Dan Card
Aerospace 523: Computational Fluid Dynamics
Homework: 2

dcard@umich.edu

Due: September 25th, 2020

1 Lagrange Interpolation

Use the method of Lagrange *interpolation* to derive an expression for the mixed derivative, u_{xy}, at the center of the stencil pictured below, i.e. point (0,0), using the values of u only at the four corner nodes in a uniform grid. Note, this problem is solved using the method of undetermined coefficients in the notes.

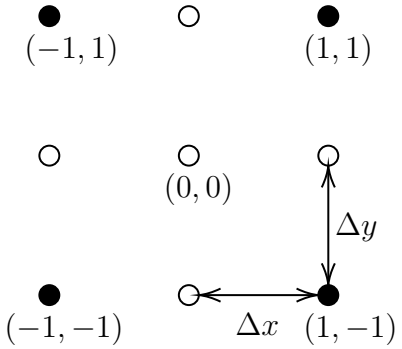

Figure 1: Stencil for Lagrange interpolation.

To conduct the Lagrange interpolation, I will conduct evaluate at each node. Taking into consideration that this is in two-dimensions that the Lagrange interpolating functions can be expressed separetly to be

$$L^z_{-1}(z) = \frac{(z-z_0)(z-z_1)}{(-\Delta z)(-2\Delta z)}, \quad \text{for node } j=-1$$

$$L^z_{0}(z) = \frac{(z-z_{-1})(z-z_1)}{(\Delta z)(-\Delta z)}, \quad \text{for node } j=-0$$

$$L^z_{1}(z) = \frac{(z-z_{-1})(z-z_0)}{(2\Delta z)(\Delta z)}, \quad \text{for node } j=1$$

Where z is either $x,\ y$ depending on the node and evaluating at each of the nodes gives,

$$\begin{aligned}
N_{1,1} &= L_1(x)\ L_1(y) \\
&= \left(\frac{(x-x_{-1})(x-x_0)}{(2\Delta x)(\Delta x)}\right)\left(\frac{(y-y_{-1})(y-y_0)}{(2\Delta y)(\Delta y)}\right) \\
N_{-1,-1} &= L_{-1}(x)\ L_{-1}(y) \\
&= \left(\frac{(x-x_0)(x-x_{-1})}{(-\Delta x)(-2\Delta x)}\right)\left(\frac{(y-y_0)(y-y_{-1})}{(-\Delta y)(-2\Delta y)}\right) \\
N_{1,-1} &= L_1(x)\ L_{-1}(y) \\
&= \left(\frac{(x-x_{-1})(x-x_0)}{(2\Delta x)(\Delta x)}\right)\left(\frac{(y-y_0)(y-y_{-1})}{(-\Delta y)(-2\Delta y)}\right) \\
N_{-1,1} &= L_{-1}(x)\ L_1(y) \\
&= \left(\frac{(x-x_0)(x-x_{-1})}{(-\Delta x)(-2\Delta x)}\right)\left(\frac{(y-y_{-1})(y-y_0)}{(2\Delta y)(\Delta y)}\right)
\end{aligned}$$

With all the Lagrange interpolating functions defined at each node the approximated solution can be expressed by the following relation,

$$\tilde{u} = \sum_{j=-l}^{r} \sum_{k=-d}^{u} L_j^x(x) L_k^y(y) u_{k,j}$$

In order to evaluate the mixed derivative at the center point, each Nodal value must have its mixed derivative computed and evalued at $x_0,\ y_0$ so,

