Extreme Computing: Homework 1

Michael S. Emanuel, Jonathan Guillotte-Blouin, Erick Ruiz, Yue Sun

February 25, 2019

1 Problem 1: Couette Flow

Couette flow is a classic, basic flow pattern in which a fluid is confined between two, smooth parallel plates. The bottom plate (at $y = -h$) is stationary, while the top plate (at $y = h$) is moving with a constant velocity U. It is assumed that the flow is steady, fully developed, two-dimensional (i.e. independent of z), and there is a zero pressure gradient. Figure 1 shows a diagram of the flow configuration.

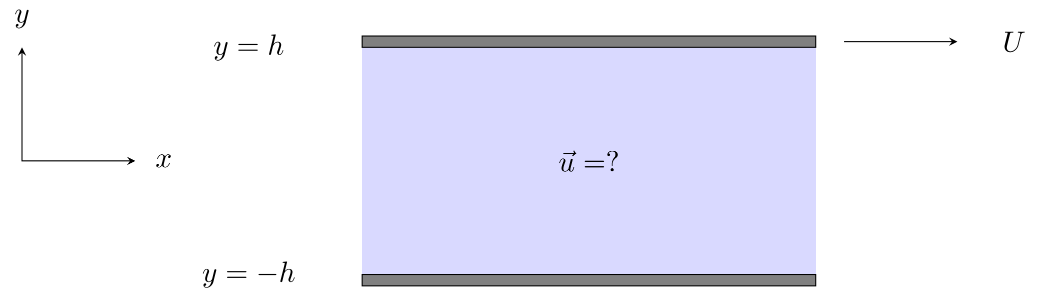

Figure 1: A schematic of the Couette flow configuration, with the velocity profile included.

Use no-slip boundary conditions to calculate the velocity and vorticity fields, the components of the shear stress tensor, the volume flow rate, and the average and maximum velocities in the channel. Also, make a plot of the velocity profile at a few different values of U. Be sure to put y on the y-axis and $u(y)$ on the x-axis of your plot.

Solution

Let us start with the Navier-Stokes equations for an incompressible fluid.

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \vec{\nabla})\vec{u} = -\frac{1}{\rho}\vec{\nabla}p + \frac{\mu}{\rho}\nabla^2\vec{u} + \vec{f} \tag{1}$$

$$\vec{\nabla} \cdot \vec{u} = 0 \tag{2}$$

Assuming that the external body forces acting on the fluid are negligible allows us to omit the last term on the right-hand side of the momentum equation.

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \vec{\nabla})\vec{u} = -\frac{1}{\rho}\vec{\nabla}p + \frac{\mu}{\rho}\nabla^2\vec{u} \tag{3}$$

Also, under the steady, fully developed flow assumption, the total time derivative can be taken as zero.

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \vec{\nabla})\vec{u} = 0 \tag{4}$$

Furthermore, if the pressure gradient is zero, we can drop the first term on the right-hand side of equation (3), which allows us to write the simplified momentum equation as

$$\nu \nabla^2 \vec{u} = 0, \tag{5}$$

where $\nu = \mu/\rho$ is the kinematic viscosity. Notice that we may rewrite the last equation as

$$\nabla^2 \vec{u} = 0 \tag{6}$$

since it is understood that the kinematic viscosity is non-zero.
Now, we are also told that the flow is two-dimensional, and since it is assumed that the flow does not vary along x, we may write $\vec{u} = u(y)\ \hat{x}$. Notice that we have concluded that $u = u(y)$, which allows us to write a single ordinary differential equation for $u(y)$.

$$\frac{d^2 u}{dy^2} = 0 \tag{7}$$

We can then solve equation (7) by direct integration to obtain the following solution.

$$u(y) = c_1 y + c_2 \tag{8}$$

At this point, we must apply the boundary conditions. Remember that we are working with no-slip boundary conditions, which implies that $u(h) = U$ and $u(-h) = 0$.

$$u(-h) = 0 = -c_1 h + c_2 \tag{9}$$
$$u(h) = U = c_1 h + c_2 \tag{10}$$

Solving equations (9) and (10) simultaneously for c_1 and c_2 yields $c_1 = U/2h$ and $c_2 = U/2$. Thus, the velocity field for this particular geometry is

$$\boxed{\vec{u} = \frac{U}{2}\left(\frac{y}{h} + 1\right)\ \hat{x}.} \tag{11}$$

Once we have the velocity field, it is straightforward to obtain the remaining relevant quantities. Let us start with the vorticity field, which we obtain by simply taking the curl of the velocity field.

$$\boxed{\vec{\omega} = \vec{\nabla} \times \vec{u} = -\frac{U}{2h}\ \hat{z}} \tag{12}$$

We now turn our attention to the shear stress. First, recall that the components of the stress tensor for an incompressible fluid may be written as follows.

$$\sigma_{ik} = -p\delta_{ik} + \mu\left(\frac{\partial v_k}{\partial x_i} + \frac{\partial v_i}{\partial x_k}\right) \tag{13}$$

Notice that the stress tensor is symmetric, and since we are only interested in the shear stress, we only need to calculate the off-diagonal elements.

$$\tau_{ik} = \mu\left(\frac{\partial v_k}{\partial x_i} + \frac{\partial v_i}{\partial x_k}\right) \tag{14}$$

Most of these will be zero because only the x-component of our velocity field is non-zero.

$$\tau_{xy} = \tau_{yx} = \mu \frac{U}{2h} \tag{15}$$

$$\tau_{yz} = \tau_{zy} = 0 \tag{16}$$

$$\tau_{zx} = \tau_{zx} = 0 \tag{17}$$

For compactness, we may write the end result in matrix form as follows.

$$\boxed{\hat{\tau} = \begin{pmatrix} 0 & \mu\frac{U}{2h} & 0 \\ \mu\frac{U}{2h} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}} \tag{18}$$

To obtain the volume flow rate, we simply integrate the velocity field with respect to y from $y = -h$ to $y = h$.

$$\boxed{Q = \int_{-h}^{h} \frac{U}{2}\left(\frac{y}{h} + 1\right)\, dy = Uh} \tag{19}$$

Finally, we calculate the maximum and average velocities. To do this, we note that the minimum and maximum velocities are at $y = -h$ and $y = h$, respectively. By inspection, the minimum velocity must be zero to obey the boundary condition at $y = -h$. Likewise, the maximum velocity must be $U_{\text{max}} = U$ to obey the boundary condition at $y = h$. Thus, we can calculate the average velocity as follows.

$$\boxed{U_{\text{avg}} = \frac{U(h) + U(-h)}{2} = \frac{U}{2}} \tag{20}$$

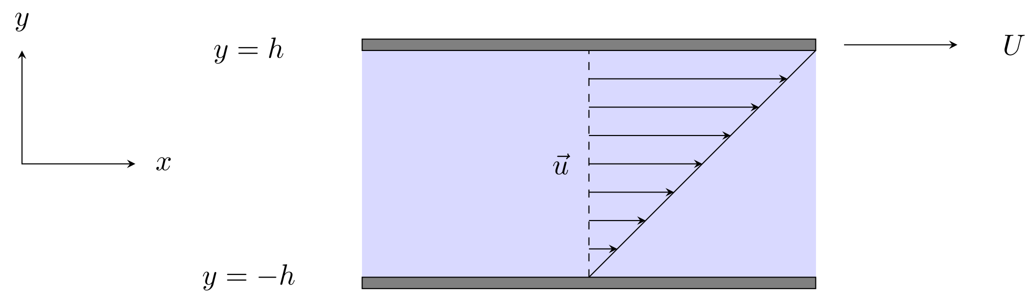

Figure 2: A schematic of the Couette flow configuration, with the velocity profile included.

