Chandler Justice - A02313187

January 22, 2024

Homework:
For question 4: determine

$$D_n h(y) = c_1 u(x_1) + c_2 u(x_2) + ... + c_n u(x_n)$$

What we will want to do is take the vanderbaun matrix and multiply it by our constants to get a vector of all 0s and a 1
Example:

$$u''(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} = err$$

$$|E| = |u''(x) - \frac{1}{h^2}(u(x+h) - 2u(x) + u(x-h)|$$

$$= |u''(x) - \frac{1}{h^2}\{(u(x) + hu'(x) + \frac{1}{2}h^2u''(x) + \frac{h^3}{6}u'''(x) + \frac{h^4}{24}u''''(\zeta_1))$$

$$-2u(u(x) - hu(x) + \frac{1}{2}h^2u''(x) - \frac{1}{6}h^3(x) + \frac{h^4}{24}u''''(\zeta_2))\}$$

$$= |u'' - \frac{1}{h^2}\{h^2u''(x) + \frac{1}{24}h^4u''''(\zeta_3)\}|$$

Heat Equation: Heat equation:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial k}k(x)\frac{\partial u}{\partial x} + t(x,t)$$

If we assume k is continuous

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{\partial x^2} + t(x,t)$$

steady state = 1 $\Rightarrow \frac{\partial u}{\partial t} = 0 \Rightarrow Ku'' + t(x,t) = 0$

$$\begin{cases} u'' = f(x) \\ u(0) = \alpha \\ u(1) = \beta \end{cases}$$

Exact solution:

$$\int u''(x)dx = \int f(x)dx = g(x) + c$$

$$u' = g(x) + c_1$$

$$u = \int g(x) + c_1x + c_2$$

We could decide to get an approximation at discrete points in the domain. Lets our domain be $[0,1]$.

So we will use equally spaced (for now) points in $[0,1]$, say $m+2$ points. Then

$$h = \frac{1}{m+1} \Rightarrow \{u_0, u_1, ..., u_m, u_{m+1}\} \quad \text{(size = } m+2)$$

$$x_j = j * h$$

$$u_0 = \alpha$$

$$u_{m+1} = \beta$$

our u_0, u_{m+1} variables will be exact values.
$u'' = f(x)$ at $x_0, x_1, ...x_n$

$$D^2 u_j = \frac{u_{j-1} 2u_j + u_{j+1}}{h^2}$$

so

$$\frac{1}{h^2}(u_{j-1} - 2u_j + u_{j+1}) \approx f(x_j)$$

for $j = 0, 1, 2, ..., m+1$

$$\begin{array}{cc} j = 0 & u_0 = \alpha \\ j = 1 & \frac{1}{h^2}(u_0 - 2u_1 + u_2) = f(x_1) \\ j = 2 & \frac{1}{h^2}(u_1 - 2u_2 + u_3) = f(x_1) \\ ... & ... \\ j = m & \frac{1}{h^2}(u_{m-1} - 2u_m + u_{m+1}) = f(x_m) \end{array}$$

$$\begin{bmatrix} 1 & 0 & ... & 0 & 0 \\ -\frac{1}{h^2} & -\frac{2}{h^2} & ... & 0 & \\ 0 & -\frac{1}{h^2} & -\frac{2}{h^2} & ... & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_n \end{bmatrix} = \begin{bmatrix} \alpha \\ f_1 \\ f_2 \\ f_n \\ \beta \end{bmatrix}$$

Then we get

$$\begin{bmatrix} \frac{\alpha}{n^2} \\ 0 \\ ... \\ 0 \\ \frac{\beta}{n^2} \end{bmatrix} \begin{bmatrix} -2 & 1 & 0 & ... & 0 & 0 \\ 1 & -2 & 1 & 0 & ... & 0 \\ 0 & 1 & -2 & 1 & 0 & ... \\ ... & ... & ... & ... & & \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ ... \\ f_m \end{bmatrix}$$

To summarize we get a matrix A that we multiply by a vector U to get the vector of functions F.

