AMath 586

Vivek Gandhi

April 21, 2025

T1

T1 (a)

Using Taylor series to find $f(x+p)$ and $f(x-p)$, centered around x:

$$f(x+p) = f(x) + pf'(x) + \frac{p^2}{2}f''(x) + \frac{p^3}{6}f'''(x) + \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5)$$
$$f(x-p) = f(x) - pf'(x) + \frac{p^2}{2}f''(x) - \frac{p^3}{6}f'''(x) + \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5)$$

Rearranging the terms and adding them:

$$\begin{aligned}
pf'(x) &= f(x+p) - f(x) - \frac{p^2}{2}f''(x) - \frac{p^3}{6}f'''(x) - \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5) \\
pf'(x) &= f(x) - f(x-p) + \frac{p^2}{2}f''(x) - \frac{p^3}{6}f'''(x) + \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5) \\
2pf'(x) &= f(x+p) - f(x-p) - \frac{p^3}{3}f'''(x) + \mathcal{O}(p^5) \\
f'(x) &= \frac{f(x+p) - f(x-p)}{2p} - \frac{p^2}{6}f'''(x) + \mathcal{O}(p^4)
\end{aligned}$$

Substituting $p = h/2$:

$$f'(x) = \frac{f(x+h/2) - f(x-h/2)}{h} - \frac{h^2}{24}f'''(x) + \mathcal{O}(h^4) \tag{1}$$

Using equation (1) with $f(x) = p(x_i)u'(x_i)$ to find $(p(x_i)u'(x_i))'$:

$$(p(x_i)u'(x_i))' = \frac{p(x_{i+1/2})u'(x_{i+1/2}) - p(x_{i-1/2})u'(x_{i-1/2}))}{h} - \frac{h^2}{24}(p(x_i)u'(x_i))''' + \mathcal{O}(h^4) \tag{2}$$

We know that $p \in C^4([a,b])$. But for the $\mathcal{O}(h^2)$ error term to be valid, we now require $u \in C^4([a,b])$ to also be true.

Using equation (1) again to expand $u'(x_{i+1/2})$ and $u'(x_{i-1/2})$:

$$
\begin{aligned}
u'(x_{i+1/2}) &= \frac{u(x_{i+1}) - u(x_i)}{h} - \frac{h^2}{24}u'''(x_{i+1/2}) + \mathcal{O}(h^4) \\
u'(x_{i-1/2}) &= \frac{u(x_i) - u(x_{i-1})}{h} - \frac{h^2}{24}u'''(x_{i-1/2}) + \mathcal{O}(h^4)
\end{aligned}
$$

Combining these like in equation (2):

$$
\begin{aligned}
&p(x_{i+1/2})u'(x_{i+1/2}) - p(x_{i-1/2})u'(x_{i-1/2})) \\
&\qquad = \frac{p(x_{i+1/2})u(x_{i+1}) - (p(x_{i+1/2}) + p(x_{i-1/2}))u(x_i) + p(x_{i-1/2})u(x_{i-1})}{h} \\
&\qquad - \frac{h^2}{24}\left[p(x_{i+1/2})u'''(x_{i+1/2}) - p(x_{i-1/2})u'''(x_{i-1/2})\right] + \mathcal{O}(h^4)
\end{aligned}
$$

Here, the $\mathcal{O}(h^2)$ error term can be combined into one $\mathcal{O}(h^3)$ term.
Using the Taylor series approximation from before, with $f(x) = p(x_i)u'''(x_i)$:

$$
\begin{aligned}
f'(x) &= \frac{f(x + h/2) - f(x - h/2)}{h} - \frac{h^2}{24}f'''(x) + \mathcal{O}(h^4) \\
f(x + h/2) - f(x - h/2) &= hf'(x) + \frac{h^3}{24}f'''(x) + \mathcal{O}(h^5) \\
p(x_{i+1/2})u'''(x_{i+1/2}) - p(x_{i-1/2})u'''(x_{i-1/2}) &= h(p(x_i)u'''(x_i))' + \mathcal{O}(h^3)
\end{aligned}
$$

The error terms then become:

$$
-\frac{h^2}{24}\left[p(x_{i+1/2})u'''(x_{i+1/2}) - p(x_{i-1/2})u'''(x_{i-1/2})\right] + \mathcal{O}(h^4) = -\frac{h^3}{24}(p(x_i)u'''(x_i))' + \mathcal{O}(h^4)
$$

Plugging this back into (2):

$$
\begin{aligned}
(p(x_i)u'(x_i))' &= \frac{p(x_{i+1/2})u(x_{i+1}) - (p(x_{i+1/2}) + p(x_{i-1/2}))u(x_i) + p(x_{i-1/2})u(x_{i-1})}{h^2} \\
&\quad - \frac{h^2}{24}\left[(p(x_i)u'''(x_i))' + (p(x_i)u'(x_i))'''\right] + \mathcal{O}(h^3)
\end{aligned}
$$