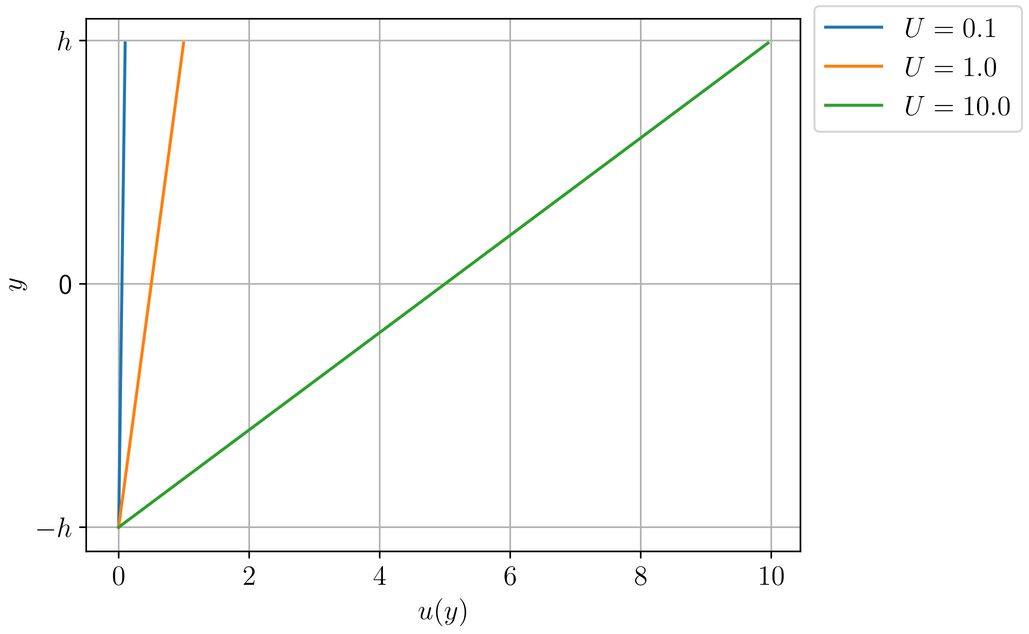

Figure 3: A plot of the velocity profile for Couette flow for $U = 0.1, 1, 10$ m/s

2 Problem 2: Rectangular Duct Flow

Consider the case where we have flow through a rectangular cross section as shown in the Figure (4). Assuming that the flow is steady, fully developed in x, and has only one component (i.e. $v = w = 0$), find the velocity field by simplifying the Navier-Stokes momentum equation and solving it using no-slip boundary conditions on all surfaces. Also, note that the flow is driven by a prescribed (known) pressure gradient in the x-direction.

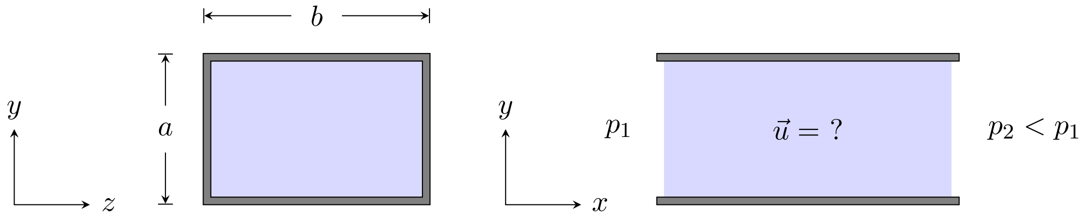

Figure 4: A schematic of the flow geometry

2.1 Governing Equations

Using the provided assumptions, show that the Navier-Stokes equations reduce to

$$\left(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) u(y,z) = \frac{1}{\mu}\frac{\partial p}{\partial x}. \tag{21}$$

What are the boundary conditions?

Solution

As before, let us start with the Navier-Stokes equations for an incompressible fluid.

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\vec{\nabla})\vec{u} = -\frac{1}{\rho}\vec{\nabla}p + \frac{\mu}{\rho}\nabla^2\vec{u} \tag{22}$$

$$\vec{\nabla}\cdot\vec{u} = 0 \tag{23}$$

Note that we have already neglected the effect of any external body forces since the problem statement makes no mention of their importance. We already know from working through the problem on Couette flow that under steady flow conditions, $\partial\vec{u}/\partial t = 0$. Furthermore, since we are dealing with a one-component flow that is independent of x (i.e. $u = u(y,z)$), then the nonlinear component vanishes.

$$\vec{u}(\vec{\nabla}\cdot\vec{u}) = 0 \tag{24}$$

Thus, obeying the assumptions given in the problem statement essentially means our velocity profile must be of the form $\vec{u} = u(y,z)\ \hat{x}$, and the Navier-Stokes momentum equation simplifies as follows.

$$\boxed{\left(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) u(y,z) = \frac{1}{\mu}\frac{\partial p}{\partial x}} \tag{25}$$

For this problem, no-slip boundary conditions imply that $u(y=0,z) = u(y=a,z) = 0$ and $u(y,z=0) = u(y,z=b) = 0$.

2.2 Solving for u: Part 1

We have an inhomogenous, linear PDE on our hands. Let's solve this using the method of eigenfunction expansions. We know that with the current boundary conditions, the homogenous solution to our equation has eigenfunctions that are sines. Hence, let's assume a solution of the form

$$u(y,z) = \sum_{n=1}^{\infty} \beta_n(z) \sin\left(\frac{n\pi}{a} y\right). \tag{26}$$

Substitute this ansatz into the governing PDE and show that the unknown coefficients, $\beta_n(z)$, are governed by the following inhomogenous ODE.

$$\beta_n'' - \gamma_n^2 \beta_n = q_n \tag{27}$$

Here, $(\cdot)'$ denotes d/dz, γ_n is the eigenvalue, and q_n and Q are defined as follows.

$$\gamma_n = \frac{n\pi}{a} \tag{28}$$

$$q_n = \frac{2}{a}\int_0^a Q \sin(\gamma_n y)\ \mathrm{d}y \tag{29}$$

$$Q = \frac{1}{\mu}\frac{\partial p}{\partial x} \tag{30}$$

2.2.1 Some Specific Details

Here is a breakdown of the required steps.

