Assignment 7: Hydrostatic forces in a ship hull

Overview

In this assignment you will utilize numerical integration to analyze the forces due to hydrostatic pressure acting on a ship hull. A schematic of a cross section of a ship hull in a rolling configuration is shown in Figure 1. The stability of the ship depends on the balance of forces and moments arising from forces such as the weight (W) of the ship and the buoyancy force (B). The hydrostatic pressure acting on the surface of the hull creates the buoyancy necessary for floatation.

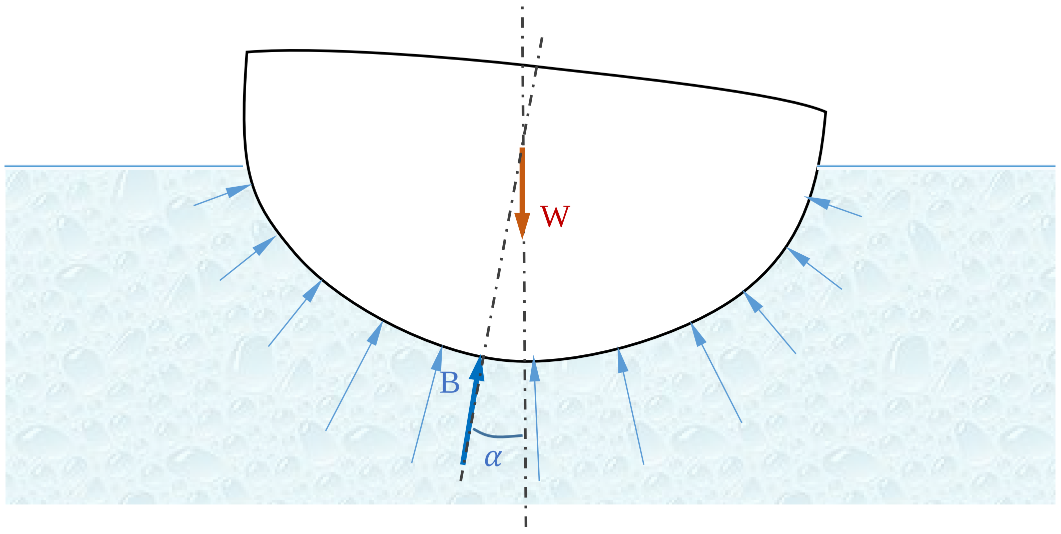


Figure 1: Schematic of section of a ship hull in rolling motion.

Instructions and deliverables

Part 1: quadrature module

The first part of this assignment is straightforward if you have successfully completed the in-class exercises from Lessons on numerical integration. Specifically, we would like you to resubmit your **`quadrature.py`** file that defines the **`midpoint`**, **`trapezoidal`**, and **`gauss_quad`** functions. This part is included to ensure the integration routines are working correctly before you modify or apply them in the subsequent parts.

Part 2: Module for estimating forces from pressure on surfaces

Create a module file named **`forces_surface.py`**. This module will contain three functions, which are described below.

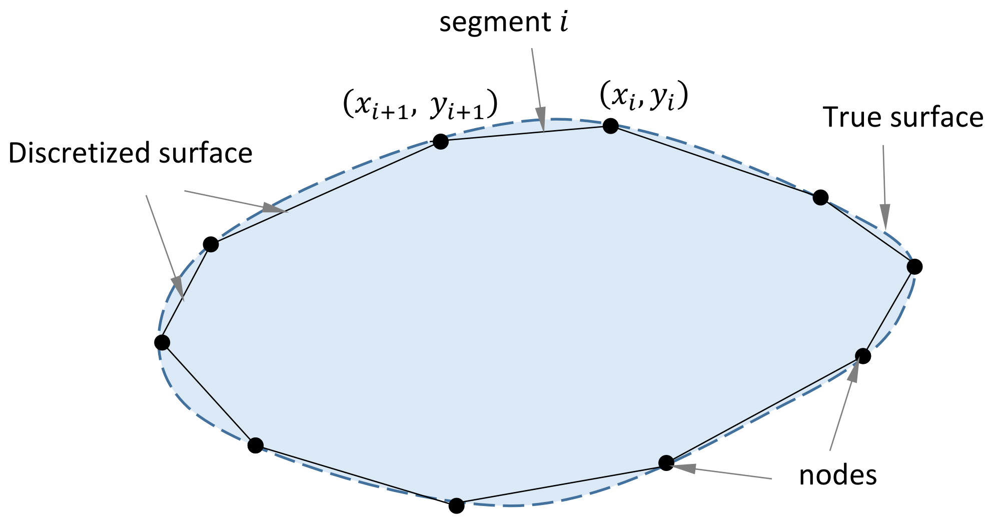


Figure 2: Illustration of discretized surface

`def panel_midpoints(x, y):`

- The parameters x and y are **`numpy`** arrays of the **`(x, y)`** coordinates of the segment or *panel nodes*: the little dots (called nodes) in Figure 2. For example, x_i from Figure 2 would be stored in **`x[i]`**, and y_i would be stored in **`y[i]`**.
- Figure 2 shows an example of a closed surface with m=10 segments, where the ith segment has nodes (x_i, y_i) and (x_{i+1}, y_{i+1}) on either side. In case of open surfaces (as the ship hull example) of m segments, the number of nodes will be m+1. But in case of closed surfaces, m+1 nodes are still stored for the m segments, even though the (m+1)th node will be redundant with the 1st node, in the same way that the ith node is redundant as the first edge of segment i and last edge of segment i-1.
- The function should compute the midpoint coordinates of each panel. The midpoint coordinates should be returned as two NumPy arrays. There should be one midpont for each panel, with no redundancies.