Again, we see that $u \in C^4[a, b]$ needs to be true for the $\mathcal{O}(h^4)$ error term to be valid.
Using the following scheme requires dropping the $\mathcal{O}(h^2)$ and higher terms. Therefore, it has a consistency error (LTE) of $\mathcal{O}(h^2)$. As the other terms in the equation are exact, they don't create consistency error.

$$
(p(x_i)u'(x_i))' \approx \frac{p(x_{i+1/2})U_{i+1} - (p(x_{i+1/2}) + p(x_{i-1/2}))U_i + p(x_{i-1/2})U_{i-1}}{h^2}
$$

T1 (b)

The resulting system can be written as a linear system of the form $AU = F$:

$$A \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ \\ U_N \end{bmatrix} = \begin{bmatrix} f(x_1) + \frac{p(x_{1/2})}{h^2}\alpha \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) + p(x_{n+1/2})\frac{\beta}{h^2} \end{bmatrix}$$

$$A = \frac{1}{h^2} R = \frac{1}{h^2} \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}$$

$$a_i = p(x_{i-1/2}) + p(x_{i+1/2}) + q(x_i)h^2$$
$$b_i = -p(x_{i+1/2})$$

My claim is that with a matrix R of size $n \times n$, the determinant $\det(R) = D_n$ is greater than or equal to $(n+1)\tilde{c}^n$, which is greater than 0.
Note that:

$$\begin{aligned} a_i &= p(x_{i-1/2}) + p(x_{i+1/2}) + q(x_i)h^2 \\ &\geq 2\tilde{c} + 0 = 2\tilde{c} && (p(x) \geq \tilde{c} > 0, \text{ and } q(x) \geq 0) \\ b_i &= -p(x_{i+1/2}) \leq \tilde{c} \end{aligned}$$

The n-th dimension determinant D_n can be computed recursively:

$$\begin{aligned} D_n &= a_n D_{n-1} - b_{n-1}^2 D_{n-2} \\ D_1 &= a_1 \geq 2\tilde{c} = (1+1)\tilde{c}^2 \\ D_2 &= a_1 a_2 - b_1^2 \geq 4\tilde{c}^2 - \tilde{c}^2 = (2+1)\tilde{c}^2 \end{aligned}$$

The 2 base cases D_1 and D_2 satisfy this inequality $D_n \geq (n+1)\tilde{c}^n$.
For induction, assume that this is true for the D_{n-1} and D_{n-2} cases for some $n \geq 3$.

$$\begin{aligned} a_n D_{n-1} &\geq 2\tilde{c}(n)\tilde{c}^{n-1} = 2n\tilde{c}^n \\ b_n^2 D_{n-2} &\leq \tilde{c}^2(n-1)\tilde{c}^{n-2} = (n-1)\tilde{c}^n \\ a_n D_{n-1} - b_n^2 D_{n-2} &\geq (2n - (n-1))\tilde{c}^n && \text{(Subtracting the two)} \\ D_n &\geq (n+1)\tilde{c}^n \end{aligned}$$

As the $n-1$ and $n-2$ cases imply the n-th case, by induction, the $D_n \geq (n+1)\tilde{c}^n$ for all $n \geq 1$.

As $A = \frac{1}{h^2}R$ and $D_n = \det(R)$,

$$\begin{aligned} \det(A) &= (\frac{1}{h^2})^n \det(R) \\ &\geq \frac{(n+1)}{h^{2n}} \tilde{c}^n \end{aligned}$$

As $n \geq 1, \quad h > 0$, and $\tilde{c} > 0$, $\det(A) > 0$ must be true.
This implies that A is invertible, and the system $AU = F$ must have a unique solution.

T2

T2 (a)

$$\begin{aligned} -p(x_i)\frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} - p'(x_i)\frac{U_{i+1} - U_{i-1}}{2h} + q(x_i)U_i = f(x_i) \\ U_{i+1}\left(\frac{-p(x_i)}{h^2} - \frac{p'(x_i)}{2h}\right) + U_i\left(\frac{2p(x_i)}{h^2} + q(x_i)\right) + U_{i-1}\left(\frac{-p(x_i)}{h^2} + \frac{p'(x_i)}{2h}\right) = f(x_i) \end{aligned}$$

Assuming 0 boundary conditions - this finite difference scheme can then be written as the system

$$A \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}, \qquad A = \begin{bmatrix} b_1 & c_1 & & \\ a_2 & b_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_n & b_n \end{bmatrix}$$

$$\begin{aligned} a_iU_{i-1} + b_iU_i + c_iU_{i+1} &= f(x_i) \\ a_i &= \frac{-p(x_i)}{h^2} + \frac{p'(x_i)}{2h} \\ b_i &= \frac{2p(x_i)}{h^2} + q(x_i) \\ c_i &= \frac{-p(x_i)}{h^2} - \frac{p'(x_i)}{2h} \end{aligned}$$

