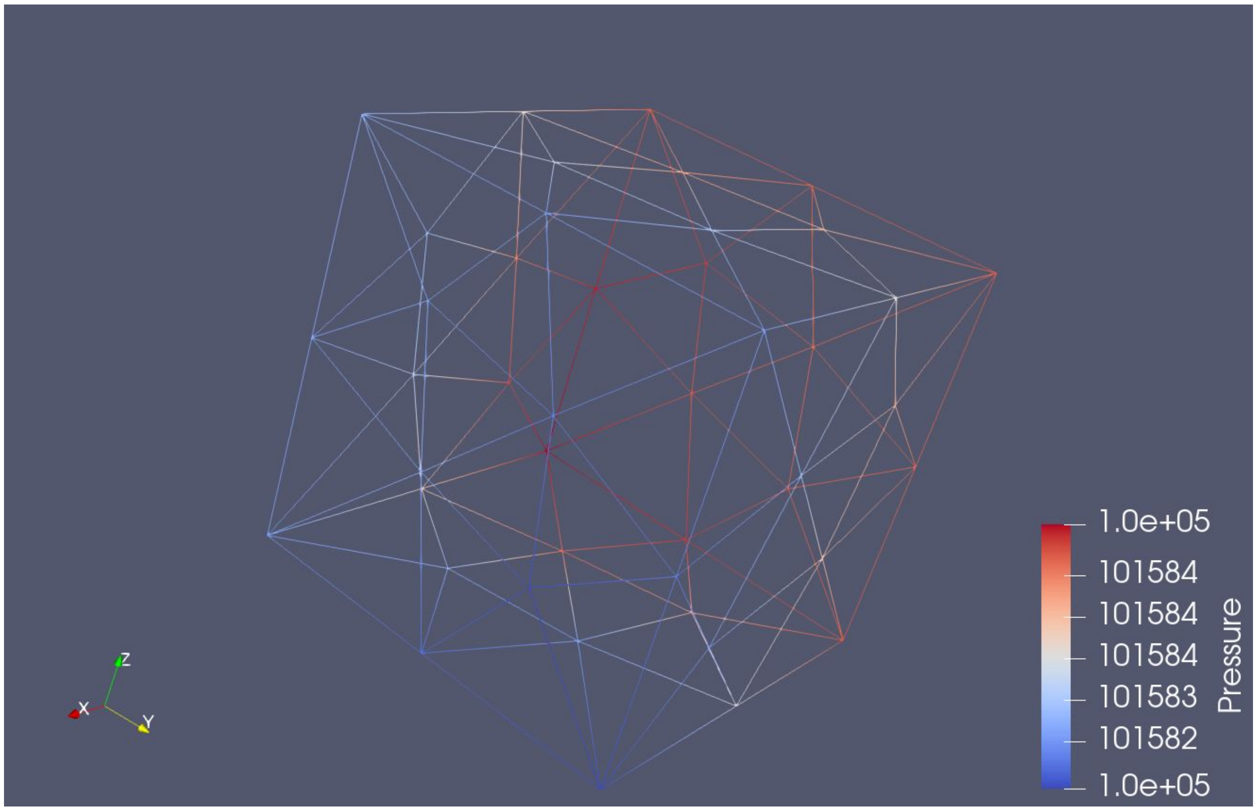
Loading test.su2 into squBox.cfg: received the same error as before (Tuesday).

Pretty sure I was entering the command wrong...re-tested on some SU2 tutorials I've already done, first one didnt work (Inviscid_ONERAM6) but unsteady square cylinder worked. Came back to the CubeTest folder, was able to run squBox.cfg using test.su2 but it crashes early, able to run squBox.su2 and it runs for the length specified on this line in the "COMMON PARAMETERS DEFINING THE NUMERICAL METHOD" section:

```
% Number of total iterations
ITER= 50000
```

Successfully finishes simulation and terminates, saving flow.vtk file. Doesn't look like a very interesting solution but that's just a matter of setting up the simulation methods and parameters now; looks like we can successfully generate mesh using Gmsh and insert it into .SU2.

Modified turb_square.cfg to match the squBox.cfg parameters but with time-stepping (step size 0.0015s). Shown below it the outer surface wireframe plotting Pressure at timestep 49.



Fri 21Feb2020

Spherical geometry with inner&outer boundary markers

Edited t5-BallTest.py to create a PhysicalGroup(dim=2) at each layer of the concentric spheres; when the gmsh is exported to .SU2 this gives two boundary markers so we can set the BCs at the planet surface and outer atmosphere.
Filipe is working on exporting his terrain to an OBJ and then to a usable gmsh file for me, which will hopefully replace this spherical shell geometry setup, but this should be:
 1) also a relevant procedure for labelling those boundaries
2) a nice test case for working examples of SU2 simulation configurations

Working in /Users/Documents/SU2/su2code.github.io-master/BallTest
Modified TimeStepsquBox to input t5-BallTest.su2 and use Euler wall conditions at the inner and outer boundaries.

