In MATLAB, create an algorithm that determines if a 2-Contact Grasp (using hard contacts) satisfies force closure.

For each case, your code will do the following:

1. Determine if the contacts satisfy force closure
2. Visualize the friction cones of each contact in 3D
   - Plot3, Plot, trisurf, patch may be useful tools for plotting
   - Note: hold on/off allows you to add data to existing plots
3. Generate plots of the wrench space. (6 plots)
   - i.e., Visually show force closure is satisfied by projecting the grasp wrench space (GWS) in XY, XZ, and YZ plots for force and torque

For the algorithm, ***assume the following:***

- The grasp wrench space only needs to resist the grasp (i.e., ignore gravity)
- The center of mass (CoM) is at the World Frame's / Universal Frame's origin
- Contacts can be any orientation in 3D space
- Normalize the force vector to 1N prior to making the friction cone
- Preferably configurable, friction cones should be approximated with 8 facets/vertices.

Please start programming using the attached project code

- Project Matlab Files: ENSC894_MatlabProjectFiles.zip
- Download ENSC894_MatlabProjectFiles.zip
  - This includes code that reads from a CSV and a sample CSV file. Do not change this
  - This includes code that saves images to an output directory. Do not change this
  - This includes an external library, inHull. inHull can test if any n-D point is inside or outside an n-D convex hull. Click the link for more details.
    - You need to define the n-D convexhull and n-D point for testing.
  - This above library needs convexhulln
  - To help generate the n-D convexhull. For a grasp wrench space (GWS), each vertex should be 6-D.
    - **QHull Issue 1**: Use opt={'QJ'} to avoid degenerate conditions. **(i.e. _output=convhulln(_input, {'QJ'}); (QJ: joggled input to avoid precision problems)**
    - **QHull Issue 2**: If all vertices tested have a common 0 in the column/row, QHull will complain.
      Replace that column/row with random points slightly offset from zero.
      - **Column Replacement:** e = -0.01 + (0.01+0.01)*rand(16,1);
        Row Replacement: e = -0.01 + (0.01+0.01)*rand(1,16);
  - **Final Test CSV file:** <Not provided yet>

Report:

Attach a report that does the following:

- Explains your algorithm and limitations
- Summarizes/Prints all results from the Final Text CSV File
- Has a discussion section that explains what was learned/observed by the algorithm