1. First show that after plugging the assumed form into the governing PDE the resulting expression is

$$\sum_{n=1}^{\infty} \beta_n'' \sin\left(\frac{n\pi}{a} y\right) - \sum_{n=1}^{\infty} \beta_n \left(\frac{n\pi}{a}\right)^2 \sin\left(\frac{n\pi}{a} y\right) = Q. \tag{31}$$

2. Next, multiply by $\sin\left(\frac{m\pi}{a} y\right)$ and integrate over y.

$$\int_0^a \sum_{n=1}^{\infty} \beta_n'' \sin\left(\frac{n\pi}{a} y\right) \sin\left(\frac{m\pi}{a} y\right)\ \mathrm{d}y - \int_0^a \sum_{n=1}^{\infty} \beta_n \left(\frac{n\pi}{a}\right)^2 \sin\left(\frac{n\pi}{a} y\right) \sin\left(\frac{m\pi}{a} y\right)\ \mathrm{d}y = \int_0^a Q \sin\left(\frac{m\pi}{a} y\right)\ \mathrm{d}y. \tag{32}$$

3. That's a pretty big mess. Now, we use a beautiful result: the orthogonality of sines:

$$\int_0^a \sin\left(\frac{n\pi}{a} y\right) \sin\left(\frac{m\pi}{a} y\right)\ \mathrm{d}y = \begin{cases} 0 & n \neq m \\ \frac{a}{2} & n = m \end{cases} \tag{33}$$

   The main point here is that every single term in the infinite sum is zero *except* for the term where $n = m$.

4. The final result is

$$\beta_n'' - \gamma_n^2 \beta_n = \frac{2}{a}\int_0^a Q \sin(\gamma_n y)\ \mathrm{d}y \tag{34}$$

What are the boundary conditions on β_n?

Solution

To proceed, we assume a solution of the form

$$u(y,z) = \sum_{n=1}^{\infty} \beta_n(z) \sin\left(\frac{n\pi y}{a}\right). \tag{35}$$

Substituting this into equation (25) yields the following.

$$-\sum_{n=1}^{\infty} \beta_n(z) \left(\frac{n\pi}{a}\right)^2 \sin\left(\frac{n\pi y}{a}\right) + \sum_{n=1}^{\infty} \beta_n''(z) \sin\left(\frac{n\pi y}{a}\right) = \frac{1}{\mu}\frac{\partial p}{\partial x} \tag{36}$$

We now exploit the orthogonality of the basis functions by multiplying through by $\sin(\gamma_m y)$ and integrating over y. We will examine each of the three terms individually to make the bookkeeping go smoothly.

$$\int_0^a \sum_{n=1}^{\infty} \beta_n''(z) \sin\left(\frac{n\pi y}{a}\right) \sin\left(\frac{m\pi y}{a}\right)\, dy = \beta_m''(z)\frac{a}{2} \tag{37}$$

$$\int_0^a \sum_{n=1}^{\infty} \beta_n(z) \left(\frac{n\pi}{a}\right)^2 \sin\left(\frac{n\pi y}{a}\right) \sin\left(\frac{m\pi y}{a}\right)\, dy = \beta_m(z) \left(\frac{m\pi}{a}\right)^2 \frac{a}{2} \tag{38}$$

$$\int_0^a \frac{1}{\mu}\frac{\partial p}{\partial x} \sin\left(\frac{m\pi y}{a}\right)\, dy = \int_0^a Q \sin\left(\frac{m\pi y}{a}\right)\, dy \tag{39}$$

Putting everything together allows to obtain an ordinary differential equation for the coefficients, $\beta_m(z)$.

$$\beta_m''(z) - \beta_m(z)\left(\frac{m\pi}{a}\right)^2 = \frac{2}{a}\int_0^a Q \sin\left(\frac{m\pi y}{a}\right) \tag{40}$$

Recall that $\gamma_m = m\pi/a$, and if we define

$$q_m = \frac{2}{a}\int_0^a Q \sin\left(\frac{m\pi y}{a}\right), \tag{41}$$

we may rewrite the previous ODE as

$$\beta_m'' - \gamma_m^2 \beta_m = q_m. \tag{42}$$

At this point, we can exchange the index m for n such that our end result can be written as follows.

$$\boxed{\beta_n'' - \gamma_n^2 \beta_n = q_n} \tag{43}$$

Recall that we are working with no-slip boundary conditions on all sides. Thus, the boundary conditions on β_n must be $\beta_n(z=0) = \beta_n(z=b) = 0$.

2.3 Solving for β_n: The Homogeneous Part

To solve for β_n, we first need to solve the homogeneous equation,

$$\beta_n'' - \gamma_n^2 \beta_n = 0. \tag{44}$$

Show that the homogeneous solution is

$$\beta_n^H = c_1 w_1 + c_2 w_2 \tag{45}$$

where $w_1 = \sinh(\gamma_n z)$ and $w_2 = \cosh(\gamma_n z)$ are the two linearly independent solutions.

Solution

We now turn our attention to solving equation (46). To do this, we first solve the homogeneous equation,

$$\beta_n'' - \gamma_n^2 \beta_n = 0, \tag{46}$$

by assuming a solution of the form $\beta_n(z) = e^{\lambda_n z}$. Substituting our ansatz into equation (46) yields the characteristic polynomial.

$$\lambda_n^2 e^{\lambda_n z} - \gamma_n^2 e^{\lambda_n z} = 0 \implies \lambda_n = \pm\gamma_n \tag{47}$$

Hence, the general solution to the homogeneous problem can be written as follows.

$$\beta_n^H(z) = ae^{-\lambda_n z} + be^{\lambda_n z} \tag{48}$$

$$= ae^{-\gamma_n z} + be^{\gamma_n z} \tag{49}$$

However, we may also write the general solution in terms of a different set of linearly independent functions. Recall that the hyperbolic sine and cosine functions are linear combinations of the exponential functions, e^x and e^{-x}.

$$\sinh(x) = \frac{1}{2}\left(e^x - e^{-x}\right) \tag{50}$$

$$\cosh(x) = \frac{1}{2}\left(e^x + e^{-x}\right) \tag{51}$$

Hence, if we let $x = \gamma_n z$, we may, without loss of generality, write the general solution to the homogeneous problem as

$$\boxed{\beta_n^H(z) = c_1 \sinh(\gamma_n z) + c_2 \cosh(\gamma_n z).} \tag{52}$$

2.4 Solving the Inhomogeneous Part

You can use the method of variation of parameters to solve the inhomogeneous equation. Let

$$\beta_n(z) = v_1(z)\, w_1(z) + v_2(z)\, w_2(z), \tag{53}$$

where $v_1(z)$ and $v_2(z)$ are the coefficients (parameters) to be varied. Now, *assume* that

$$w_1 v_1' + w_2 v_2' = 0. \tag{54}$$

Substituting (53) into (27) and using (54) yields two equations for v_1' and v_2'.

$$w_1 v_1' + w_2 v_2' = 0 \tag{55}$$

$$w_1' v_1' + w_2' v_2' = q_n \tag{56}$$

Show that

$$v_1 = \frac{q_n}{\gamma_n^2}\sinh(\gamma_n z) + \alpha_1 \tag{57}$$

$$v_2 = -\frac{q_n}{\gamma_n^2}\cosh(\gamma_n z) + \alpha_2 \tag{58}$$

where α_1 and α_2 are constants of integration.

