Student: Eyan Documet
Due: 2/28/2025

Project 2

1. (8 points) Solve the following boundary value problem analytically.

$$\frac{d}{dx}\left(E\frac{du}{dx}\right) = k^2 \sin\left(\frac{\pi kx}{L}\right) + k\cos\left(\frac{2\pi kx}{L}\right)$$
$$\Omega = (x_0, L) \quad \text{where} \quad x_0 = 0.1, \quad L = 1.2$$
$$E = 0.2$$
$$k = 6$$
$$E\frac{du}{dx}\bigg|_{x=L} = -0.7$$
$$u(x_0) = 1$$

Just as in the previous project, this can be solved as a **seperable ODE.** First, we'll seperate the variables and integrate:

$$\int d\left(E\frac{du}{dx}\right) = \int \left(k^2 \sin\left(\frac{\pi kx}{L}\right) + k\cos\left(\frac{2\pi kx}{L}\right)\right) dx.$$

$$E\frac{du}{dx} = -\frac{kL}{\pi}\cos\left(\frac{\pi kx}{L}\right) + \frac{L}{2\pi}\sin\left(\frac{2\pi kx}{L}\right) + C_1.$$

Then, we integrate again...

$$\int E du = \int \Big(-\frac{kL}{\pi}\cos\left(\frac{\pi kx}{L}\right) + \frac{L}{2\pi}\sin\left(\frac{2\pi kx}{L}\right) + C_1\Big) dx$$

$$Eu(x) = -\frac{L^2}{\pi^2}\sin\left(\frac{\pi kx}{L}\right) - \frac{L^2}{4\pi^2 k}\cos\left(\frac{2\pi kx}{L}\right) + C_1 x + C_2$$

This is our general form of $u(x)$. Now, we can solve our boundary conditions to find C_1 and C_2. First, our Neumann condition:

$$E\frac{du}{dx}\bigg|_{x=L} = -0.7 = -\frac{kL}{\pi}\cos(\pi k) + \frac{L}{2\pi}\sin(2\pi k) + C_1$$

$$C_1 = -0.7 + \frac{kL}{\pi}\cos(\pi k) - \frac{L}{2\pi}\sin(2\pi k)$$

And our Dirchelet:

$$Eu(x_0) = 0.2 = -\frac{L^2}{\pi^2}\sin\left(\frac{\pi k x_0}{L}\right) - \frac{L^2}{4\pi^2 k}\cos\left(\frac{2\pi k x_0}{L}\right) + C_1 x_0 + C_2$$

$$C_2 = 0.2 + \frac{L^2}{\pi^2}\sin\left(\frac{\pi k x_0}{L}\right) + \frac{L^2}{4\pi^2 k}\cos\left(\frac{2\pi k x_0}{L}\right) - C_1 x_0$$

Combining all together, we can obtain our $u(x)$! In summary:

$$\boxed{Eu(x) = -\frac{L^2}{\pi^2}\sin\left(\frac{\pi k x}{L}\right) - \frac{L^2}{4\pi^2 k}\cos\left(\frac{2\pi k x}{L}\right) + C_1 x + C_2}$$

Where:

$$\boxed{C_1 = -0.7 + \frac{kL}{\pi}\cos(\pi k) - \frac{L}{2\pi}\sin(2\pi k)}$$

$$\boxed{C_2 = 0.2 + \frac{L^2}{\pi^2}\sin\left(\frac{\pi k x_0}{L}\right) + \frac{L^2}{4\pi^2 k}\cos\left(\frac{2\pi k x_0}{L}\right) - C_1 x_0}$$

2. (2 points) Rewrite the weak form from Project 1, Question 2 accounting for the fact that you now have a Neumann boundary condition.

In the previous project, we assumed the weak form to be:

$$\int_\Omega \frac{d\nu}{dx} E \frac{du}{dx} dx = \int_\Omega \sigma \frac{d\nu}{dx} dx = \int_\Omega f\nu dx$$

This was fine, because we were only dealing with Dirichlet conditions. Thus, it was safe to drop the term $t^*\nu|_{\Gamma_t}$ from the equation since both boundary condition prescribed a fixed displacement, making the traction term irrelevant.

However, since we're now working with both Neumann and Dirichlet conditions, we cannot exclude this term. Therefore, our new weak form is:

$$\boxed{\int_\Omega \sigma \frac{d\nu}{dx} dx = \int_\Omega f\nu dx + t^*\nu|_{\Gamma_t}}$$

The rest of the assumptions carry over from previous work.

3. Now write a 1D finite element method program using linear equal-sized elements to solve Part 1's BVP. Your numerical solution, uN , must satisfy the following error tolerance:

   (a) (20 points) How many finite elements (N_e) are needed to meet the error tolerance when using linear ($p = 1$), quadratic ($p = 2$), or cubic ($p = 3$) shape functions?