As the finite difference scheme only uses U_i, U_{i+1}, and U_{i-1}, we get a tridiagonal matrix.
But as $c_i \neq a_{i+1}$, this matrix is not symmetric.
$p'(x_i)$ can be approximated using a similar finite difference scheme if necessary, but this doesn't change the symmetry of the matrix as $p(x_{i+1}) \neq p(x_i)$:

$$p'(x_i) \approx \frac{p(x_{i+1/2}) - p(x_{i-1/2})}{h}$$

T2 (b)

$$-u''(x) + a(x)u'(x) + b(x)u(x) = f(x), \qquad a < x < b, \quad u(a) = \alpha, \; u(b) = \beta$$

From **T1**, we have these:

$$f(x+p) = f(x) + pf'(x) + \frac{p^2}{2}f''(x) + \frac{p^3}{6}f'''(x) + \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5) \tag{1}$$

$$f(x-p) = f(x) - pf'(x) + \frac{p^2}{2}f''(x) - \frac{p^3}{6}f'''(x) + \frac{p^4}{24}f^{(4)}(x) + \mathcal{O}(p^5) \tag{2}$$

$$f'(x) = \frac{f(x+p) - f(x-p)}{2p} - \frac{p^2}{6}f'''(x) + \mathcal{O}(p^4)$$

$$f'(x) \approx \frac{f(x+p) - f(x-p)}{2p}, \qquad \text{LTE} = \mathcal{O}(p^2) \tag{3}$$

To derive an approximation of $f''(x)$, we add (1) and (2):

$$\begin{aligned} f(x+p) + f(x-p) &= 2f(x) + p^2 f''(x) + \frac{p^4}{12}f^{(4)}(x) + \mathcal{O}(p^5) \\ p^2 f''(x) &= f(x+p) - 2f(x) + f(x-p) - \frac{p^4}{12}f^{(4)}(x) + \mathcal{O}(p^5) \\ f''(x) &= \frac{f(x+p) - 2f(x) + f(x-p)}{p^2} - \frac{p^2}{12}f^{(4)}(x) + \mathcal{O}(p^3) \\ f''(x) &\approx \frac{f(x+p) - 2f(x) + f(x-p)}{p^2}, \qquad \text{LTE} = \mathcal{O}(p^2) \end{aligned} \tag{4}$$

Using (3) to approximate $a(x)u'(x)$:

$$a(x)u'(x) \approx \frac{a(x_i)U_{i+1} - a(x_i)U_{i-1}}{2h}$$

Using (4) to approximate $-u''(x)$:

$$-u''(x) \approx \frac{-U_{i-1} + 2U_i - U_{i+1}}{h^2}$$

Combining them:

$$\frac{-U_{i-1} + 2U_i - U_{i+1}}{h^2} + \frac{a(x_i)U_{i+1} - a(x_i)U_{i-1}}{2h} + b(x_i)U_i = f(x_i)$$

$$U_{i-1}\left(\frac{-1}{h^2} - \frac{a(x_i)}{2h}\right) + U_i\left(\frac{2}{h^2} + b(x_i)\right) + U_{i+1}\left(\frac{-1}{h^2} + \frac{a(x_i)}{2h}\right) = f(x_i)$$

This finite difference scheme has a consistency error of $\mathcal{O}(h^2)$.

It can be written as the following linear system:

$$A \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ \\ U_N \end{bmatrix} = \begin{bmatrix} f(x_1) + \alpha\left(\frac{1}{h^2} + \frac{a(x_1)}{2h}\right) \\ f(x_2) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) + \beta\left(\frac{1}{h^2} - \frac{a(x_n)}{2h}\right) \end{bmatrix}, \qquad A = \begin{bmatrix} b_1 & c_1 & & \\ a_2 & b_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_n & b_n \end{bmatrix}$$

$$\begin{aligned} a_i U_{i-1} + b_i U_i + c_i U_{i+1} &= f(x_i) \\ a_i &= \frac{-1}{h^2} - \frac{a(x_i)}{2h} \\ b_i &= \frac{2}{h^2} + b(x_i) \\ c_i &= \frac{-1}{h^2} + \frac{a(x_i)}{2h} \end{aligned}$$

For this system to have a unique solution, the matrix A has to be non-singular (no $AU = 0$ solutions), or equivalently, if all its eigenvalues are non-zero.
We can find a sufficient condition to ensure this using Gershgorin's theorem.
For any matrix $A \in \mathbb{C}^{n\times n}$, define the disk:

$$G_i(A) = \{z \in \mathbb{C} : |z - a_{ii}| \le \sum_{j=1, j\neq i}^{n} |a_{ij}|\}, \quad i = 1, \dots, n$$

Gerschgorin's theorem says that $\Lambda(A) \subset \bigcup_{i=1}^{m} G_i(A)$.
As our matrix only has 3 elements per row, the disks can be reduced to:

$$G_i(A) = \{z \in \mathbb{C} : |z - b_i| \le |a_i| + |c_i|\}, \quad i = 1, \dots, n$$

This forms a circle around b_i with radius $|a_i| + |c_i|$. So for 0 to not be within this ring, we require $|b_i| > |a_i| + |c_i|$.

$$\begin{aligned} |b_i| &> |a_i| + |c_i| \\ \left|\frac{2}{h^2} + b(x_i)\right| &> \left|\frac{-1}{h^2} - \frac{a(x_i)}{2h}\right| + \left|\frac{-1}{h^2} + \frac{a(x_i)}{2h}\right| \\ \frac{2}{h^2} + b(x_i) &> \left|\frac{1}{h^2} + \frac{a(x_i)}{2h}\right| + \left|\frac{1}{h^2} - \frac{a(x_i)}{2h}\right| && (b(x_i) > 0) \\ &= \qquad A \qquad + \qquad B \end{aligned}$$

We can analyze the 4 cases separately based on the sign of A and B.