$$\begin{aligned}
\frac{\partial^2 N_{1,1}}{\partial x \partial y}|_{x_0,\ y_0} &= \frac{(x_0 - 2x + x_{-1})(y_0 - 2y + y_{-1})}{4\Delta x^2 \Delta y^2}|_{x_0,\ y_0} \\
&= \frac{\overbrace{(-x_0 + x_{-1})}^{-\Delta x}\overbrace{(-y_0 + y_{-1})}^{-\Delta y}}{4\Delta x^2 \Delta y^2} u_{1,1} = \frac{1}{4\Delta x \Delta y} u_{1,1} \\
\frac{\partial^2 N_{-1,-1}}{\partial x \partial y}|_{x_0,\ y_0} &= \frac{(x_0 - 2x + x_1)(y_0 - 2y + y_1)}{4\Delta x^2 \Delta y^2}|_{x_0,\ y_0} \\
&= \frac{\overbrace{(-x_0 + x_1)}^{\Delta x}\overbrace{(-y_0 + y_1)}^{\Delta y}}{4\Delta x^2 \Delta y^2} u_{-1,-1} = \frac{1}{4\Delta x \Delta y} u_{-1,-1} \\
\frac{\partial^2 N_{1,-1}}{\partial x \partial y}|_{x_0,\ y_0} &= \frac{(x_0 - 2x + x_{-1})(y_0 - 2y + y_1)}{4\Delta x^2 \Delta y^2}|_{x_0,\ y_0} \\
&= \frac{\overbrace{(-x_0 + x_{-1})}^{-\Delta x}\overbrace{(-y_0 + y_1)}^{\Delta y}}{4\Delta x^2 \Delta y^2} u_{1,-1} = -\frac{1}{4\Delta x \Delta y} u_{1,-1} \\
\frac{\partial^2 N_{-1,1}}{\partial x \partial y}|_{x_0,\ y_0} &= \frac{(x_0 - 2x + x_1)(y_0 - 2y + y_{-1})}{4\Delta x^2 \Delta y^2}|_{x_0,\ y_0} \\
&= \frac{\overbrace{(-x_0 + x_1)}^{\Delta x}\overbrace{(-y_0 + y_{-1})}^{-\Delta y}}{4\Delta x^2 \Delta y^2} u_{-1,1} = -\frac{1}{4\Delta x \Delta y} u_{-1,1}
\end{aligned}$$

Summing all the contributions and substituting back into the equation we get that,

$$\boxed{u_{xy} \approx \frac{1}{4\Delta x \Delta y}\left(u_{1,1} + u_{-1,-1} - (u_{1,-1} + u_{-1,1})\right)}$$

2 Truncation Error Analysis

Use the method of undetermined coefficients to derive a finite-difference approximation to the second derivative of u at point 0, $u_{xx}|_0$, using u values at points 0,1,2 on a uniform grid. Repeat this with the addition of u at point 3. In both cases, determine the order of accuracy of your formula.

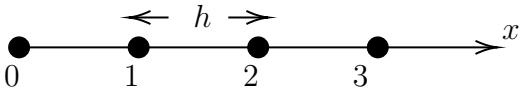

Figure 2: Undetermined coefficients on a uniform grid.

Firstly start with the equation for the method of undetermined coefficients,

$$\frac{d^2u}{dx^2} = \sum_{j=0}^{2} = a_2u_2 + a_1u_1 + a_0u_0$$

Conducting Taylor-Expansions for $u_2,\ u_1,\ u_0$ gives,

$$= a_2 \underbrace{\left(u_0 + (2h)\frac{du}{dx} + \frac{1}{2}(2h)^2\frac{d^2u}{dx^2} + \frac{1}{6}(2h)^3\frac{d^3}{dx^3} + \dots\right)}_{\text{Taylor-Expansion about } u_2}$$

$$+ a_1 \underbrace{\left(u_0 + h\frac{du}{dx} + \frac{1}{2}h^2\frac{d^2u}{dx^2} + \frac{1}{6}h^3\frac{d^3u}{dx^3} + \dots\right)}_{\text{Taylor-Expansion about } u_1}$$

$$+ a_0u_0$$

Grouping all the terms gives,

$$= (a_2 + a_1 + a_0)\, u_0 + (2a_2 + a_1)\, h\frac{du}{dx}$$
$$+ \left(2a_2 + \frac{1}{2}a_1\right) h^2\frac{d^2u}{dx^2} + \left(\frac{4}{3}a_2 + \frac{1}{6}a_1\right) h^3\frac{d^3u}{dx^3} + \dots \mathcal{O}(h^4)$$

With three unknowns and three equations to solve for $a_0,\ a_1,\ a_2$ writing out in matrix form is,

$$\begin{bmatrix} 0 \\ 0 \\ \frac{1}{h^2} \end{bmatrix} = \begin{bmatrix} a_2 + a_1 + a_0 \\ 2a_1 + a_1 \\ 2a_2 + \frac{1}{2}a_1 \end{bmatrix}$$

Solving for $a_0,\ a_1,\ a_2$ then becomes,

$$\boxed{a_0 = \frac{1}{h^2}, \quad a_1 = -\frac{2}{h^2}, \quad a_2 = \frac{1}{h^2}}$$