Finally, show that

$$\begin{aligned}\beta_n(z) &= C_n\left[-\sinh(\gamma_n b) + \sinh(\gamma_n z) - (\sinh(\gamma_n z)\cosh(\gamma_n b) - \cosh(\gamma_n z)\sinh(\gamma_n b))\right] && (59)\\ &= C_n\left[\sinh(\gamma_n z) - \sinh(\gamma_n b) - \sinh(\gamma_n(z-b))\right] && (60)\\ &= 2C_n\sinh\left(\frac{\gamma_n}{2}(z-b)\right)\left[\cosh\left(\frac{\gamma_n}{2}(z+b)\right) - \cosh\left(\frac{\gamma_n}{2}(z-b)\right)\right]. && (61)\end{aligned}$$

where

$$C_n = \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}. \tag{62}$$

Solution

Following the procedure outlined in the problem statement, we will start by substituting equation (53) into equation (27).

$$\beta_n' = v_1 w_1' + v_1' w_1 + v_2 w_2' + v_2' w_2 \tag{63}$$

If we assume that $w_1 v_1' + w_2 v_2' = 0$, the previous equation simplifies as follows.

$$\beta_n' = v_1 w_1' + v_2 w_2' \tag{64}$$

From here, we can differentiate equation (64) to obtain

$$\beta_n'' = v_1' w_1' + v_1 w_1'' + v_2' w_2' + v_2 w_2''. \tag{65}$$

Putting all of this together yields the following.

$$v_1' w_1' + v_1 w_1'' + v_2' w_2' + v_2 w_2'' - \gamma_n^2(v_1 w_1 + v_2 w_2) = q_n \tag{66}$$

Now, recall that we have defined $w_1 = \sinh(\gamma_n z)$ and $w_2 = \cosh(\gamma_n z)$. Therefore, $w_1'' = \gamma_n^2 w_1$ and $w_2'' = \gamma_n^2 w_2$, and equation (66) can be rewritten as follows.

$$v_1' w_1' + v_2' w_2' + \gamma_n^2(v_1 w_1 + v_2 w_2) - \gamma_n^2(v_1 w_1 + v_2 w_2) = q_n \tag{67}$$

Notice that several of the terms cancel, leaving us with

$$v_1' w_1' + v_2' w_2' = q_n. \tag{68}$$

We now have two equations for v_1' and v_2'.

$$w_1 v_1' + w_2 v_2' = 0 \tag{69}$$

$$v_1' w_1' + v_2' w_2' = q_n \tag{70}$$

These can be written explicitly in terms of $\sinh(\gamma_n z)$ and $\cosh(\gamma_n z)$.

$$\sinh(\gamma_n z)v_1' + \cosh(\gamma_n z)v_2' = 0 \tag{71}$$
$$\gamma_n \cosh(\gamma_n z)v_1' + \gamma_n \sinh(\gamma_n z)v_2' = 0 \tag{72}$$

To proceed, we will use the first equation to isolate v_2'.

$$v_2' = -\frac{\sinh(\gamma_n z)}{\cosh(\gamma_n z)}v_1' \tag{73}$$

From here, we can substitute equation (73) into equation (72) to obtain an expression for v_1'.

$$\gamma_n \cosh(\gamma_n z)v_1' + \gamma_n \sinh(\gamma_n z)\left[-\frac{\sinh(\gamma_n z)}{\cosh(\gamma_n z)}v_1'\right] = q_n \tag{74}$$
$$\gamma_n v_1'\left[\cosh(\gamma_n z) - \frac{\sinh^2(\gamma_n z)}{\cosh(\gamma_n z)}\right] = q_n \tag{75}$$
$$\frac{\gamma_n v_1'}{\cosh(\gamma_n z)} = q_n \tag{76}$$
$$v_1' = \frac{q_n}{\gamma_n}\cosh(\gamma_n z) \tag{77}$$

Integrating equation (77) yields

$$\boxed{v_1 = \frac{q_n}{\gamma_n^2}\sinh(\gamma_n z) + \alpha_1,} \tag{78}$$

where α_1 is a constant of integration. Likewise, we can now use equation (77) to obtain a solution for v_2.

$$\begin{aligned} v_2' &= -\frac{\sinh(\gamma_n z)}{\cosh(\gamma_n z)}\left[\frac{q_n}{\gamma_n}\cosh(\gamma_n z)\right] \\ &= -\frac{q_n}{\gamma_n}\sinh(\gamma_n z) \end{aligned} \tag{79}$$

Integrating equation (79) yields

$$\boxed{v_2 = -\frac{q_n}{\gamma_n^2}\cosh(\gamma_n z) + \alpha_2,} \tag{80}$$

where α_2 is a second constant of integration. Putting all of this together allows us to write a general solution for $\beta_n(z)$.

$$\begin{aligned} \beta_n(z) &= v_1 w_1 + v_2 w_2 \\ &= \left[\frac{q_n}{\gamma_n^2}\sinh(\gamma_n z) + \alpha_1\right]\sinh(\gamma_n z) + \left[\alpha_2 - \frac{q_n}{\gamma_n^2}\cosh(\gamma_n z)\right]\cosh(\gamma_n z) \end{aligned} \tag{81}$$

We can now impose the boundary conditions to determine the constants of integration, α_1 and α_2.

$$\beta_n(z=0) = 0 = -\frac{q_n}{\gamma_n^2} + \alpha_2 \implies \alpha_2 = \frac{q_n}{\gamma_n^2} \tag{82}$$

Obtaining an expression for α_1 involves more algebra, but the idea is the same.

$$\beta_n(z=b) = 0 = \left[\frac{q_n}{\gamma_n^2}\sinh(\gamma_n b) + \alpha_1\right]\sinh(\gamma_n b) + \left[\frac{q_n}{\gamma_n^2} - \frac{q_n}{\gamma_n^2}\cosh(\gamma_n b)\right]\cosh(\gamma_n b) \tag{83}$$

$$\alpha_1 = \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}\left[\cosh^2(\gamma_n b) - \sinh^2(\gamma_n b) - \cosh(\gamma_n b)\right] \tag{84}$$

$$\alpha_1 = \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}\left[1 - \cosh(\gamma_n b)\right] \tag{85}$$

Putting all of this together allows us to rewrite (81) as follows.

$$\begin{aligned}
\beta_n(z) &= \frac{q_n}{\gamma_n^2}\left[\sinh(\gamma_n z) + \frac{1-\cosh(\gamma_n b)}{\sinh(\gamma_n b)}\right]\sinh(\gamma_n z) + \frac{q_n}{\gamma_n^2}\left[1-\cosh(\gamma_n z)\right]\cosh(\gamma_n z) \\
&= \frac{q_n}{\gamma_n^2}\left[\sinh^2(\gamma_n z) + \frac{\sinh(\gamma_n z) - \cosh(\gamma_n b)\sinh(\gamma_n z)}{\sinh(\gamma_n b)} + \cosh(\gamma_n z) - \cosh^2(\gamma_n z)\right] \\
&= \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}\left[\sinh(\gamma_n z) - \sinh(\gamma_n b) - \cosh(\gamma_n b)\sinh(\gamma_n z) + \cosh(\gamma_n z)\sinh(\gamma_n b)\right] \\
&= \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}\left[\sinh(\gamma_n z) - \sinh(\gamma_n b) - \sinh(\gamma_n(z-b))\right] \qquad (86)
\end{aligned}$$