We need to be able to define and interpret an error. Suppose we define

$$\hat{U} = \begin{bmatrix} u(x_1) \\ u(x_2) \\ ...u(x_m) \end{bmatrix}, \quad U = \begin{bmatrix} u_1 \\ u_2 \\ ...u_n \end{bmatrix} \in \mathbb{R}$$

Then

$$E = U - \hat{U}$$

Definition: *Vector norm -* Any function $|| * ||: \mathbb{R}^m \in \mathbb{R}$ is a norm if

1. for any vector $v \in \mathbb{R}^m$, $||v|| \geq 0$ and $||v|| = 0$ iff $x = 0$
2. for any vector $v \in \mathbb{R}^m$ and scaler a, $||av|| = |a| * ||v||$
3. for $u, v \in \mathbb{R}^m$, $||u + v|| \leq ||u|| + ||v||$

The norms:

- $||E||_\infty = max_{1 \leq j \leq m} |u_j - u(xj)|$
- $||E||_1 = h \sum_{j=1}^{m} |u_j - u(x_j)|$
- $||E||_2 = (h \sum_{j=1}^{n} |u - u(x_j)|^2)^{\frac{1}{2}}$

January 24, 2024

Example: 2 point boundary problem

$$\begin{cases} u'' = f(x) & 0 < x < 1 \\ u(0) = \alpha \\ u(1) = \beta \end{cases}$$

Lets employ a centered difference

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

If we want the error we can compute

$$|e(x)| = |u''(x) - \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}| \leq O(h^2)$$

We want to approximate u at discrete points. To do this, we need to pick points over an interval where each point is h apart from the next, $h = 1/(m+1)$, $x_j = j * h$, and $j = \{1, 2, 3, 4, ..., m+1\}$.

Definition:

$$u(x_j) = u_j \to u''(x_j) = \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} \quad j = 1, 2, ...m$$

$$\to \frac{u_{j-1} - 2u_j + u_{j+1}}{h^2}$$

Working through these computations we get $AU = F$

$$\begin{bmatrix} -2 & 1 & 0 & ... & 0 \\ 1 & -2 & 1 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ ... \\ u_m \end{bmatrix} = \begin{bmatrix} f_1 - \alpha/h^2 \\ f_2 \\ ... \\ f_{m-1} \\ f_{m-1} - Ch^2 \end{bmatrix}$$

In general, we can use the classic $Ax = b$ method from linear algebra to solve these systems of equations. We can do this in code with

```
for k = 1, m = 1
    for i = k + 1, m
        factor = a[i][k]
        for j = k + 1, m
            a[i][j] = a[i][j] - factor * a[i][j]
        end
        b[i] = b[i] - factor * b[j]
    end
end
```

For a tridiagonal matrix we can perform elimination via

```
for k=1, m=1
    factor = a[k + 1][k] / a[k][k]
    a[k + 1][k+1] = a[k+1][k+1] - factor * a[k][k+1]
    b[k+1] = b[k+1] - factor * b[k]
end
```

After running this algorithm on a tridiagonal matrix, it will result in a matrix with all zeros except for along the two center diagonal entries where $a'_{i,i}, a'_{i+1,i}$. From here, we can to our $AU = F$ compuattion, where

$$u_m = (b'_m / a'_m m)$$

$$u_{m-1} = (b'_{m=1} - a_{m-1,m-1} a_m)/a_{m-1,m-1}$$
$$u_k = (b'_k - a'_{k,m+1} * u_{m+1})/a'_{k,k}$$

Putting all this shit together is referred to as the **Thomas algorithm**.