Expanding the expression further gives that to the next higher term $\frac{d^3}{dx^3}$ gives,

$$0 + \frac{1}{6}a_1 + \frac{4}{3}a_2 = h^3\frac{d^3}{dx^3}$$

Substituting in the a values from above,

$$-\frac{1}{3h^3} + \frac{4}{3h^2} = h^3\frac{d^3}{dx^3}, \quad 1 = h\frac{d^3u}{dx^3}$$
$$\frac{d^2u}{dx^2} = \frac{u_2 - 2u_1 + u_0}{h^2} + h\frac{d^3u}{dx^3}$$

Shown above this is first-order accurate since the error is $\mathcal{O}(\mathbf{h})$.

Repeating with the addition of u at point 3 gives,

$$\frac{d^2u}{dx^2} = \sum_{j=0}^{3} = a_3u_3 + a_2u_2 + a_1u_1 + a_0u_0$$

Conducting Taylor-Expansions for $u_3,\ u_2,\ u_1,\ u_0$ gives,

$$= a_3 \underbrace{\left(u_0 + (3h)\frac{du}{dx} + \frac{1}{2}(3h)^2\frac{d^2u}{dx^2} + \frac{1}{6}(3h)^3\frac{d^3u}{dx^2} + \dots\right)}_{\text{Taylor-Expansion about } u_3}$$

$$+ a_2 \underbrace{\left(u_0 + (2h)\frac{du}{dx} + \frac{1}{2}(2h)^2\frac{d^2u}{dx^2} + \frac{1}{6}(2h)^3\frac{d^3}{dx^3} + \dots\right)}_{\text{Taylor-Expansion about } u_2}$$

$$+ a_1 \underbrace{\left(u_0 + h\frac{du}{dx} + \frac{1}{2}h^2\frac{d^2u}{dx^2} + \frac{1}{6}h^3\frac{d^3u}{dx^3} + \dots\right)}_{\text{Taylor-Expansion about } u_1}$$

$$+ a_0u_0$$

Grouping all the terms gives,

$$= (a_3 + a_2 + a_1 + a_0)\, u_0 + (3a_3 + 2a_2 + a_1)\, h\frac{du}{dx}$$

$$+ \left(\frac{9}{2}a_3 + 2a_2 + \frac{1}{2}a_1\right) h^2\frac{d^2u}{dx^2} + \left(\frac{9}{2}a_3 + \frac{4}{3}a_2 + \frac{1}{6}a_1\right) h^3\frac{d^3u}{dx^3}$$

$$+ \dots \mathcal{O}(h^5)$$

With four unknowns and four equations to solve for $a_0,\ a_1,\ a_2,\ a_3$ writing out in matrix form is,

$$\begin{bmatrix} 0 \\ 0 \\ \frac{1}{h^2} \\ 0 \end{bmatrix} = \begin{bmatrix} a_3 + a_2 + a_1 + a_0 \\ 3a_3 + 2a_2 + a_1 \\ \frac{9}{2}a_3 + 2a_2 + \frac{1}{2}a_1 \\ \frac{9}{2}a_3 + \frac{4}{3}a_2 + \frac{1}{6}a_1 \end{bmatrix}$$

Solving for $a_0,\ a_1,\ a_2,\ a_3$ then becomes,

$$\boxed{a_0 = \frac{2}{h^2}, \quad a_1 = -\frac{5}{h^2}, \quad a_2 = \frac{4}{h^2}, \quad a_3 = -\frac{1}{h^2}}$$

Expanding the expression further gives that to the next higher term $\frac{d^3}{dx^3}$ gives,

$$0 + \frac{1}{24}a_1 + \frac{16}{24}a_2 + \frac{81}{24}a_3 = -\frac{11}{12h^2}\left(h^4\frac{d^4}{dx^4}\right)$$

Substituting in the a values from above,

$$\frac{d^2u}{dx^2} = \frac{2u_0 - 5u_1 + 4u_2 - u_3}{h^2} + h^2\frac{d^4u}{dx^4}$$

Shown above this is second-order accurate since the error is $\mathcal{O}(\mathbf{h^2})$.

3 Domain Mapping

A reference-to-global coordinate mapping $\vec{x}\left(\vec{\xi}\right)$ is given by

$$x = \xi + \frac{1}{2}\eta^2,$$
$$y = 1 + \eta - \frac{1}{4}\xi\eta,$$

where $\vec{\xi} = (\xi, \eta)$ and $\vec{x} = (x, y)$. Consider a unit square domain in reference space, $\vec{\xi} \in [0, 1]^2$.

a. Show what this domain looks like in global space by providing a computer-generated plot of the domain edges.