Notice that if we let

$$C_n = \frac{q_n}{\gamma_n^2\sinh(\gamma_n b)}, \tag{87}$$

equations (60) and (86) agree. Thus, we may write the end result as

$$\boxed{\beta_n(z) = 2C_n\sinh\left(\frac{\gamma_n}{2}(z-b)\right)\left[\cosh\left(\frac{\gamma_n}{2}(z+b)\right) - \cosh\left(\frac{\gamma_n}{2}(z-b)\right)\right].} \tag{88}$$

2.5 The Velocity Field

Put everything together to show that the velocity profile can be written as follows.

$$u\left(y,z\right) = \sum_{n=1}^{\infty} 2C_n\sinh\left(\frac{\gamma_n}{2}\left(z-b\right)\right)\left[\cosh\left(\frac{\gamma_n}{2}\left(z+b\right)\right) - \cosh\left(\frac{\gamma_n}{2}\left(z-b\right)\right)\right]\sin\left(\gamma_n y\right) \tag{89}$$

A good sanity check is to make sure the boundary conditions are satisfied by this solution.

2.5.1 Other Thoughts

There are a variety of things that can be computed and inspected from here. For example:

- Calculate the vorticity and streamlines. Any surprises here?
- How does the solution change if you have different boundary conditions?
  - Neumann boundary conditions on each surface
  - Dirichlet boundary conditions on all surfaces *except* at $y = a$ for which we have a Neumann condition. (This would be like fluid flowing through an open channel.)

Note: You are not required to compute any of these for this assignment!

Solution

Recall that our analysis has assumed a product solution of the form

$$u(y,z) = \sum_{n=0}^{\infty} \beta_n(z) \sin\left(\frac{n\pi y}{a}\right). \tag{90}$$

Substituting our solution for $\beta_n(z)$ into equation (90) yields the end result.

$$\boxed{u(y,z) = \sum_{n=0}^{\infty} 2C_n \sinh\left(\frac{\gamma_n}{2}(z-b)\right)\left[\cosh\left(\frac{\gamma_n}{2}(z+b)\right) - \cosh\left(\frac{\gamma_n}{2}(z-b)\right)\right]\sin(\gamma_n y)} \tag{91}$$

From here, we can obtain other relevant quantities, such as the vorticity and streamlines. Remember that the vorticity is the curl of the velocity field.

$$\begin{aligned}
\vec{\omega} &= \vec{\nabla} \times \vec{u} && (92)\\
&= \left(\frac{\partial}{\partial x}\hat{x} + \frac{\partial}{\partial y}\hat{y} + \frac{\partial}{\partial z}\hat{z}\right) \times u(y,z)\hat{x} && (93)\\
&= \frac{\partial u}{\partial x}(\hat{x}\times\hat{x}) + \frac{\partial u}{\partial y}(\hat{y}\times\hat{x}) + \frac{\partial u}{\partial z}(\hat{z}\times\hat{x}) && (94)\\
&= \frac{\partial u}{\partial z}\hat{y} - \frac{\partial u}{\partial y}\hat{z} && (95)\\
&= \beta_n'(z)\sin(\gamma_n y)\ \hat{y} - \gamma_n\beta_n(z)\cos(\gamma_n y)\ \hat{z} && (96)
\end{aligned}$$

Substituting in the explicit form of $\beta_n(z)$ and $\beta_n'(z)$ and simplifying our expression gives the following result.

$$\begin{aligned}
\vec{\omega} =& \gamma_n C_n \left[\cosh(\gamma_n z) - \cosh(\gamma_n(z-b))\right]\sin(\gamma_n y)\ \hat{y}\dots\\
&- 2\gamma_n C_n \sinh\left(\frac{\gamma_n}{2}(z-b)\right)\left[\cosh\left(\frac{\gamma_n}{2}(z+b)\right) - \cosh\left(\frac{\gamma_n}{2}(z-b)\right)\right]\cos(\gamma_n y)\ \hat{z} \qquad (97)
\end{aligned}$$

2.6 Visualization

Now that we have an explicit formula for the velocity field, we can visualize it. Try to make the following plots:

- Plot $u(y, z = z^*)$ at a few values of z^* (perhaps near the boundary, 1/4 of the channel width, and at half the channel width).
- Plot $u(y = y^*, z)$ at a few values of y^* (perhaps near the boundary, 1/4 of the channel height, and at half the channel height).
- Make a surface plot of $u(y, z)$.