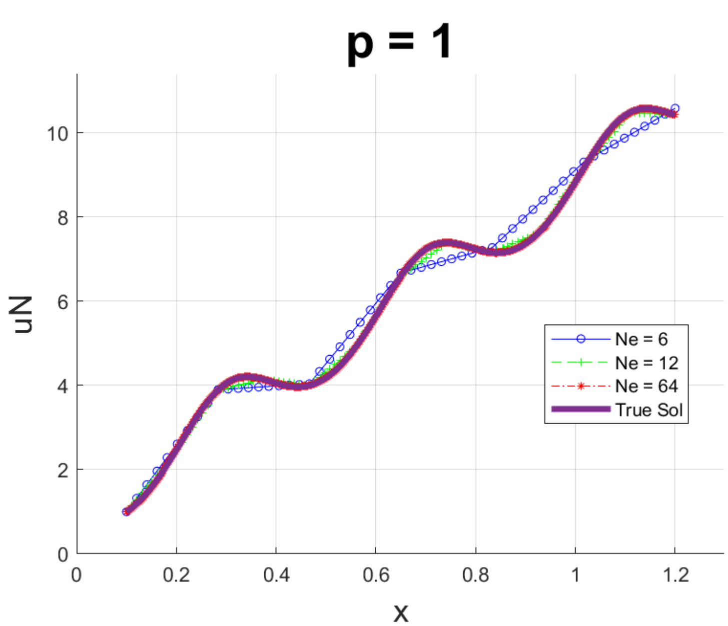
   After running my new FEM code, I found the following minimum elements were required with p-refinement:

p	N_e	**error**
1	70	0.049725
2	13	0.047314
3	7	0.037543

   Table 1: Polynomial order vs. # Elements vs. Error

   Clearly, p-refinement has a remarkable impact on the number of elements required to solve FME problems, as increasing p from 1 to 3 represented a x10 reduction in the number of elements required.

   (b) 35 points) For each p, plot the numerical solutions for N e = 6,12,64 and the true solution on the same plot.



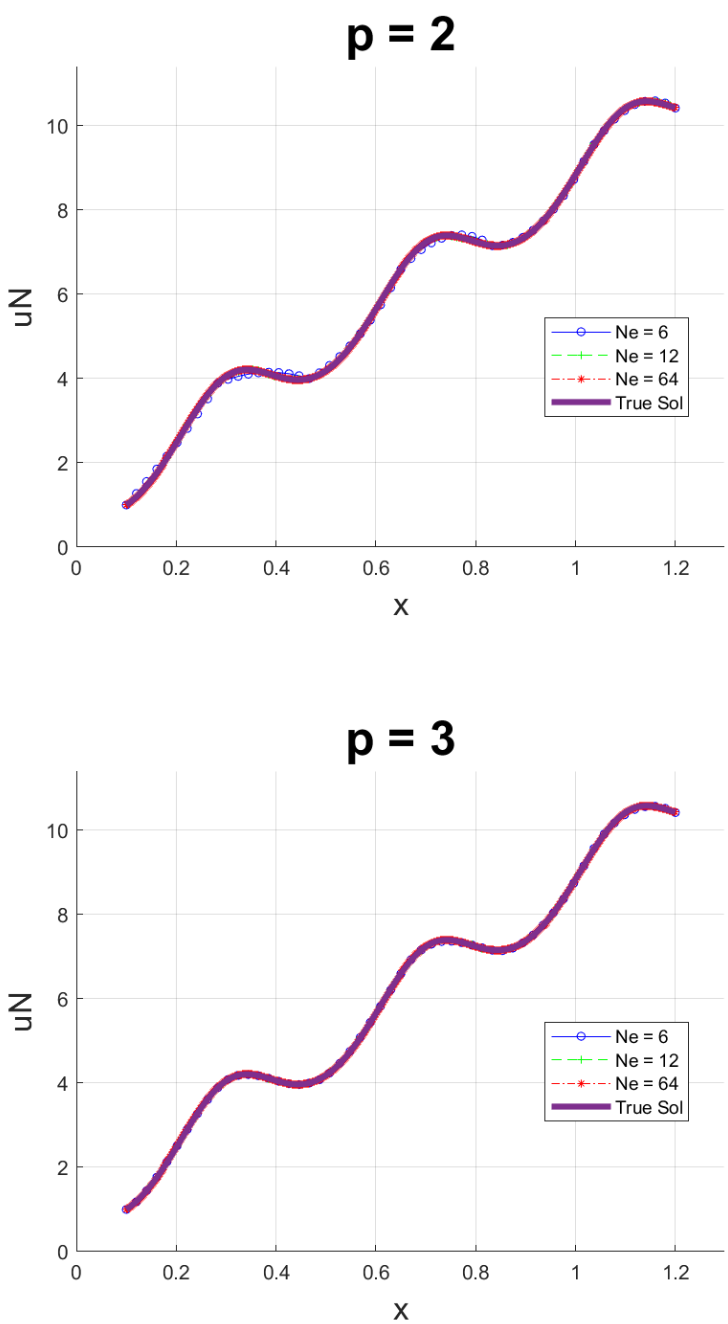


Figure 1: My FEM Solutions with p-refinement.