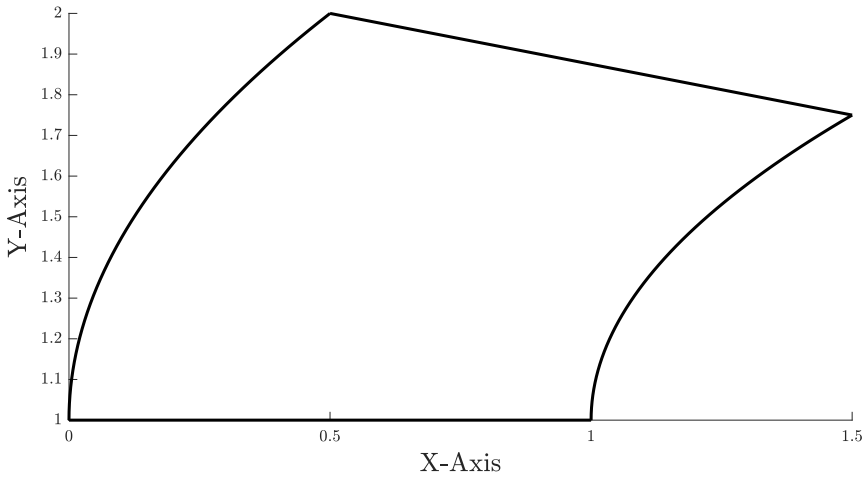

Figure 3: Reference space mapped to global space.

Mapping the reference space to the global space results in Figure 3 shown above.

b. Determine an analytical expression for the 2×2 mapping Jacobian matrix, $\mathbf{J} \equiv \frac{\partial \vec{x}}{\partial \vec{\xi}}$, and its determinant, $\mathbf{J} \equiv \det(\mathbf{J})$.

The Jacobian matrix can be written as,

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \xi}\left(\xi + \frac{1}{2}\eta^2\right) & \frac{\partial}{\partial \eta}\left(\xi + \frac{1}{2}\eta^2\right) \\ \frac{\partial}{\partial \xi}\left(1 + \eta - \frac{1}{4}\xi\eta\right) & \frac{\partial}{\partial \eta}\left(1 + \eta - \frac{1}{4}\xi\eta\right) \end{bmatrix}$$

Then taking the derivatives give,

$$\underline{\underline{J}} = \begin{bmatrix} 1 & \eta \\ -\frac{1}{4}\eta & 1 - \frac{1}{4}\xi \end{bmatrix}$$

Then taking the determinant gives,

$$\det(J) = 1 \cdot \left(1 - \frac{1}{4}\xi\right) - \eta\left(-\frac{1}{4}\eta\right)$$

This then results in the following determinant result,

$$\det(J) = 1 + \frac{1}{4}(\eta^2 - \xi)$$

c. Calculate the normal vectors at the four domain edge midpoints (where the midpoint is identified in reference space).

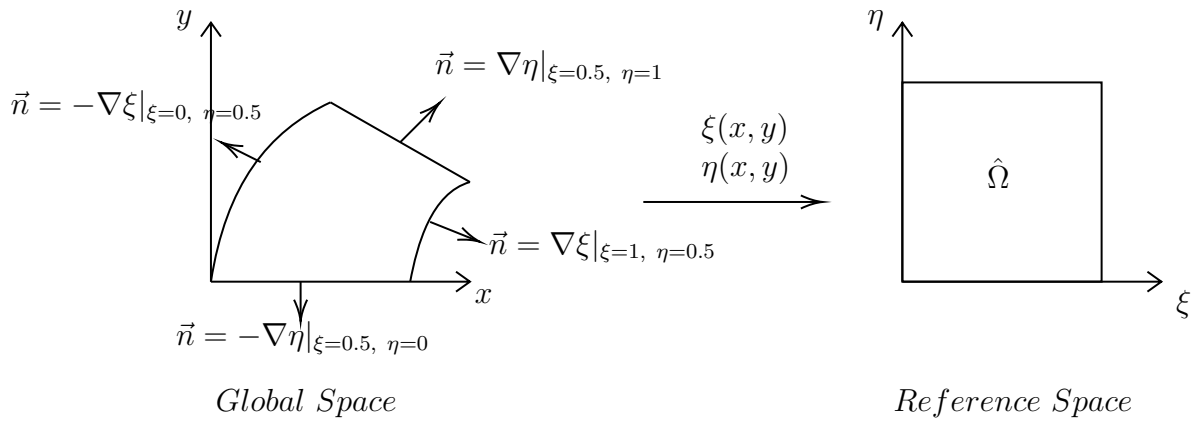

Figure 4: Global to Reference space transformation.