Solution

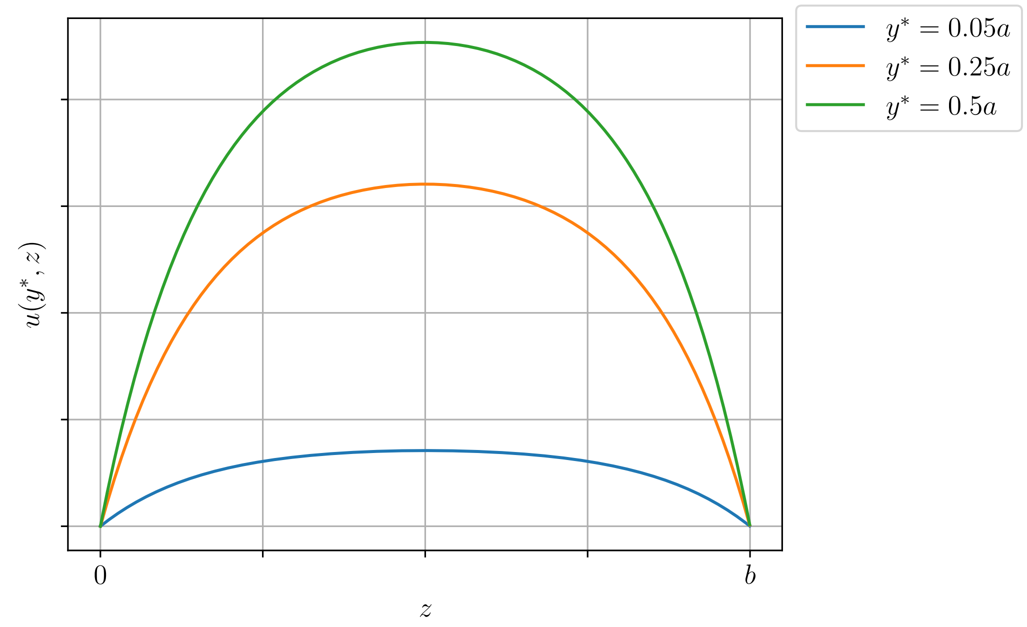

Figure 5: A plot of the velocity profiles at several values of y^*

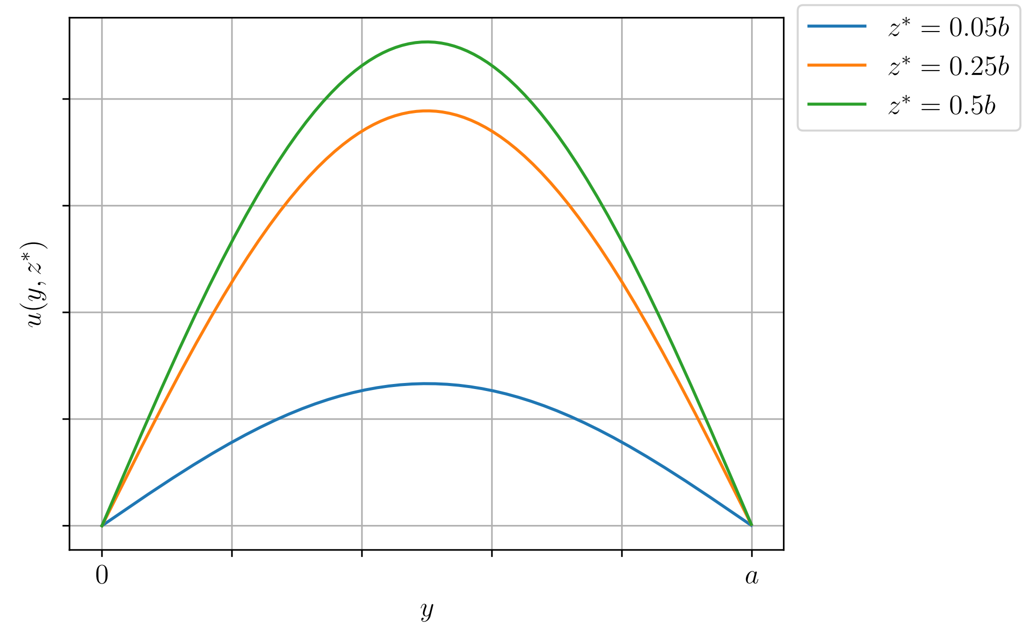

Figure 6: A plot of the velocity profiles at several values of z^*

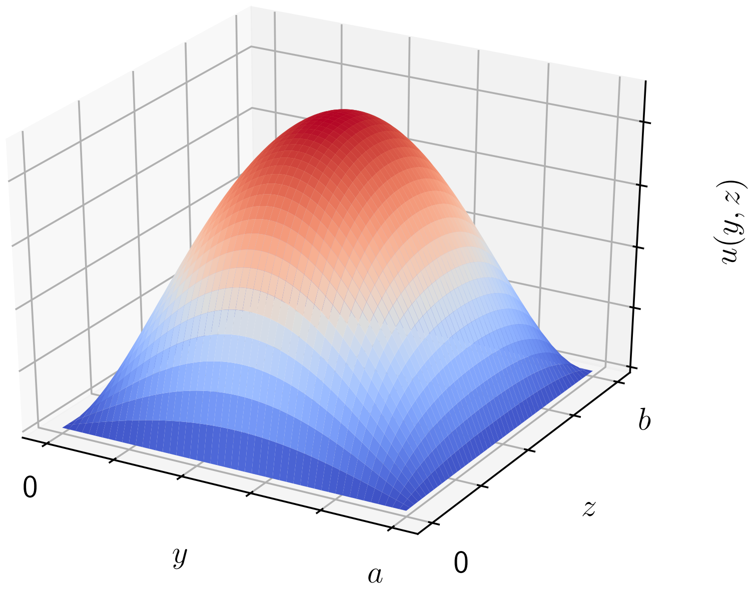

Figure 7: A surface plot of the velocity profile for flow in a rectangular duct

3 Problem 3: The Finite Element Method

Consider the steady one-dimensional advection-diffusion-reaction equation,

$$a\partial_x u - \lambda u - k\partial_x^2 u = f, \qquad x \in (0,1) \tag{98}$$

with boundary conditions

$$u(0) = u_0, \quad -k\partial_x u(1) = b_1. \tag{99}$$

In (98), a is the constant advection speed, $\lambda > 0$ is the constant reaction coefficient, and $k > 0$ is the constant diffusion coefficient.

3.1 Weak Form

Write the weak form corresponding to the strong form (98). Be sure to specify all function spaces. Don't forget to include any boundary terms.

Solution

Equation (98) denotes the strong form **S**. Multiplying **S** by a test function, $w(x)$, and integrating over the domain yields the following.

$$\int_0^1 w\left(a\partial_x u - \lambda u - k\partial_x^2 u\right) dx = \int_0^1 wf dx \tag{100}$$

$$\int_0^1 wa\partial_x u dx - \int_0^1 w\lambda u dx - \int_0^1 wk\partial_x^2 u dx = \int_0^1 wf dx \tag{101}$$

Use integration by parts on the third term on the LHS, we have

$$\int_0^1 wk\partial_x^2 u dx = k\left(w\partial_x u\Big|_0^1 - \int_0^1 \partial_x w \partial_x u dx\right) \tag{102}$$

Set $w(0) = 0$, and we know that $k\partial_x u(1) = -b_1$, substitute Eq.(102) into Eq.(101), we have

$$\int_0^1 wa\partial_x u dx - \int_0^1 w\lambda u dx - w(1)k\partial_x u(1) + w(0)k\partial_x u(0) + \int_0^1 k\partial_x w\partial_x u dx = \int_0^1 wf dx$$

$$\int_0^1 wa\partial_x u dx - \int_0^1 w\lambda u dx - w(1)b_1 + \int_0^1 k\partial_x w\partial_x u dx = \int_0^1 wf dx \tag{103}$$

Function spaces: $S = \left\{u\middle|u \in H^1, u(0) = u_0\right\}$ and $V = \left\{v\middle|v \in H^1, v(0) = 0\right\}$.

The weak form: Find $u \in S$, s.t. $\forall w \in V$, $u \in V$,

$$a\int_0^1 w\partial_x u dx - \lambda\int_0^1 wu dx + k\int_0^1 \partial_x w\partial_x u dx = \int_0^1 wf dx + w(1)b_1$$

3.2 Galerkin Statement

From the weak form, write the Galerkin statement. Again, specify all function spaces.

Solution

Function spaces: $S = \left\{u \middle| u \in H^1, u(0) = u_0\right\}$ and $V = \left\{v \middle| v \in H^1, v(0) = 0\right\}$. Let $u^h = v^h + g^h$ where $g^h(0) = u_0$, we have

$$a \int_0^1 w^h \partial_x (v^h + g^h) dx - \lambda \int_0^1 w^h (v^h + g^h) dx + k \int_0^1 \partial_x w^h \partial_x (v^h + g^h) dx \\ = \int_0^1 w^h f dx + w^h(1) b_1 \tag{104}$$

The Galerkin statement: Find $u^h \in S^h \subset S$, s.t. $\forall w^h \in V^h \subset V$, $u^h \in V^h$,

$$a \int_0^1 w^h \partial_x v^h dx - \lambda \int_0^1 w^h v^h dx + k \int_0^1 \partial_x w^h \partial_x v^h dx \\ = \int_0^1 w^h f dx + w^h(1) b_1 - a \int_0^1 w^h \partial_x g^h dx + \lambda \int_0^1 w^h g^h dx - k \int_0^1 \partial_x w^h \partial_x g^h dx$$

the weak form is satisfied.