The second function in the **`forces_surface`** module is called **`panel_midforces`**. This function computes the x- and y-components of the force that each panel experiences. The function's definition should be

`def panel_midforces(x, y, press):`

The function must satisfy the following requirements:

- The parameters **`x`** and **`y`** are defined exactly like they are for **`panel_midpoints`**; that is, they are the coordinates of the panel nodes in Figure 2.
- <u>Important</u>: you can assume that the user gives you the node coordinates x and y ordered in a counter-clockwise manner over the surface of the body. This is an important assumption when computing the normal to the panels.
- The parameter **`press`** is the coefficient of pressure measured at each of the panel nodes.
- The function should return two NumPy arrays, let's call them **`dfx`** and **`dfy`**. The entries **`dfx[i]`** and **`dfy[i]`** should hold the approximate pressure-force components on panel **`i`** in the x and y direction. The exact pressure force on panel **`i`** is given by

$$\boldsymbol{df} = \int_{s_i}^{s_{i+1}} -p\, \boldsymbol{n}_i ds,$$

where $\boldsymbol{df} = [df_x, df_y]$ is the force vector on the panel and $\boldsymbol{n}_i = \left[n_x, n_y\right]_i$ is the unit normal vector perpendicular to panel i. You can approximate this force using the average measured coefficient of pressure and the normal vector scaled by the length of the panel, Δs:

$$\boldsymbol{df} \approx -\frac{1}{2}(p_i + p_{i+1})\, \boldsymbol{n}_i\, \Delta s$$

The formula on the right is what you should use to compute **`dfx[i]`** and **`dfy[i]`**. You will have to determine how to calculate $\boldsymbol{n}_i\, \Delta s$.

- Hint: one approach is to consider finite differences, centered at the panel midpoint, of the coordinates **`x[i+1]`** and **`x[i]`** and **`y[i+1]`** and **`y[i]`**. What does a finite difference give you?

The third and final function in the **`forces_surface`** module takes the panel forces and integrates them to find the total x and y components of the force. The function is called **`integrate_forces`** and should be defined as

```
def integrate_forces(dfx, dfy):
```

and must satisfy the following requirements:

- **`dfx`** and **`dfy`** are arrays containing the x and y components of the forces on each panel: in practice, they will be the arrays produced by function **`panel_midforces`**.
- This function should return two scalars (not arrays), for example **`force_x`** and **`force_y`**, that hold the estimated total force components.
- Since **`dfx`** and **`dfy`** are approximate values of the forces at the panel midpoints, you can use the **`midpoint`** function to integrate the panel forces **`dfx`** and **`dfy`**; just use **`dx=1.0`** since the interval widths have already been accounted for in **`dfx`** and **`dfy`**.

Part 3: Plot of force distribution on the surface

In the final part of the assignment, you will use the **`forces_surface`** module you developed in Part 2 to make a plot displaying the contribution of forces on each segment of the hull. The coordinates of nodes and the pressure applied on each node is available in the text file **`hydro_pressure.dat`**. There are four columns of data in the file:

1. The angle (theta) measured counter-clockwise from the positive x-axis (in radians);
2. The x coordinates of the panel nodes;
3. The y coordinates of the panel nodes; and
4. The coefficient of pressure measured at the panel nodes.

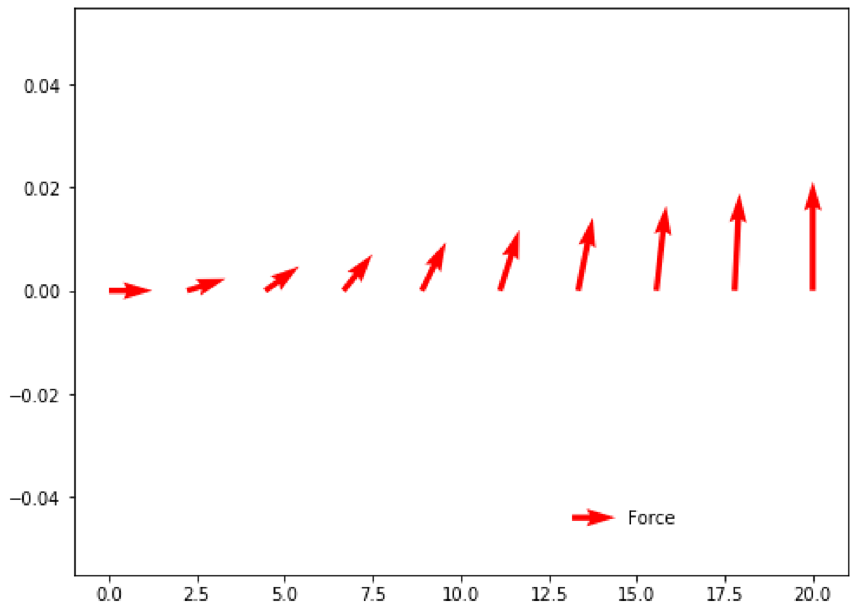
Each row in the file corresponds to one node. You can read the data into NumPy arrays using **`np.loadtxt`**, as given below.