In order to compute the normal vectors in the global space with reference to the mid-points in the reference space, I will solve for the expressions for ξ and η. Conducting this in Matlab and simplifying gives that ξ and η are,

$$\xi = x - \frac{\left(\frac{\frac{2x}{3} - \frac{8}{3}}{\left(4y + \sqrt{(4y-4)^2 - \left(\frac{2x}{3} - \frac{8}{3}\right)^3} - 4\right)^{1/3}} + \left(4y + \sqrt{(4y-4)^2 - \left(\frac{2x}{3} - \frac{8}{3}\right)^3} - 4\right)^{1/3} \right)^2}{2}$$

$$\eta = \frac{\frac{2x}{3} - \frac{8}{3}}{\left(4y + \sqrt{(4y-4)^2 - \left(\frac{2x}{3} - \frac{8}{3}\right)^3} - 4\right)^{1/3}} + \left(4y + \sqrt{(4y-4)^2 - \left(\frac{2x}{3} - \frac{8}{3}\right)^3} - 4\right)^{1/3}$$

With ξ and η having been solved for, the normal vectors can be solved for by taking the gradient at the mid-points of the references as shown in Figure 4. With the expressions for ξ and η in terms of x and y so taking the gradient at each respective reference midpoint through Matlab gives that the normal vectors are,

$$n_1 = [0, -1]$$
$$n_2 = [0.83205, -0.5547]$$
$$n_3 = [0.24254, 0.97014]$$
$$n_4 = [-0.89443, 0.44721]$$

d. Calculate the area of the domain in global space by an analytical integration over the reference unit square.

Integrating over the global domain by using the transformation from the reference domain. The integral of some function $f(\vec{x})$ transforms as,

$$\int_{\Omega} f(\vec{x})\ dxdy = \int_{\hat{\Omega}} f\left(\vec{x}\left(\vec{\xi}\right)\right) \det\left(J\right)\ d\xi d\eta$$

Since the area in global space can be represented in the reference domain, then this integration can be simplified significantly. In other words this can be an integration above the top of the reference unit-square where ξ can vary and $\eta = 1$. This gives that the function $f\left(\vec{x}\left(\vec{\xi}\right)\right) = 1$

Re-writing and substituting in the determinant of the Jacobian gives,

$$A_{\Omega} = \int_{\partial\hat{\Omega}} \underbrace{1 + \frac{1}{4}(\eta^2 - \xi)}_{\det\left(J\right)}\ d\xi d\eta$$

Where the domains are $\xi \in [0, 1]$ and $\eta \in [0, 1]$, simplifying gives,

$$A_{\Omega} = \int_{\eta=0}^{\eta=1} \int_{\xi=0}^{\xi=1} 1 + \frac{1}{4}(\eta^2 - \xi)\ d\xi d\eta$$

Conducting the integral gives,

$$\begin{aligned} A_{\Omega} &= \int_{\eta=0}^{\eta=1} \left(\xi + \frac{1}{4}\eta^2\xi - \frac{1}{8}\xi^2\right) |_{\xi=0}^{\xi=1}\ d\eta \\ &= \int_{\eta=0}^{\eta=1} 1 + \frac{1}{4}\eta^2 - \frac{1}{8}\ d\eta \\ &= \int_{\eta=0}^{\eta=1} \frac{7}{8} + \frac{1}{4}\eta^2\ d\eta \\ &= \frac{7}{8}\eta + \frac{1}{12}\eta^3 |_{\eta=0}^{\eta=1} \\ &= \frac{7}{8} + \frac{1}{12} \end{aligned}$$

This gives that the area in the global space is,

$$\boxed{A_{\Omega} = \frac{23}{24}}$$

4 Area Calculation

In the last problem of homework 1, you identified loops of nodes using edges. You should have found that these loops enclosed a computational domain surrounding a three-element airfoil. Moreover, the edges were constructed such that the computational domain is always on the left as an edge is traversed from the first node to the second. This corresponds to a clockwise ordering for the inner loops, and a counter-clockwise ordering for the outer loop. In this problem, your task is to compute the area of that computational domain. Do this by a numerical (midpoint) integration along the edges, after applying the divergence theorem for a vector function $\vec{F} = x\hat{i}$. Repeat with $\vec{F} = y\hat{j}$ and show that you obtain the same answer.