Case 1: $A \ge 0, B \ge 0$:

This happens when $\left|\frac{a(x_i)}{2h}\right| \leq \frac{1}{h^2}$, or $|a(x_i)| \leq \frac{2}{h}$.

$$\begin{aligned}\frac{2}{h^2} + b(x_i) &> \frac{1}{h^2} + \frac{a(x_i)}{2h} + \frac{1}{h^2} - \frac{a(x_i)}{2h} \\ &= \frac{2}{h^2}\end{aligned}$$

As $b(x_i) > 0$, this is always true.

Case 2: $A > 0$ **and** $B < 0$:
This happens when $\frac{a(x_i)}{2h} > \frac{1}{h^2}$, or $a(x_i) > \frac{2}{h}$.

$$\begin{aligned}\frac{2}{h^2} + b(x_i) &> \frac{1}{h^2} + \frac{a(x_i)}{2h} - \frac{1}{h^2} + \frac{a(x_i)}{2h} \\ &= \frac{a(x_i)}{h}\end{aligned}$$

As $a(x_i)/h > 2/h^2$, there's no condition that will ensure that $2/h^2 + b(x_i) > a(x_i)/h$ for all b.

Case 3: $A < 0$ **and** $B < 0$**:**
This happens when $\frac{a(x_i)}{2h} < -\frac{1}{h^2}$ and $\frac{a(x_i)}{2h} > \frac{1}{h^2}$. As $1/h^2 > 0$, these conditions can never be met.

Case 4: $A < 0$ **and** $B > 0$**:**
This happens when $\frac{a(x_i)}{2h} < -\frac{1}{h^2}$, or $a(x_i) < -\frac{2}{h}$.

$$\begin{aligned}\frac{2}{h^2} + b(x_i) &> -\frac{1}{h^2} - \frac{a(x_i)}{2h} + \frac{1}{h^2} - \frac{a(x_i)}{2h} \\ &= \frac{-a(x_i)}{h}\end{aligned}$$

As $-a(x_i)/h > 2/h^2$, there's no condition that will ensure that $2/h^2 + b(x_i) > -a(x_i)/h$ for all b.

Considering the left and right boundaries:
As a_1 and c_n don't exist, the terms A and B are omitted from the RHS. As they only add absolute values, to the lesser side of the inequality, omitting them only relaxes the condition further.

Conclusion:
The system always has a solution (A has no 0 eigenvalues) when $|a(x_i)| \leq \frac{2}{h}$.
So by choosing $h \leq \dfrac{2}{\max|a(x)|}$, we can ensure that the system has a consistent solution.

T3

T3 (a)

$$-u'' = f(x), \qquad x \in (0,1), \quad u'(0) = \alpha, \quad u'(1) = \beta$$

Consider $v(x) = u'(x)$. The problem becomes:

$$v' = -f(x), \qquad x \in (0,1), \quad v(0) = \alpha, \quad v(1) = \beta$$

From the fundamental theorem of calculus:

$$\begin{aligned} v(1) &= v(0) + \int_0^1 v'(x)\mathrm{d}x \\ \beta &= \alpha - \int_0^1 f(x)\mathrm{d}x \end{aligned}$$

So, a solution only exists when $\int_0^1 f(x)\mathrm{d}x = \alpha - \beta$.
With a solution to $v(x) = u'(x)$, we still need a condition to ensure the uniqueness of the solution.
Given a solution $u(x)$, $w(x) = u(x) + C$ for any constant C will also be a solution, as $w'(x) = u'(x)$.

T3 (b)

From **T2**, we have:

$$f''(x) \approx \frac{f(x+p) - 2f(x) + f(x-p)}{p^2}, \qquad \text{LTE} = \mathcal{O}(p^2)$$

This gives us the finite difference scheme:

$$\frac{-U_{i-1} + 2U_i - U_{i+1}}{h^2} = f(x_i)$$

To apply the left boundary condition, we create a ghost point U_{-1}, use it to derive formulas for $u''(0)$ and $u'(0)$, and combine them to get rid of the ghost point.

$$\frac{-U_{-1} + 2U_0 - U_1}{h^2} = -u''(0) = f(0) \tag{1}$$

We use a central difference formula as it is $\mathcal{O}(h^2)$

$$\begin{aligned} \frac{U_1 - U_{-1}}{2h} &= u'(0) = \alpha \\ U_{-1} &= U_1 - 2h\alpha \end{aligned} \tag{2}$$