Mon24Feb2020

Installing SU2 on workstation

In ubuntu command line, enter the following command to access directories in Users with the same command line interface as a mac.
cd /mnt/c/Users/blablabla

Mon9March2020

Manually meshing in Gmesh

Used the video from Filipe to add boundary markers and mesh the volume between surface and outer atmosphere. https://www.youtube.com/watch?v=ZK8_RxVKuUE&feature=youtu.be&t=548

Able to do this for the cylinder example, only issue with the planet geometry is that i'm not sure how to click on the inner surface since the outer surface is in the way. If I'm not able to get around this by zooming or something similar, I could start with just the inner surface, then merge with a file containing the outer surface. Either way this should be an essentially resolved problem.

Wed11March2020

Adding Gravitational Source Term

https://www.cfd-online.com/Forums/su2/129165-add-source-terms-user-defined-boundary-condition-set-up-transient-simulation.html
From SU2 forum: "To add a source term in SU2 is easy but you should be familiar with C++: There is an elegant way based on creating a class in Numerics (files numerics_direct_mean.cpp, and numerics_structure.hpp). The class CSourceGravity is a good example. Once you have created the class, you should go to Numerics_Preprocessing subroutine (definition_structure.cpp) and allocate the class numerics_container[iMGlevel][FLOW_SOL][SOURCE_FIRST_TERM] = new CSourceGravity(nDim, nVar_Flow, config);

There is a quick way to add a source term just modifying the residual in the subroutine void CEulerSolver::Source_Residual(CGeometry *geometry, CSolver **solver_container, CNumerics *numerics, CNumerics *second_numerics, CConfig *config, unsigned short iMesh)”

In SU2_CFD/src/solver_direct_mean.cpp, looks like we can modify the source terms for the CEulerSolver:

1) Looks like we should set gravity = FALSE and implement the central gravity term using body_force = TRUE since the default gravity term doesnt work for our spherical coordinates:

```
void CEulerSolver::Source_Residual(CGeometry *geometry, CSolver **solver_container,
CNumerics *numerics, CNumerics *second_numerics,
                                   CConfig *config, unsigned short iMesh) {

  unsigned short iVar;
  unsigned long iPoint;
  bool implicit          = (config->GetKind_TimeIntScheme_Flow() == EULER_IMPLICIT);
  bool rotating_frame    = config->GetRotating_Frame();
  bool axisymmetric      = config->GetAxisymmetric();
  bool gravity           = (config->GetGravityForce() == YES);
  bool harmonic_balance = (config->GetTime_Marching() == HARMONIC_BALANCE);
  bool windgust          = config->GetWind_Gust();
  bool body_force        = config->GetBody_Force();
```

2)
https://www.cfd-online.com/Forums/su2-shape-design/143792-optimization-source-term-ns-equation.html
This is from the SU2 forum, posted by Thomas Economon (SU2 developer). Supports 1) which indicates we need to use Source_Residual in solver_direct_mean to implement the source term:

“Indeed, it is straightforward to include source terms in the flow and adjoint problems.

You can add an additional source term in the direct problem in the routine CEulerSolver::Source_Residual(...) in the file solver_direct_mean.cpp. You might follow the rotating frame source term at the top of the routine as an example.”
This

3) Looks like we need to implement a “new child class in CNumerics” which will calculate the source term. CNumerics is a class in the file SU2_CFD/include/numerics_structure.hpp.

```
void CEulerSolver::Source_Template(CGeometry *geometry, CSolver **solver_container,
CNumerics *numerics,
                                   CConfig *config, unsigned short iMesh) {

  /* This method should be used to call any new source terms for a particular problem*/
  /* This method calls the new child class in CNumerics, where the new source term should
be implemented.  */
```

```
  /* Next we describe how to get access to some important quanties for this method */
  /* Access to all points in the current geometric mesh by saying: nPointDomain */
  /* Get the vector of conservative variables at some point iPoint =
nodes->GetSolution(iPoint) */
  /* Get the volume (or area in 2D) associated with iPoint = nodes->GetVolume(iPoint) */
  /* Get the vector of geometric coordinates of point iPoint = nodes->GetCoord(iPoint) */

}
```

4) This thread from the forum involves a longer exchange on how to implement a custom source term: https://www.cfd-online.com/Forums/su2/112529-custom-source-terms.html

In numerics_direct_mean we could edit the body force function: CSourceBodyForce; below this are a number of examples of body force options which have been implemented.

5) In the configuration template file, the following chunk is given:

```
% ----------------------- BODY FORCE DEFINITION -------------------------------%
%
% Apply a body force as a source term (NO, YES)
BODY_FORCE= NO
%
% Vector of body force values (BodyForce_X, BodyForce_Y, BodyForce_Z)
BODY_FORCE_VECTOR= ( 0.0, 0.0, 0.0 )
```

It looks like this is a uniform body force in a chosen direction. We could possibly edit this to be pointed at the planet COM for all points. Not sure how exactly BODY_FORCE_VECTOR is defined.