Firstly we wish to evaluate,

$$A = \oint_{\partial S} \vec{F} \cdot \hat{n} \; dl$$

However, discretizing this over the edges gives,

$$A \approx \sum_{\text{edge}} \left(\vec{F} \cdot \hat{n} \right)_{\text{mid-point}} \cdot \Delta l_{\text{edge}}$$

Conducting this for $\vec{F} = x\hat{i}$ gives,

$$A(\vec{F} = x\hat{i}) = 17020.19$$

Repeating with $\vec{F} = y\hat{j}$ results in,

$$A(\vec{F} = y\hat{j}) = 17020.19$$

As shown above and shown in my attached Matlab code at the end of my assignment, the results are in agreement with each other despite using a different vector function.

5 Advection-Diffusion

Apply the finite-difference method to solve the steady advection-diffusion equation with a source term,

$$u_x - \nu u_{xx} = \sin(\pi x), \qquad u(0) = u(1) = 0.$$

Write a code to solve this problem on a uniform grid of N intervals. Use central difference formulas for both the first and second derivatives. Using N= 16, plot the solution for ν= 0.1 and ν= 1 on one figure. For ν= 0.1, run simulations using $N = 4,\ 8,\ 16,\ 32$, and plot these on a single figure. Comment on the effect of ν and N on the solution. Also state whether and why the solution makes sense physically, based on your understanding of advection-diffusion.

Firstly, re-writing the equation above in finite terms gives,

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$
$$u_{xx} \approx \frac{u_{i+1} - 2u_i + u_{i+1}}{\Delta x^2}$$

Substituting into the given equation gives,

$$\underbrace{\frac{u_{i+1} - u_{i-1}}{2\Delta x}}_{u_x} - \nu \underbrace{\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}}_{u_{xx}} = \sin(\pi x)$$

Simplifying and grouping terms gives,

$$(\Delta x - 2\nu)u_{i+1} + 4\nu u_i - (\Delta x + 2\nu)u_{i-1} = 2\Delta x^2 \sin(\pi x)$$

Implementing these into Python and running the simulation gives,

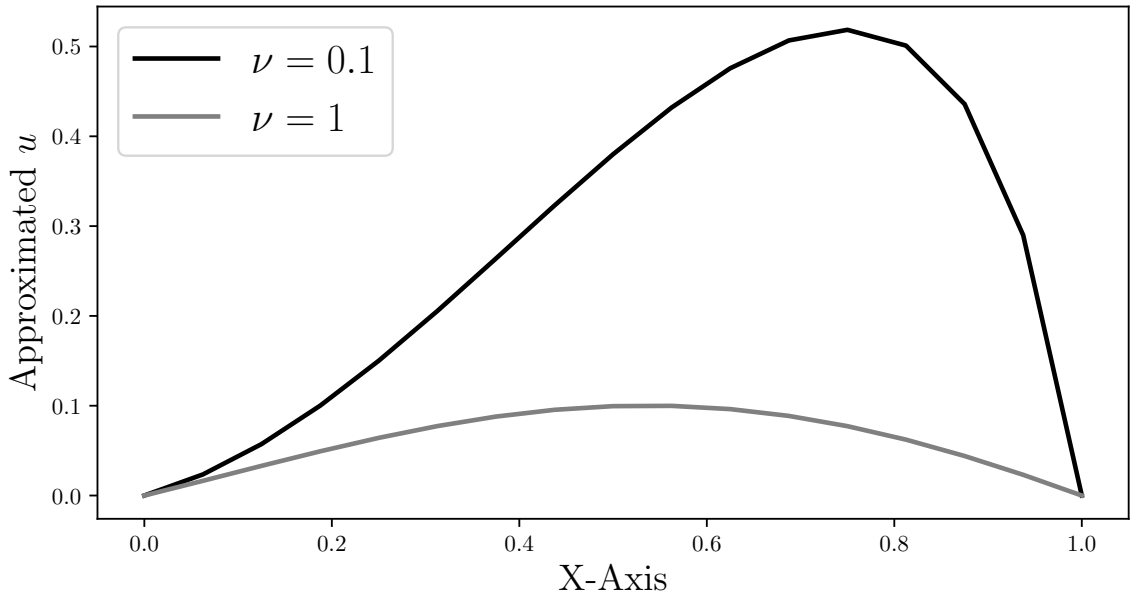

Figure 5: Advection-diffusion with varying ν values.