Plugging (2) into (1)

$$\begin{aligned} \frac{2U_0 - 2U_1}{h^2} + \frac{2h\alpha}{h^2} &= f(0) \\ \frac{2U_0 - 2U_1}{h^2} &= f(0) - \frac{2\alpha}{h} \end{aligned}$$

Similarly for the right boundary condition, using ghost point U_{n+2}:

$$\frac{-U_{n+2} + 2U_{n+1} - U_n}{h^2} = -u''(1) = f(1) \tag{3}$$

$$\frac{U_{n+2} - U_n}{2h} = u'(1) = \beta$$

$$U_{n+2} = U_n + 2h\beta \tag{4}$$

Plugging (4) into (3)

$$\frac{2U_{n+1} - 2U_n}{h^2} - \frac{2h\beta}{h^2} = f(1)$$

$$\frac{2U_{n+1} - 2U_n}{h^2} = f(1) + \frac{2\beta}{h}$$

This can be combined into the following linear system:

$$\begin{array}{ccc} A & U & = & F \\ \frac{1}{h^2}\begin{bmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{bmatrix} & \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_n \\ U_{n+1} \end{bmatrix} & = & \begin{bmatrix} f(0) - 2\alpha/h \\ f(x_1) \\ \vdots \\ f(x_n) \\ f(1) + 2\beta/h \end{bmatrix} \end{array}$$

The matrix here is singular, so we don't have a guaranteed or unique solution.
It's easy to see why it's singular. Multiply the top and bottom rows by $1/2$ to get $[1, -1, \dots]$ and $[\dots, -1, 1]$.
Starting from the top, adding the i-th row to the $(i+1)$-th row recursively leaves us with $[\dots, 1, -1, \dots]$ in each $(i+1)$-th row, which eventually cancels out the last row.

We can fix this by introducing a constraint on $u(0)$, say $u(0) = U_0 = 0$. As for any solution $u(x)$, $u(x) + C$ is also a solution, the value of this constraint doesn't matter. We get a general family of solutions either way.

$$\begin{array}{ccc} A & U & = & F \\ \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{bmatrix} & \begin{bmatrix} U_1 \\ \vdots \\ U_n \\ U_{n+1} \end{bmatrix} & = & \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \\ f(1) + 2\beta/h \end{bmatrix} \end{array}$$

A is no longer singular. Recall how solutions only exist if $\beta = \alpha - \int_0^1 f(x)\mathrm{d}x$. By removing our dependence on α and creating a consistent system, we ensure that the solution converges with $u'(0) \approx \beta + \int_0^1 f(x)\mathrm{d}x$.
As the system is only defined on $(0, 1]$, we also manage to avoid using $f(0)$ or approximations of $u'(0), u''(0)$.

T4

T4 (a)

Showing that $L_h\Phi_{i,j} = -1$:

$$L_h U_{i,j} := -\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} - \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2}$$

The L_h operator is linear in $U_{i-1,j}, U_{i+1,j}, U_{i,j}, U_{i,j+1}, U_{i,j-1}$.

$$\begin{aligned}
\Phi_{i,j} &= \frac{1}{4}((x_i - 1/2)^2 + (y_j - 1/2)^2) \\
L_h\Phi_{i,j} &= -\frac{\Phi_{i-1,j} - 2\Phi_{i,j} + \Phi_{i+1,j}}{h^2} - \frac{\Phi_{i,j-1} - 2\Phi_{i,j} + \Phi_{i,j+1}}{h^2} \\
&= \frac{4\Phi_{i,j}}{h^2} - \frac{1}{h^2}\left(\Phi_{i-1,j} + \Phi_{i+1,j} + \Phi_{i,j} + \Phi_{i,j+1} + \Phi_{i,j-1}\right) \qquad (1)
\end{aligned}$$

Note that

$$\begin{aligned}
(a+h)^2 + (a-h)^2 &= a^2 + 2ah + h^2 + a^2 - 2ah + h^2 \\
&= 2(a^2 + h^2) \qquad (2)
\end{aligned}$$

Choosing some terms from the right term in (1):

$$\begin{aligned}
\Phi_{i-1,j} + \Phi_{i+1,j} &= \frac{1}{4}((x_{i-1} - 1/2)^2 + (y_j - 1/2)^2) + \frac{1}{4}((x_{i+1} - 1/2)^2 + (y_j - 1/2)^2) \\
&= \frac{(y_j - 1/2)^2}{2} + \frac{(x_i - 1/2 - h)^2 + (x_i - 1/2 + h)^2}{4} \\
&= \frac{(y_j - 1/2)^2}{2} + \frac{(x_i - 1/2)^2 + h^2}{2} = 2\Phi_{i,j} + \frac{h^2}{2} \qquad \text{(Using (2))}
\end{aligned}$$

Choosing the other terms:

$$\begin{aligned}
\Phi_{i,j+1} + \Phi_{i,j-1} &= \frac{1}{4}((x_i - 1/2)^2 + (y_{j-1} - 1/2)^2) + \frac{1}{4}((x_i - 1/2)^2 + (y_{j+1} - 1/2)^2) \\
&= \frac{(x_i - 1/2)^2}{2} + \frac{(y_j - 1/2 - h)^2 + (y_j - 1/2 + h)^2}{4} \\
&= \frac{(x_i - 1/2)^2}{2} + \frac{(y_j - 1/2)^2 + h^2}{2} = 2\Phi_{i,j} + \frac{h^2}{2} \qquad \text{(Using (2))}
\end{aligned}$$

Combining these back into (1):

$$\begin{aligned}
L_h\Phi_{i,j} &= \frac{4\Phi_{i,j}}{h^2} - \frac{1}{h^2}\left(4\Phi_{i,j} + h^2\right) \\
&= 0 - 1 = -1
\end{aligned}$$

Therefore, $L_h\Phi_{i,j} = -1$.