The plot should follow the guidelines below:

- Use a separate file named **`hw07_plot.py`** to create the plot; do not add to the **`forces_surface.py`** file.
- Plot the surface by plotting its panel node coordinates. Use a solid black line plot.
- Import **`forces_surface`** and use **`panel_midforces`** to get the force on each panel. In addition, use **`panel_midpoints`** to get the panel midpoints. Finally, plot the forces on each panel as red vectors (i.e. arrows), centered at the midpoints.
- The matplotlib function quiver shall be utilized to plot arrows as given by the following example.

```
nplot = 10
x = np.linspace(0.,20.,nplot)
y = np.linspace(0., 0.,nplot)
u = np.linspace(2., 0.,nplot)
v = np.linspace(0., 5.,nplot)
fig,ax= plt.subplots(1,1,figsize=(8,6))
q = ax.quiver(x,y,u,v, color='r',units='xy', scale=2.0)
ax.quiverkey(q, X=0.7, Y=0.1, U=2.0, \
             label="Force", labelpos ='E')
plt.show()
```

The output image of the example plot is given below.



Here, the quiver function takes the start coordinates of the arrows in the **`numpy`** arrays **`x`** and **`y`**, while the **`u`** and **`v`** are the **`numpy`** arrays containing the vector coordinates of the arrow.

- Compute the total force by using the **`integrate_forces`** function. Then, generate a text string that displays the force components as shown in the sample figure.

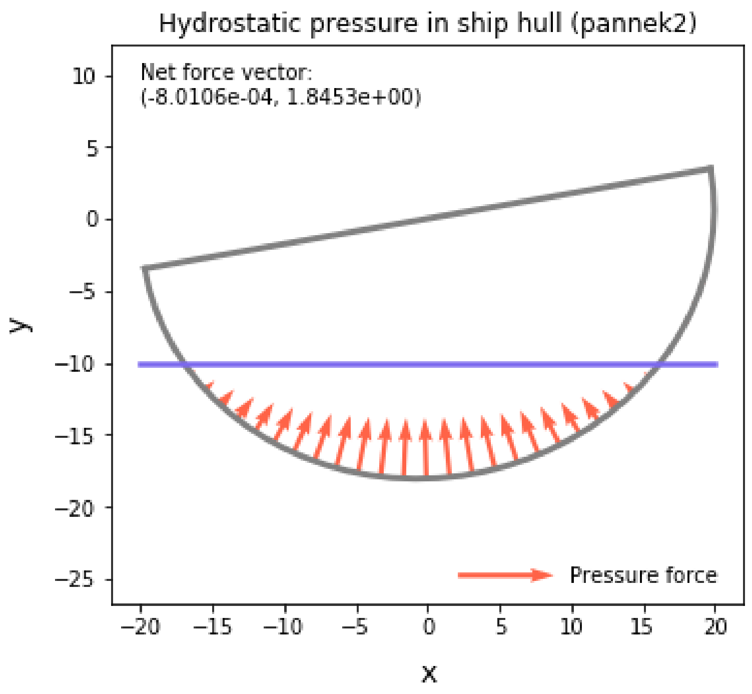
```
mega = 1.e-6
text_string = "Net force vector:\n("
text_string += '{:6.4e}'.format(force_x*mega)
text_string += ", "
text_string += '{:6.4e}'.format(force_y*mega)
text_string += ")"
plt.text(-30.0,10., text_string)
```

where **force_x** and **force_y** are the values returned by **integrate_forces**. The forces are expressed in mega newtons.

- You might do this by constructing a **title_string** as outlined below (with the details left to you, but be sure to include your RCSid)

```
title_string = "Hydrostatic pressure in ship hull (pannek2)"
plt.title(title_string)
```

- Label the x and y axes.
- Use **ax.axis('equal')** to ensure the x and y axes scales are equivalent in the plot; that is, a distance of 1 unit in the x direction is the same as a distance of 1 unit in the y direction. Adjust the x and y ranges so that the blue force arrows can be seen completely.
- You may need to supply a keyword argument to **labelpad=** in **ax.set_xlabel** (and for y label) to get your axis labels to show properly. And/or, you may need to use **ax.set_position([xlo, ylo, xhi, yhi])** to make the plot you print to file fit within the print bounds. As always, inspect your plot before you submit it.
- Save the figure as a **png** file named **hydrostatic_force.png**, and submit it to Submitty with the other parts of the assignment.
- A sample plot is given below.



For details of quiver and quiverkey functions refer the following matplotlib documentation:

https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.quiver.html https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.axes.Axes.quiverkey.html#matplotlib.axes.Axes.quiverkey