Doing so again but varying N

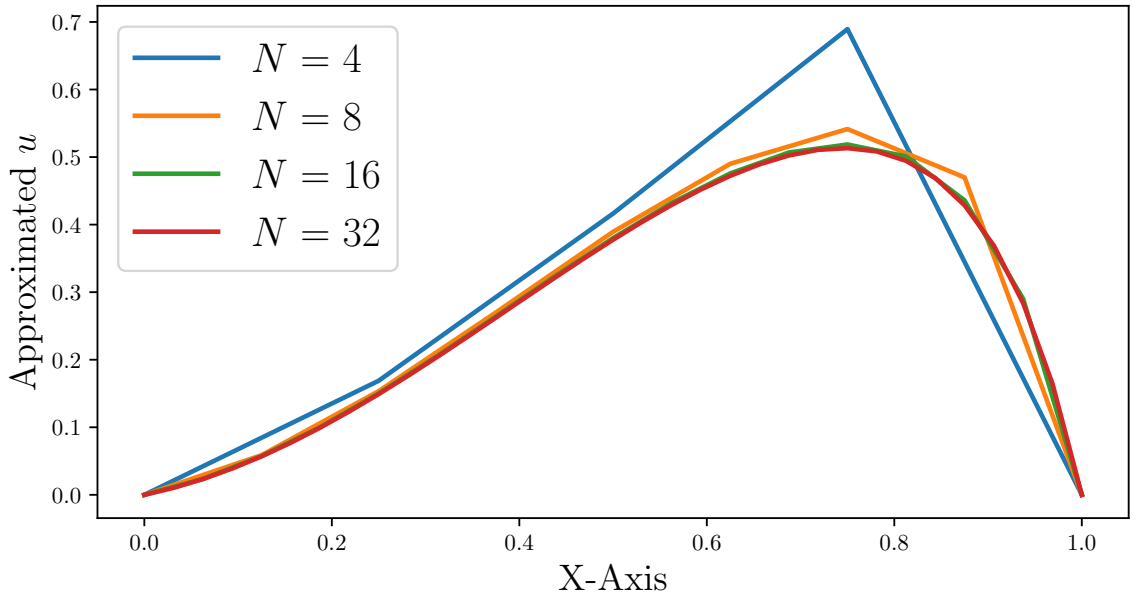

Figure 6: Advection-diffusion with varying N values.

Looking to Figures 5, 6 above we can observe the differences from varying N, and ν. Varying N only affects the refinment of the simulation whereas varying ν varies the physics of the problem. With a higher ν value, the more the approximated solution will match the point-source since the medium is more likely to diffuse to match the point source.

Supporting Code

Algorithm 1: Matlab Code for Homework 2, question 1.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

syms x_0 x_p1 x_m1 y_0 y_p1 y_m1 Delta_x Delta_y x y u_m1p1 u_m1m1 u_00 u_p1p1 u_p1m1

x_nodep1 = ((x - x_m1) * (x - x_0)) / ((2*Delta_x) * (Delta_x));
x_nodem1 = ((x - x_0) * (x - x_p1)) / ( (-Delta_x) * (-2*Delta_x));

y_nodep1 = ((y - y_m1) * (y - y_0)) / ((2*Delta_y) * (Delta_y));
y_nodem1 = ((y - y_0) * (y - y_p1)) / ( (-Delta_y) * (-2*Delta_y));