Showing that $L_h\phi_{i,j} \leq 0$:

$$\phi_{i,j} := e_{i,j} + T\Phi_{i,j}$$

As the L_h operator is a linear operator,

$$\begin{aligned}
L_h\phi_{i,j} &= L_h e_{i,j} + T\left(L_h\Phi_{i,j}\right) \\
&= T_{i,j} + T\left(L_h\Phi_{i,j}\right) \\
&\leq T + T\left(L_h\Phi_{i,j}\right) && (\text{As } 0 \leq |T_{i,j}| \leq T,\ T_{i,j} \leq T) \\
&= T(1-1) = 0 && (L_h\Phi_{i,j} = -1)
\end{aligned}$$

Therefore, $L_h\phi_{i,j} \leq 0$.

Using the Discrete Maximum Principle:
Consider the system $-\Delta\phi(x,y) = p(x,y)$, defined on the same domain $\Omega = (0,1)^2$, with numerical solution $\phi_{i,j}$.

Consider the value of $\Phi_{i,j}$ on the domain $[0,1]^2$.
As Φ is a sum of squares in two different variables, minimizing or maximizing Φ can be split into two separate optimization problems.

$$\begin{aligned}
\Phi_{i,j} &= \frac{(x_i - 1/2)^2}{4} + \frac{(y_j - 1/2)^2}{4}, \qquad x_i, y_j \in [0,1] \\
\max \Phi_{i,j} &= \max_{x_i} \frac{(x_i - 1/2)^2}{4} + \max_{y_j} \frac{(y_j - 1/2)^2}{4} \\
&= \frac{(1/2)^2}{4} + \frac{(1/2)^2}{4} = \frac{1}{8} && (\text{At } 0 \text{ or } 1) \\
\min \Phi_{i,j} &= \min_{x_i} \frac{(x_i - 1/2)^2}{4} + \min_{y_j} \frac{(y_j - 1/2)^2}{4} \\
&= 0 && (\text{At } (1/2, 1/2))
\end{aligned}$$

Our finite difference scheme assumes $L_h\phi_{i,j} = p(x_i, y_j) \leq 0$ (as shown above).
As $p(x,y) \leq 0$, we can use the Discrete Maximum Principle to assume that the maximum value of $\phi_{i,j}$ occurs on the boundaries.

$$\begin{aligned}
\max_{\Omega} \phi_{i,j} &= \max_{\partial\Omega}(e_{i,j} + T\Phi_{i,j}) \\
&= T \max_{\partial\Omega} \Phi_{i,j} = \frac{T}{8} && (\text{The error } e_{ij} = 0 \text{ on the boundaries})
\end{aligned}$$

$$\begin{aligned}
\phi_{i,j} \leq \max_{\Omega} \phi_{i,j} &= \frac{T}{8} \\
e_{i,j} + T\Phi_{i,j} &\leq \frac{T}{8}
\end{aligned}$$

As $\min \Phi_{i,j} = 0$

$$e_{i,j} \leq \frac{T}{8}$$

T4 (b)

Showing that $L_h\bar{\phi}_{i,j} \geq 0$:

$$\bar{\phi}_{i,j} := e_{i,j} - T\Phi_{i,j}$$

As the L_h operator is a linear operator,

$$\begin{aligned} L_h\bar{\phi}_{i,j} &= L_h e_{i,j} - T\left(L_h\Phi_{i,j}\right) \\ &= T_{i,j} - T\left(L_h\Phi_{i,j}\right) \\ &\geq -T - T\left(L_h\Phi_{i,j}\right) && \text{(As } 0 \leq |T_{i,j}| \leq T\text{, } -T_{i,j} \geq -T\text{)} \\ &= T(-1+1) = 0 && (L_h\Phi_{i,j} = -1) \end{aligned}$$

Therefore, $L_h\bar{\phi}_{i,j} \geq 0$.

Using the Discrete Minimum Principle:
Consider the system $-\Delta\bar{\phi}(x,y) = p(x,y)$, defined on the same domain $\Omega = (0,1)^2$, with numerical solution $\bar{\phi}_{i,j}$.