Thurs26March2020

Making 3D mesh of .stl Terrain

Trying to generate a mesh inside the annulus between a sphere and the terrain surface shell.
Can create a sphere by selecting geometry>elementary entities>add>sphere.
In order to perform boolean operations: select the desired boolean, change “selection mode” to “volumes” using the drop-down menu, click on the first volume and press e, then click on the second volume and press e. The prompts should ask you to “select object” and then to “select tool”.
Options:
-Intersection, gives the common volume
-Union, gives the combined volume of both objects

-Difference, gives the part that is not shared (i.e. opposite of intersection)
-Fragments, gives a coherent union such that the intersection is re-meshed to be its own self-contained volume instead of being part of both intersecting meshes and therefore being meshed twice.

Plan: Loading the terrain file (LowerBound) and creating a large sphere around it, use the difference in order to just select the annulus.

Fri27March2020

Making 3D mesh of .stl Terrain

The Boolean plan above was successful with two spheres, however it requires the centres of the spheres to be slightly offset so that we can still click on the second volume. May be able to do this using a script command that would avoid having to offset them.

<u>Update:</u> Succesful using the following script command to remove the part of 1 which is not inside 2:

```
//+
Sphere(1) = {0, 0, 0, 5, -Pi/2, Pi/2, 2*Pi};
//+
Sphere(2) = {0, 0, 0, 10, -Pi/2, Pi/2, 2*Pi};
//+
BooleanDifference{ Volume{2}; Delete; }{ Volume{1}; Delete; }
```

Now using the real LowerBound.stl terrain file:
Inner radius is around 3.05, using an outer sphere of radius 4.
Used “selection mode” = “volumes” for object then “surfaces” for tool to select the outer then inner. Receive error message:

```
Error   : Unknown OpenCASCADE entity of dimension 2 with tag 1
Error   : '', line 3 : Could not apply boolean operator
```

Looks like i can’t do a boolean operation with two objects of different dimensions.

This post seems to be discussing our problem but it basically seems to use our procedure from above (add elementary entities>Volume, then Mesh>3D), which doesn’t work:
https://fengl.org/2012/05/21/converting-stl-surface-mesh-to-volume-mesh-using-gmsh/

Background Material for Paper:

- Oreskovic, Alexei (September 14, 2018). "Why EA's former boss believes the 3D tech that powers video games will make way more money outside of gaming". *Business Insider*. Archived from the original on September 15, 2018. Retrieved October 17, 2018.

LeVeque, Randall J. *Numerical Methods for Conservation Laws", Birkhauser Verlag, 1992, p. 125.*

Balakrishnan, N. and Fernandez, G. (1998), Wall boundary conditions for inviscid compressible flows on unstructured meshes. Int. J. Numer. Meth. Fluids, 28: 1481-1501. doi:10.1002/(SICI)1097-0363(19981230)28:10<1481::AID-FLD776>3.0.CO;2-B

T.D. Economon (2018) Simulation and Adjoint-based Design for Variable Density Incompressible Flows with Heat Transfer. American Institute of Aeronautics and Astronautics, Multidisciplinary Analysis and Optimization Conference.

https://www.tandfonline.com/doi/abs/10.1080/10407790500314947

Mon30March2020

Conversion to 3D mesh of .stl Terrain

This forum convo has people attempting to solve what I believe is the same as our problem.
http://onelab.info/pipermail/gmsh/2017/011221.html
This seems to be addressed by tutorial t13, however this fails to run with an error message of:

```
Warning : Unable to open file 't13.stl'
Traceback (most recent call last):
  File "t13Test/t13.py", line 13, in <module>
    gmsh.merge("t13.stl")
  File "/Users/Documents/School/Phys449/gmsh-4.5.1-MacOSX-sdk/lib/gmsh.py", line 244, in merge
    ierr.value)
ValueError: ('gmshMerge returned non-zero error code: ', 1)
```

Since we're able to load a .stl into gmsh to create a .geo file, my thought is that we could load that .geo file into python and then follow the procedure of t13. I found a python library that seems to be capable of loading .geo files:
https://stackoverflow.com/questions/25126506/how-to-import-gmsh-geo-file-in-fipy
https://www.ctcms.nist.gov/fipy/fipy/generated/fipy.meshes.html#module-fipy.meshes.gmshMesh

Concluding Remarks

Unfortunately due to the shortening of the semester due to COVID-19 we will not be able to make any further progress on the project this semester. I have done my best towards the end of this notebook to document my searches for materials pertaining to possible solutions to our current problems, such as spherical coordinates, gravitational source term, and conversion of 2D .stl terrain to 3D .geo terrain. These and some other plans I had for improving our SU2 numerical model from the very basic test model I have built are discussed in greater length in my final report for this course. While we were hoping to have a completed pipeline and a working VR visualization by the end of this semester, extreme external circumstances prevented us from reaching this goal. However, our work up to this point has shown promising indications of feasibility; most of the problems we have run into seem to have been solved previously by users on CFD forums, we have simply run out of time to develop and fully iron out these solutions. We are hopeful that further work will be able to achieve a functioning model, and eventually a useful real-time weather simulator built upon the foundation developed this semester.

Good luck!
Noah LeFrancois