Lets say we don't want to waste computation time and memory storing a bunch of zeros. We can avoid this by decomposing the nonzero elements into vectors where

$$A = \begin{cases} a_d = \text{main diagonal} \\ a_{l1} = \text{1st ??? diagonal} \\ a_{s1} = \text{1nd super diagonal} \end{cases}$$

We can implement this in code using

```
# forward elimination
for k=1, m=1
    factor = al1[k] / ad[k]
    ad[k+1][k+1] = ad[k+1][k+1] - factor * as1[k]
    b[k+1] = b[k+1] - factor * b[k]
end

# back substitution
u(m) = k[m] / ad[m]
for k = m-1, 1
    u[k] = (b[k] - as1[k] * u[k+1]) / ad[k])
end
```

January 26, 2024

Example:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} h \frac{\partial u}{\partial x} + t(x, t)$$

simplify to

$$\begin{cases} u''(x) = f(x) \\ u(0) = \alpha \\ u(1) = \beta \end{cases}$$

which gives us

$$\frac{u(x_j - h) - 2u(x_j) + u(x_j + h)}{h^2} \approx f(x_j)$$

we will rerepsent this as

$$\frac{U_{j-2} - 2U_j + U_{j+1}}{h^2} = f(x_j) = F_j$$

We can compose this expression into a matrix with the shape

$$A = \begin{bmatrix} -2 & 1 & \dots & 0 \\ 1 & -2 & \dots & 0 \\ 0 & \dots & 1 & -2 \end{bmatrix}, U = \begin{bmatrix} U_1 \\ U_2 \\ \dots \\ U_N \end{bmatrix}, F = \begin{bmatrix} f_1 - \alpha/h^2 \\ f_2 \\ \dots \\ f_m - \beta/h^2 \end{bmatrix}$$

Using this form allows us to use the *Thomas algorithm.*

$$\Rightarrow U = A^{-1} * F$$

Errors:

$$||\hat{U} - U|| \leq Ch^2$$
$$||\hat{U} - U||_\infty = max_{l \leq j \leq m} |\hat{u}_j - u_j$$
$$||\hat{u} - U||_1 = h * \sum_{j=1}^{m} |\hat{u}_j = u_j|$$
$$[||\hat{u} - U||_2 = (h \sum_{j=1}^{m} (\hat{u}_j - u_j)^2)^{1/2} \rightarrow ||\hat{u} = U||_2^2$$

We talked about jacobi iteration which we covered in substantial detail last semester, so look at those notes; specifically notes back from nov 2023

N-n-n-new shit

$$\begin{aligned} x^{k+1} &= D^{-1}(b - (L + D - D + U)x^{(k)} \\ &= D^{-1}(b - (L + D + U)x^{(k)}) \\ &= D^{-1}(b - Ax^{(k)}) + x^{(k)} \\ &= D^{-1}r^{(k)} + x^{(k)} \\ x^{(k+1)} &= x^{(k)} + D^{-1}r^{(k)} \end{aligned}$$

Du: if

$$Ax = b$$

We derive the residual vector to be

$$r = b - Ax$$

If $r = 0 \rightarrow 0 = b - Ax$

$$\rightarrow Ax = b$$

We have two methods for solving $AU = F$

1. Thomas algoritm (stupid)
2. Jacobi Iteration (also stupid)

jacobi
input A, b, x^0
initialize $r^0 = b - Ax^0$
loop:

```
x_k = D_-1 * r_0 + x_0
    = x_0 + D_-1 * r_0
r_1 = b - A * x_1
test: ||r_1|| -> 0
```

Full matrix mutliplication algorithm
$Ax \to y$

```
for i in range(m):
    y[i] = 0.0
    for j in range(m):
        y[i] = y[i] + a[i][j] * x[j]
    end
end
```

We could prolly make this a bit faster

```
for i in range(m):
    sum = 0.0
    for j in range(i-1, i+2):
        sum = sum + A[i][j] + x[j]
    end
    y[i] = sum
end
```

January 29, 2024

We have a way to approximately solve these systems

$$Au = F \Rightarrow u = A^{-1}F$$

Use back-substituion to find A^{-1} or Jacobi iteration.

Methods:

- Gauss-Elimination with back substitution
- jacobi interation

We want to figure out as $h \to 0$, what do we converge to?