Our finite difference scheme assumes $L_h\bar{\phi}_{i,j} = p(x_i, y_j) \geq 0$ (as shown above).
As $p(x,y) \geq 0$, we can use the Discrete Minimum Principle to assume that the minimum value of $\bar{\phi}_{i,j}$ occurs on the boundaries.

$$\begin{aligned} \min_{\Omega}\bar{\phi}_{i,j} &= \min_{\partial\Omega}(e_{i,j} + T\Phi_{i,j}) \\ &= T\min_{\partial\Omega}\Phi_{i,j} = 0 && \text{(The error } e_{ij} = 0 \text{ on the boundaries)} \end{aligned}$$

$$\begin{aligned} \bar{\phi}_{i,j} &\geq \min_{\Omega}\bar{\phi}_{i,j} = 0 \\ e_{i,j} - T\Phi_{i,j} &\geq 0 \end{aligned}$$

As $\max\Phi_{i,j} = \frac{T}{8}$

$$e_{i,j} \geq \frac{T}{8}$$

C1

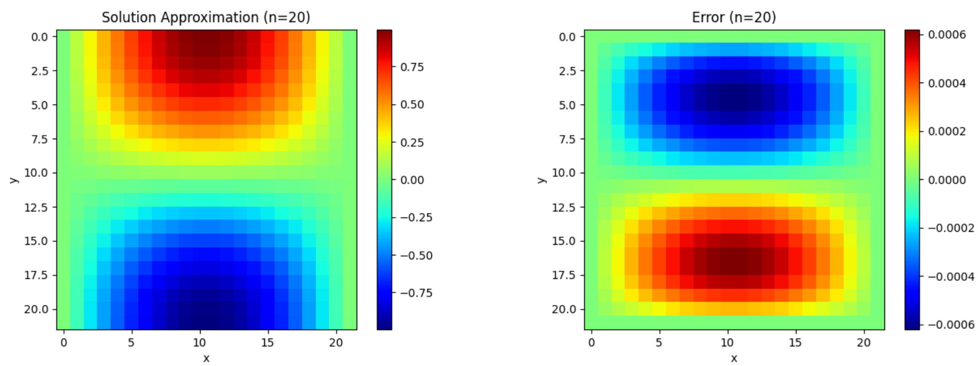

Figure 1: Numerical solution and error against analytic solution ($n = 20$)

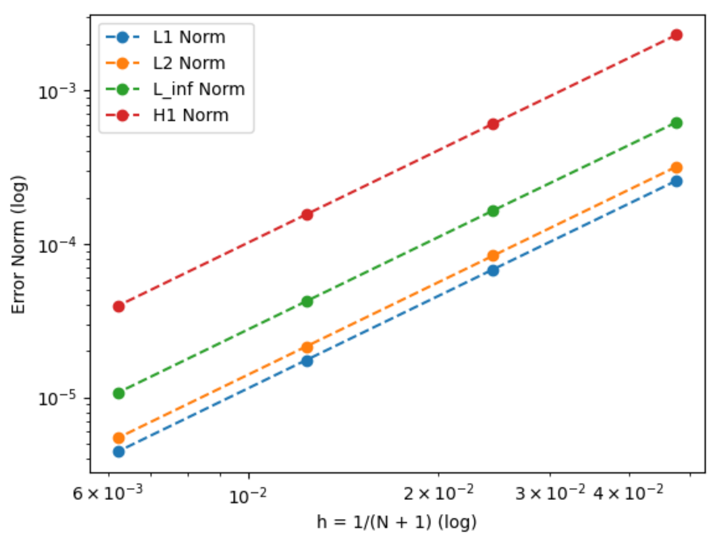

Figure 2: Solution errors with different values of h in different norms on a log-log plot

Norms	Slope
L_1	1.9971
L_2	1.9999
L_∞	1.9943
H^1	1.9979

Table 1: I used polyfit to find the slopes of each line on the log-log plot. This shows that the solution does indeed converge quadratically ($\mathcal{O}(h^2)$) in each norm.

```
import numpy as np

def NormL1(v, h):
  return (h**2)*np.sum(np.abs(v[1:-1, 1:-1]))

def NormLInf(v, h):
  return np.max(np.abs(v[1:-1, 1:-1]))

def NormL2(v, h):
  v = v[1:-1, 1:-1]
  return h * np.sqrt(np.sum(v**2))

def NormH1(v, h):
  return np.sqrt(NormL2(v, h)**2 + DxV_xhNorm2(v, h) + DyV_yhNorm2(v, h))

def DxV_xhNorm2(v, h):
  base = np.copy(v[1:-1, 1:])
  base = base - v[1:-1, :-1]
  return np.sum(base**2)

def DyV_yhNorm2(v, h):
  return DxV_xhNorm2(v.transpose(), h)
```