As remarked earlier, p-refinement converged on u_{true} in significantly fewer iterations compared to Project 1. The higher polynomial degree of the shape functions led to a rapid reduction in the approximation error, even with fewer elements.

Furthermore, re-sampling had a huge impact on the visual fidelity of my output curves. By increasing the number of sample points used for plotting and evaluating the solution (as opposed to just the initial nodes), I was able to far better capture the fine details of the numerical solution curve.

As both p and N_e increase, the visual fidelity of the output curves improve.

(c) Plot e^N vs $1/N_e$ for each p in log-log scaling. What trends (specifically about the slope) do you observe for each p?

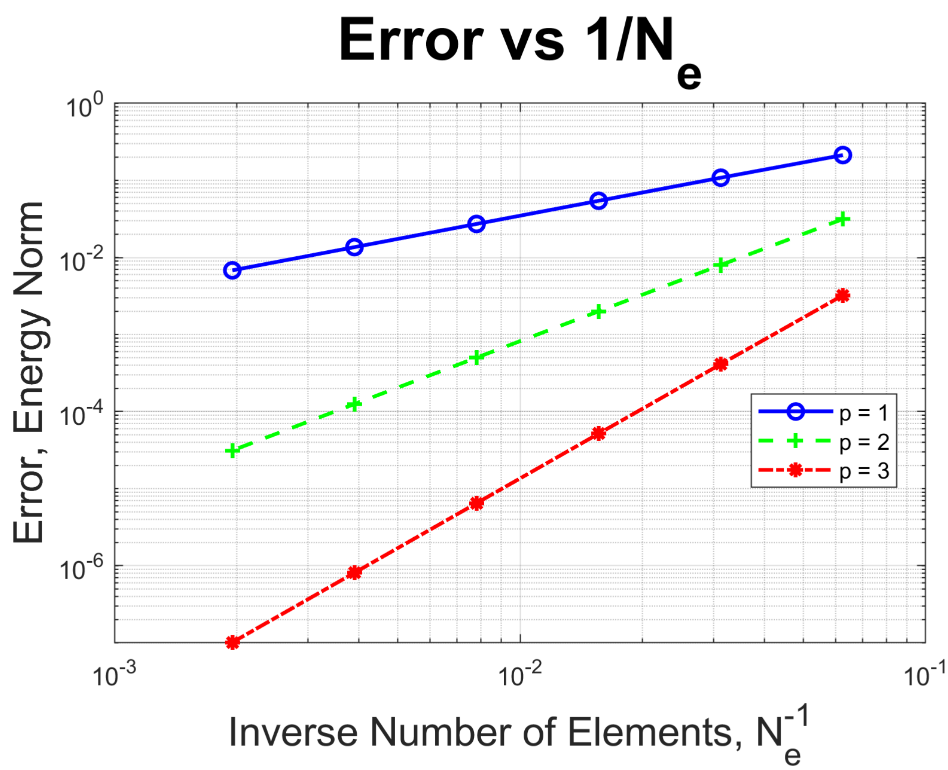


Figure 2: e^N vs $1/N_e$ for various polynomial orders.

As we increase the polynomial order of the elements, the slope of convergence increases. In other words, as p increases, our results converge to the true solution at a much faster rate per added element. Since this is plotted on a log-log scale, this implies a power-law relationship, meaning the convergence accelerates significantly as p grows.

Furthermore, higher-order elements also seemed to experience less error, even with the same amount of elements.

This interplay between h-refinement and p-refinement suggests the existence of a "sweet spot," where the balance between element count and polynomial degree yields the most efficient convergence.

4. (10 points) Based on your results, which refinement method, h-refinement or p-refinement, is more effective in producing the most accurate solutions for less computational cost? What characteristics of the given differential equation causes one to be more effective than the other?

From what I observed, p-refinement seems to be more effective in producing more accurate solutions for less computational cost.

In these experiments, we needed significantly fewer elements (and in turn, relatively fewer nodes) to capture details very quickly. For example, in Problem 3a, we were able to achieve tolerance with 7 third-order elements (22 total nodes) vs 70 first-order elements (141 total nodes).

While there are naturally situations where h-refinement will be more apt (notably "rough" solutions with sharp gradients and/or discontinuities, where polynomial order wont help as much), p-refinement seems to converge on acceptable results significantly faster.