3.3 Finite Element Discretization

Introduce the finite element basis and arrive at a linear algebra problem. Clearly define the form of all resulting matrices. You may write everything in terms of the basis functions (i.e. there is no need to directly compute the entries of the matrices).

Solution

Since $w^h, v^h \in V^h$, let

$$w^h = \sum_{A=1}^{n} w_A N_A(x) \tag{105}$$

$$v^h = \sum_{B=1}^{n} u_B N_B(x) \tag{106}$$

$$g^h = u_0 N_{n+1}(x) \tag{107}$$

$$u^h = v^h + g^h = \sum_{B=1}^{n} u_B N_B(x) + u_0 N_{n+1}(x) \tag{108}$$

where $N_A(0) = 0, N_B(0) = 0, N_{n+1}(0) = 1$. Substitute Eq.(105-108) into the Galerkin statement, we have

$$a \int_0^1 \left(\sum_{A=1}^{n} w_A N_A(x)\right) \partial_x \left(\sum_{B=1}^{n} u_B N_B(x)\right) dx - \lambda \int_0^1 \left(\sum_{A=1}^{n} w_A N_A(x)\right) \left(\sum_{B=1}^{n} u_B N_B(x)\right) dx \\ + k \int_0^1 \partial_x \left(\sum_{A=1}^{n} w_A N_A(x)\right) \partial_x \left(\sum_{B=1}^{n} u_B N_B(x)\right) dx$$

$$= \int_0^1 \left(\sum_{A=1}^n w_A N_A(x)\right) f dx + \left(\sum_{A=1}^n w_A N_A(1)\right) b_1 - a \int_0^1 \left(\sum_{A=1}^n w_A N_A(x)\right) \partial_x \left(u_0 N_{n+1}(x)\right) dx$$
$$+\lambda \int_0^1 \left(\sum_{A=1}^n w_A N_A(x)\right) \left(u_0 N_{n+1}(x)\right) dx - k \int_0^1 \partial_x \left(\sum_{A=1}^n w_A N_A(x)\right) \partial_x \left(u_0 N_{n+1}(x)\right) dx \tag{109}$$

Denote the basis functions $N_A(x)$ and $N_B(x)$ to be nonnegative, by the Fubini Theorem, we can simplify Eq.(109) into

$$\sum_{A=1}^n w_A \left(a \int_0^1 N_A(x) \partial_x \left(\sum_{B=1}^n u_B N_B(x)\right) dx - \lambda \int_0^1 N_A(x) \left(\sum_{B=1}^n u_B N_B(x)\right) dx\right.$$
$$+ k \int_0^1 \partial_x \left(N_A(x)\right) \partial_x \left(\sum_{B=1}^n u_B N_B(x)\right) dx - \int_0^1 f N_A dx - N_A(1) b_1$$
$$+ a \int_0^1 N_A(x) \partial_x \left(u_0 N_{n+1}(x)\right) dx - \lambda u_0 \int_0^1 N_A(x) N_{n+1}(x) dx$$
$$\left. + k u_0 \int_0^1 \partial_x (N_A(x)) \partial_x (N_{n+1}(x)) dx \right) = 0 \tag{110}$$

To satisfy Eq.(110), the terms in the bracket must be zero:

$$a \int_0^1 N_A(x) \partial_x \left(\sum_{B=1}^n u_B N_B(x)\right) dx - \lambda \int_0^1 N_A(x) \left(\sum_{B=1}^n u_B N_B(x)\right) dx$$
$$+ k \int_0^1 \partial_x \left(N_A(x)\right) \partial_x \left(\sum_{B=1}^n u_B N_B(x)\right) dx - \int_0^1 f N_A dx - N_A(1) b_1$$
$$+ a \int_0^1 N_A(x) \partial_x \left(u_0 N_{n+1}(x)\right) dx - \lambda u_0 \int_0^1 N_A(x) N_{n+1}(x) dx$$
$$+ k u_0 \int_0^1 \partial_x (N_A(x)) \partial_x (N_{n+1}(x)) dx = 0 \tag{111}$$

By applying the Fubini Theorem again, we have

$$\sum_{B=1}^n a \int_0^1 N_A(x) \partial_x \left(u_B N_B(x)\right) dx - \sum_{B=1}^n \lambda \int_0^1 N_A(x) \left(u_B N_B(x)\right) dx$$
$$+ \sum_{B=1}^n k \int_0^1 \partial_x \left(N_A(x)\right) \partial_x \left(u_B N_B(x)\right) dx$$
$$= \int_0^1 f N_A dx + N_A(1) b_1 - a \int_0^1 N_A(x) \partial_x \left(u_0 N_{n+1}(x)\right) dx$$
$$+ \lambda u_0 \int_0^1 N_A(x) N_{n+1}(x) - k u_0 \int_0^1 \partial_x (N_A(x)) \partial_x (N_{n+1}(x)) dx \tag{112}$$

Denote the series and integrals in Eq.(112) as

$$\sum_{B=1}^n A_{AB} u_B = \sum_{B=1}^n a \int_0^1 N_A(x) \partial_x \left(u_B N_B(x)\right) dx \tag{113}$$

$$\sum_{B=1}^{n} \Lambda_{AB} u_B = \sum_{B=1}^{n} \lambda \int_0^1 N_A(x) \left(u_B N_B(x)\right) dx \tag{114}$$

$$\sum_{B=1}^{n} K_{AB} u_B = \sum_{B=1}^{n} k \int_0^1 \partial_x \left(N_A(x)\right) \partial_x \left(u_B N_B(x)\right) dx \tag{115}$$

$$\begin{aligned} F_A = &\int_0^1 f N_A dx + N_A(1) b_1 - a \int_0^1 N_A(x) \partial_x \left(u_0 N_{n+1}(x)\right) dx \\ &+ \lambda u_0 \int_0^1 N_A(x) N_{n+1}(x) - k u_0 \int_0^1 \partial_x (N_A(x)) \partial_x (N_{n+1}(x)) dx \end{aligned} \tag{116}$$

where

$$A_{AB} = \int_0^1 N_A(x) \partial_x \left(N_B(x)\right) dx \tag{117}$$

$$\Lambda_{AB} = \int_0^1 N_A(x) \left(N_B(x)\right) dx \tag{118}$$

$$K_{AB} = \int_0^1 \partial_x \left(N_A(x)\right) \partial_x \left(N_B(x)\right) dx \tag{119}$$

We reach the linear algebra problem: Solve for

$$\sum_{B=1}^{n} a A_{AB} u_B - \sum_{B=1}^{n} \lambda \Lambda_{AB} u_B + \sum_{B=1}^{n} k K_{AB} u_B = F_A, A = 1, \cdots, n$$

where $A_{AB}, \Lambda_{AB}, K_{AB}, F_A$ are denoted in Eq.(116-119).

4 Problem 4: Implementation

We wish to use the finite element method to solve

$$-\partial_x^2 u = f(x), \qquad x \in (0,1) \tag{120}$$

$$-\partial_x u(0) = \hbar, \quad u(1) = \not{g} \tag{121}$$

for $u(x)$ where $f(x)$ is a known forcing function and $\not{g}$ and $\hbar$ are constant boundary data.