Listing 1: 'Norms.py'

```

import numpy as np
import matplotlib.pyplot as plt

# For A
from scipy.sparse import spdiags
# To solve AL = U
from scipy.sparse.linalg import gmres, spsolve
from scipy.linalg import solve
from Norms import *

# ----------------------------------------------

do_matrix_viz = False # True to see matrix A's structure
do_viz = False       # True to plot the solution and error
mat_viz_mode = 0

# ==============================================

def main():
  Ns = [20, 40, 80, 160]
```

```
    Hs = [1/(n+1) for n in Ns]

    L1s = []
    L2s = []
    Linfs = []
    H1s = []

    for n in Ns:
        data = gather_norms(n)

        L1s.append(data[0])
        L2s.append(data[1])
        Linfs.append(data[2])
        H1s.append(data[3])

    get_slope = lambda errors: np.polyfit(np.log10(Hs), np.log10(errors), 1)[0]

    plt.loglog(Hs, L1s, label='L1 Norm', linestyle='--', marker='o')
    plt.loglog(Hs, L2s, label='L2 Norm', linestyle='--', marker='o')
    plt.loglog(Hs, Linfs, label='L_inf Norm', linestyle='--', marker='o')
    plt.loglog(Hs, H1s, label='H1 Norm', linestyle='--', marker='o')
    plt.legend()
    plt.xlabel('h = 1/(N + 1) (log)')
    plt.ylabel('Error Norm (log)')
    plt.show()

    print(f'L1 slope - {get_slope(L1s)}')
    print(f'L2 slope - {get_slope(L2s)}')
    print(f'L_inf slope - {get_slope(Linfs)}')
    print(f'H1 slope - {get_slope(H1s)}')

# ==============================================

def gather_norms(n = 200):

    h = 1/(n+1)
    h2_n = (n+1)**2

    # ------ Setup -----------------------

    xRange = np.linspace(0, 1, n+2)
    yRange = np.linspace(0, 1, n+2)

    # ------ Produce A -------------------

    A = buildA(n, h2_n)
    vizMatrix(A)

    # ------ Produce F -------------------

    BC_0 = np.sin(np.pi*xRange[1:-1])
    BC_1 = -np.sin(np.pi*xRange[1:-1])
    BC = np.concatenate((BC_0, np.zeros(n*(n-2)), BC_1))

    initializer = lambda y, x: 2*(np.pi**2)*np.sin(np.pi*x)*np.cos(np.pi*y)
    w0 = generate_w0(xRange[1:-1], yRange[1:-1], initializer)

```

```
  F = deform(w0) + h2_n*deform(BC)

  # ------ Solve for U -----------------

  U_sol = spsolve(A, F)       # Sparse solve
  # U_sol = solve(A.toarray(), F)    # Dense solve
  # U_sol = gmres(A, F)[0]      # GMREs

  U_sol = reform(U_sol)

  # ------ Append BCs ------------------

  U = np.zeros((n+2, n+2))
  U[1:1+U_sol.shape[0], 1:1+U_sol.shape[1]] = U_sol
  U[0,:] = np.sin(np.pi*xRange)
  U[-1,:] = -np.sin(np.pi*xRange)

  viz(U, f'Solution (n={n})')

  # ------ Compare against solution-----

  initializer = lambda y, x: np.sin(np.pi*x)*np.cos(np.pi*y)
  real = generate_w0(xRange, yRange, initializer)

  error = real-U
  viz(error, f'Error (n={n})')

  return [NormL1(error, h), NormL2(error, h),
        NormLInf(error, h), NormH1(error, h)]

# ==============================================
# Helpers
# ==============================================

# Unroll an nxn matrix into an mx1 one
def deform(A):
  return A.reshape((A.size, 1))

# ----------------------------------------------
# Reform an mx1 matrix into an nxn one
def reform(A):
  n = np.sqrt(A.size).astype(int)
  return A.reshape((n, n))

# ----------------------------------------------
# Generate an nxn matrix across ranges with an initializer func
def generate_w0(xRange, yRange, func):
  w0 = np.zeros((yRange.size, xRange.size))
  for x in range(xRange.size):
    for y in range(yRange.size):
      w0[x,y] = func(xRange[x], yRange[y])
  return w0

# ----------------------------------------------
# Vizualize phi or w
def viz(A, title):
  if do_viz:
```

```
    plt.imshow(A, cmap='jet')
    plt.colorbar()
    plt.title(title)
    plt.show()

# ---------------------------------------------
# Visualize matrix A using either [0] plt.spy() or [1] plt.imshow()
def vizMatrix(mat):
  global mat_viz_mode
  if not do_matrix_viz:
    return
  if mat_viz_mode == 0:
    plt.figure(5)
    plt.spy(mat)
  else:
    plt.imshow(mat.toarray(), interpolation='none', cmap='binary')
    plt.colorbar()
  plt.title('Matrix Structure')
  plt.show()

# =============================================
# A = -(dxx + dyy)
# =============================================

def buildA(n, h2_n):
  m = n**2          # Size of A

  e1 = np.ones(m)      # vector of 1s
  e4 = 4*np.copy(e1)     # vector of 4s

  e_n = np.ones(n)
  e_n[-1] = 0
  e_n = -1*np.tile(e_n, n)  # vector of -1s, with 0 at the end

  e_m = np.roll(e_n, 1)

  diagonals = [-1*e1, e_n, e4, e_m, -1*e1]
  offsets = [-n, -1, 0, 1, n]

  matA = spdiags(diagonals, offsets, m, m, format = 'csr')
  return h2_n * matA

# =============================================

main()
```

Listing 2: 'HW1-C1.py