pretty(simplify(diff(diff(x_nodep1*y_nodep1, x), y))) % xp1yp1
pretty(simplify(diff(diff(x_nodem1*y_nodem1, x), y))) % xm1ym1
pretty(simplify(diff(diff(x_nodep1*y_nodem1, x), y))) % xp1ym1
pretty(simplify(diff(diff(x_nodem1*y_nodep1, x), y))) % xm1yp1
```

Algorithm 2: Matlab Code for Homework 2, question 2.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

syms a_3 a_2 a_1 a_0 h

eqn1 = 0 == a_2 + a_1 + a_0;
eqn2 = 0 == 2*a_2 + a_1;
eqn3 = 1/h^2 == 2*a_2 + 1/2*a_1;
sol = solve([eqn1, eqn2, eqn3],[a_2, a_1, a_0]);

tolatex('p1_a2',sol.a_2)
tolatex('p1_a1',sol.a_1)
tolatex('p1_a0',sol.a_0)

eqn1 = 0 == a_3 + a_2 + a_1 + a_0;
eqn2 = 0 == 3*a_3 + 2*a_2 + a_1;
eqn3 = 1/h^2 == 9/2*a_3 + 2*a_2 + 1/2*a_1;
eqn4 = 0 == 9/2*a_3 + 4/3*a_2 + 1/6*a_1;
sol = solve([eqn1, eqn2, eqn3, eqn4],[a_3, a_2, a_1, a_0]);

tolatex('p2_a3',sol.a_3)
tolatex('p2_a2',sol.a_2)
tolatex('p2_a1',sol.a_1)
tolatex('p2_a0',sol.a_0)
```

Algorithm 3: Matlab Code for Homework 2, question 4.

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
clear all; clc; close all
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% Read in values
E = dlmread('E.txt');
V = dlmread('V.txt');

fxsum = 0; fysum = 0; % Initialize the summations
for i = 1:length(E)
    node1 = V(E(i,1),:);  % Nodal values
    node2 = V(E(i,2),:);  % Nodal values

    lvec = node1 - node2; % Direction of L
    deltal = norm(lvec);  % Calculate the total length of L
    midpoint = (node1 + node2)/2;   % Determine mid-point of L

    nhat = lvec * [0, 1; -1, 0]; % Apply 90degree rotation
    nhat = nhat./norm(nhat);     % Normalize
    fxsum = fxsum + dot([midpoint(1), 0], nhat)*deltal; % Apply divergence
    fysum = fysum + dot([0, midpoint(2)], nhat)*deltal; % Apply divergence
end

% Output to latex
fid = fopen('fx_out','w');
fprintf(fid,'& = %.2f', fxsum);
fclose(fid);

fid = fopen('fy_out','w');
fprintf(fid,'& = %.2f', fysum);
fclose(fid);
```

Algorithm 4: Python Code for Homework 2, question 5.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import sparse
from scipy.sparse import linalg

plt.rc('text', usetex=True)
plt.rc('font', family='serif')

def  advec_diff(N, nu):
    L = 1
    x = np.linspace(0, L, N+1)
    dx = float(L)/N
    data = np.ones((3, N-1))
    data[0,:] *= -(dx + 2*nu); data[1,:] *= 4*nu; data[2,:] *= (dx - 2*nu)
    diags = np.array([-1, 0, 1])

    A = sparse.spdiags(data, diags, N-1, N-1, 'csr')
    b = np.zeros(N+1)

    q = 2*dx**2*np.sin(np.pi * x)
    b = q[1:N]
    b[0] += 0
    b[N-2] += 0

    Tt = linalg.spsolve(A, b)
    T = np.zeros(N+1)
    T[0] = 0; T[1:N] = Tt; T[N] = 0
    return x, T

x, T = advec_diff(16, 0.1)
x2, T2 = advec_diff(16, 1)

plt.figure(figsize=(8,4))
plt.plot(x, T, color = 'black', lw = 2, label=r'$\nu = 0.1$')
plt.plot(x2, T2, color = 'gray', lw = 2, label=r'$\nu = 1$')
plt.xlabel(r'X-Axis', fontsize = 16)
plt.ylabel(r'Approximated $u$', fontsize = 16)
plt.legend(fontsize = 18)
plt.savefig('nu_values.pdf', bbox_inches = 'tight')
plt.show()



Ns = [4, 8, 16, 32]
plt.figure(figsize=(8,4))
for i in range(len(Ns)):
    x, T = advec_diff(Ns[i], 0.1)
    plot_label = r'$N$ = ' + str(Ns[i])
    plt.plot(x, T, lw = 2, label = plot_label)
plt.xlabel(r'X-Axis', fontsize = 16)
plt.ylabel(r'Approximated $u$', fontsize = 16)
plt.legend(fontsize = 18)
plt.savefig('varying_Ns.pdf', bbox_inches = 'tight')
plt.show()
```