Write a one-dimensional finite element code that uses piecewise linear finite elements to solve for $u(x)$. The following specifications are required:

- The code should work for any constant $\not{g}$ and $\hbar$ as well as arbitrary $f(x)$. These will be inputs to the code.
- Other inputs should include the domain size. You have some flexibility on how to do this. For example, you may pass in a fully-formed mesh if you wish. Alternatively, you can require the user to specify the number of elements from which your code can compute the uniform mesh size.
  - **Note:** You may assume a uniform mesh.
- The code **must** use the local point of view. That is, loop over individual elements and perform the finite element assembly operation to form the global stiffness matrix and force vector.
- Use Gaussian quadrature to perform the integrals. Although not strictly necessary here, it will give you some intuition for how things are actually done.
- You may use an external library to solve the resulting linear system.
- Use must use a compiled language such as `C`, `C++`, or (modern) Fortran.
- Try to submit your job on Odyssey!

Some other things you may want to consider are the following:

- Start simple. A natural progression may be the following:
  - Select $f = 0$, $g = 0$ and $h = 1$ to begin. The exact solution will be $u = 1 - x$.
  - Then make $g = 1$, $h = 1$, $f = 0$. The exact solution will be $u = 2 - x$.
  - Finally, try $g = 1$, $h = 1$, and $f = 1$. The exact solution will be $u = 2 - x + \frac{1}{2}\left(1 - x^2\right)$.
  - At this point, you will have some confidence in your code. This is called code verification. You can try more complicated $f(x)$ if you'd like to.
- Organize your code into separate files. Use a `Makefile` to do the linking.
  - You may want to have files for the following:
    - Quadrature routines
    - Stiffness matrix routines
    - Right-hand-size (RHS) routines
  - Feel free to consider other code designs.

Solution

Here, we will discuss how to build the `C++` program which solves the FEM problem, as well as some implementation details.

To build this program, you must first install the dependency `yaml-cpp`. On our system, we put it in `/SoftwareLibraries/yaml-cpp`. We built it and then dropped a link pointing to `libyaml-cpp.a` into the directory `/SoftwareLibraries/lib`. Note that this link must be a hard link to work on CentOS 7 / Odyssey! A soft link works fine on Ubuntu but fails on CentOS.

The makefile always needs one environment variables on a default Linux system, including Odyssey: `INCLUDE_USR` points to `/SoftwareLibraries` (or wherever you have it on your system). This is used to derive both the include and library directories.

In addition, you can optionally provide a directory `BOOST_DIR`. This is not necessary on Odyssey if you first load the module for `Boost 1_69_0`.

To build on Odyssey, first load the modules for gcc 8.2; `LAPACK / BLAS`; and `Boost 1_69_0`. Clone our repo into e.g. hw1 or Poisson and then run `make`. It should build cleanly if the dependencies are all there.

One instance of the Poisson Problem is described by a `YAML` configuration file. Please see the example files `example_1.yml` through `example_4.yml`. The first 3 are the examples suggested in the assignment. The fourth one is a more interesting case with $u(x) = \sin(x)$, $u''(x) = -\sin(x)$ and $f(x) = \sin(x)$. Appropriate boundary conditions $h = -\cos(0) = -1$ and $g = \sin(1) = 0.84147098$ were also added. There is a choice to define the force vector as either a hard coded vector `(forceType = "VECTOR")` and a function `(forceType = "FUNCTION")`. The first three examples originally used vectors then switched to functions; this makes it easier to increase n. Example 4 only uses a function because it's too tedious to type in a bunch of trig function values.

`PoissonExamples` (`hpp` and `cpp`) defines functions for the force $F(x)$ and solution $U(x)$ for these four examples. These support both input of the force vector via a function as well as automated testing. We didn't add the capability to write a function as a string to be compiled dynamically because it would have added too much complexity.

The bulk of the calculations are done in the `PoissonFiniteElement` class. It is constructed using a string that points to a file name of the `YAML` configuration file. All calculations of the finite element method closely follow the treatment in "The Finite Element Method" by Hughes. The calculations of the local (element) K_e and F_e are currently specialized for piecewise linear interpolation. In this case, Gaussian Quadrature reduces to the midpoint method, and simple formulas come out. These are in the Hughes book and were used. The code framework is flexible enough that we could drop in a modular Gaussian Quadrature routine later, but given that we are only using piecewise linear splines it was simpler to start with the explicit formulas.

There are separate methods to build up the K matrix (`assemble_K`) and the F vector that is the right hand side of the equation (`assemble_F`). A third method, `solve()`, solves for U once K and F are available. The current version is simple and doesn't take advantage of the symmetric tridiagonal structure of K. Instead, K is naively stored as a dense n by n matrix. We solve for U using the most generic `LAPACK` routine `DGESV`, which solves a dense system using Gaussian Elimination. If we wanted to solve this problem in earnest for a large data set, by far the biggest improvement we could make would be to specialize the storage of K and to solve using the optimized `LAPACK` function. Micheal's copy of the Users' Guide suggests `SPTSV` is designed for symmetric Hermitian positive definite tridiagonal systems. There is also `SPBSV` for symmetric positive definite banded if we increased the bandwidth, e.g. with a different basis.

The `LAPACK` functions were called using a `C++` wrapper function. This was provided in the examples for Applied Math 225; it was not original. (It's only a few lines of simple code). One important detail is that while some data in the `PoissonProblem` class was stored as a `C++` vector (e.g. the mesh grid), all matrices and vectors that need to interface with `LAPACK` were stored as arrays of doubles. Large arrays were created with operator new and deleted in the destructor to avoid memory leaks. Arrays were stored in column major order to match Fortran conventions. (It happens not to matter here because K is symmetric, but in general this is a crucial detail.) The wrappers to `LAPACK` functions are in `LinearSolve` (`hpp` and `cpp`).

The main program is called `poisson.cpp`. It runs tests on all four examples, and indicates the results. The 3 simple tests were run with $n = 8$ to allow the vectors to be entered in explicitly. The more interesting test with $u(x) = \sin(x)$ was run with $n = 128$, which gave excellent results (error of than 8E-7). It is also possible to run a specific test, rather than all of them sequentially. To do so, you need to specify two command-line arguments: the first one being the path to the YAML config file, and the second being the name of the basis function used to compare the results. The names of the said functions can be found in `PoissonExamples.cpp` (see 'makeFuncTable').

The `Makefile` conforms to the example shown in class. Some care was taken to add line breaks and tabs so the resulting commands are legible on the screen. The `Makefile` was originally prototyped as an "idiot proof" manual file, `manual.mk`. This allowed us to isolate variables when trouble shooting. Once the manual version worked for the first three files, it was automated into `config.mk`, `Makefile.dep` and `Makefile`. We've left the original because it makes it easier to see what's going on and is a good backup if something breaks.

All functions were commented with tags compatible with `Doxygen`. Documentation was automatically generated. The configuration file is in the repository, but the latex and html working directories were excluded to avoid clutter. You can quickly generate documentation on your local repository by running `doxygen poisson.config`.