Local Truncation Error
Our finite difference method is

$$\begin{cases} \frac{U_{j-1}-2U_j+U_{j+1}}{h^2} = f(x_j) \\ U_0 = \alpha \\ u_{m+1} = \beta \end{cases}$$

The local truncation error is computed by substituting the exact solution $U(x_j)$ explicitily with taylor series.

$$\begin{aligned} \tau_j = \tau(x_j) &= \frac{1}{h^2}(u(x_{j-1} - 2u(x_j) + u(x_{j+1}) - f(x_j) \\ &= [u''(x_j) + \frac{1}{12}h^2u''''(x_j) + O(h^4)] - f(x_j) \\ &= \frac{1}{12}h^2u''''(x_j) + O(h^4) \\ &\le Ch^2 \end{aligned}$$

We do not know $u''''(x_j)$, but we can assume $u''''(x_j)$ is independent of h. There will be m LTEs (local truncation errors) we are concerned with.

Definition:

$$\tau = A\hat{U} - F = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \dots \\ \tau_3 \end{bmatrix} \Rightarrow A\hat{U} = F + \tau$$

Definition: Global error

$$E = U - \hat{U} \rightarrow ||E|| = ||U - \hat{U}||$$

$$\begin{cases} A\hat{U} - F = \tau \\ AU - F = 0 \end{cases}$$

$$(AU - F) - (A\hat{U} - F) = -\tau$$

$$A(U - \hat{U}) = \tau$$

$$AE = -\tau$$

$$\frac{E_{j+1} - 2E_j + E_{j-1}}{h^2} = -tau_j$$

$$E_0 = 0 \Rightarrow E_{m+1} = 0$$

Given τ_j in $O(h^2)$ we want $\underline{||E|| \le Ch^2}$.
We can write a continous analogue to the ODE:

$$\begin{cases} e''(x) = -\tau(x) \\ e(x) = 0 \\ e(1) = 0 \end{cases} \quad \Rightarrow \tau(x) \approx \frac{1}{12}h^2u''''(x)$$

$$e'' = \int_0^1 \frac{1}{12}h^2u''''(x)dx \Rightarrow e(x) = \frac{2}{12}h^2u''$$

Stability

given

$$A^hE^h = -\tau^h \quad \text{where } h \text{ is the width of the mesh}$$

$$\begin{aligned} E^h &= -(A^h)^{-1}\tau^h \\ ||E^h|| &= ||(A^h)^{-1}\tau^h \\ &\le ||(A^h)^{-1}||||\tau^h|| \rightarrow ||E^h|| \le ||(A^h)^{-1}||||\tau_j|| \end{aligned}$$

Definition: Suppose a finite difference method for a BVP give a sequence of matrix equations of the form $A^hU^h = F^h$, where h is the mesh width. We say the method is stable if $(A^h)^{-1}$ exists for all h sufficiently small $(h < h_0)$ and a constant exists independent of h such that

$$||(A^h)^{-1}|| \le C$$

If (LTE $\rightarrow 0$ as $h \rightarrow 0$) + stability $\Rightarrow$ Convergence.

Matrix Norms.
Defintition: A norm on a vector space is a function that satisifes three properties

1. $||x|| > 0$ for all x in the V.S.
2. $||\alpha x|| = |\alpha|||x||$ for all x and $\alpha \in \mathbb{R}$
3. $||x|g|| \le ||x|| + ||g||$
4. $||AB|| \le ||A||||B||$

If the norm satisfies (4), the norm is said to be a consistent norm.
Example: $||(A^h)^{-1}\tau|| \leq ||(A^h)^{-1}||||\tau||$ We will want individual norms:

$$||A||_1 = max_{x \in \mathbb{R}_m} \frac{||Ax||}{AxU}$$

$$||A||_p = max \frac{||A_x||_p}{||x||_p}$$

$$||A||_\infty = max_{x \in \mathbb{R}_m} \frac{||Ax||_\infty}{||x||_\infty}$$