First Milestone - Tools and Techniques

Kenneth L Meyer, Jenil Shah, Rodrigo Gonzalez

October 19th, 2023

1 Governing equations, nomenclature, etc

The goal of this project is to use finite-difference schemes to solve the heat equation in 1D and 2D space. In 1D, we are solving

$$\begin{cases} -k\frac{d^2}{dx^2}T(x) = q(x) \text{ in } \Omega \\ T(x) = T_b(x) \text{ on } \partial\Omega \end{cases}$$

and in 2D,

$$\begin{cases} -k\nabla^2 T(x,y) = q(x,y) \text{ in } \Omega \\ T(x,y) = T_b(x,y) \text{ on } \partial\Omega \end{cases}$$

where $\Omega = [0,1]^d$ is the physical and computational domain in both cases. The coefficient k is a constant and as shown above, the boundary conditions are Dirichlet conditions.

1.1 Manufactured solution: 1D case.

Let the manufactured solution be:

$$T(x) = \sin(2\pi x)$$

Then it solves the following equation

$$\begin{cases} -k\frac{d^2}{dx^2}T(x) = \sin(2\pi x) =: q(x) \text{ in } \Omega \\ T(x) = 0 \text{ on } \partial\Omega \\ k = 1/(4\pi^2) \end{cases}$$

1.2 Manufactured solution: 2D case.

Let the manufactured solution be:

$$T(x,y) = \sin(2\pi x)\,\sin(2\pi y)$$

Then it solves the following equation

$$\begin{cases} -k\nabla^2 T(x,y) = \sin(2\pi x)\,\sin(2\pi y) =: q(x,y) \text{ in } \Omega \\ T(x,y) = 0 \text{ on } \partial\Omega \\ k = 1/(8\pi^2) \end{cases}$$

These two manufactured solutions will serve to quantify the error produced by the numerical scheme by comparing approximated and the analytical solutions of these two problems.

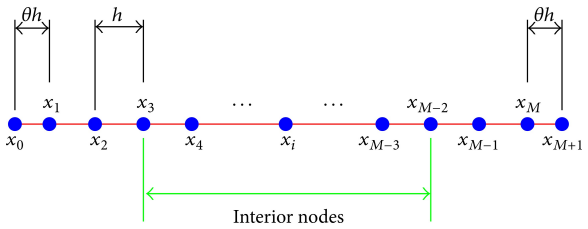

Figure 1: 1D difference representative figure

2 Numerical Methods

2.1 Finite Difference approximations

2.1.1 2^{nd} order approximation

1D case:

$$\frac{T(x+\Delta x)-2T(x)+T(x-\Delta x)}{\Delta x^2}+\mathcal{O}(\Delta x^2)=-\frac{q(x)}{k} \tag{1}$$

2D case:

$$\frac{T(x+\Delta x,y)-2T(x,y)+T(x-\Delta x,y)}{\Delta x^2}+\frac{T(x,y+\Delta y)-2T(x,y)+T(x,y-\Delta y)}{\Delta y^2} + \mathcal{O}(\Delta x^2)=-\frac{q(x)}{k} \tag{2}$$

2.1.2 4^{th} order approximation

1D case:

$$\frac{-T(x+2\Delta x)+16T(x+\Delta x)-30T(x)+16T(x-\Delta x)-T(x-2\Delta x)}{12\Delta x^2} + \mathcal{O}(\Delta x^4)=-\frac{q(x)}{k} \tag{3}$$

2D case:

$$\frac{-T(x+2\Delta x,y)+16T(x+\Delta x,y)-30T(x,y)+16T(x-\Delta x,y)-T(x-2\Delta x,y)}{12\Delta x^2} + \frac{-T(x,y+2\Delta y)+16T(x,y+\Delta y)-30T(x,y)+16T(x,y-\Delta y)-T(x,y-2\Delta y)}{12\Delta y^2} + \mathcal{O}(\Delta x^4)=-\frac{q(x)}{k} \tag{4}$$

The truncation error in the central second order approximation are $\mathcal{O}(h^2)$, whereas in the fourth order approximation, they go as $\mathcal{O}(h^4)$ (where h is $\max(\Delta y, \Delta x)$, W.L.G, we assume it is Δx) The above equations are taken from the linked PDF[1].

2.2 Representative Figures[1][2]

Figure 1 [2] represents the 1D and Figure 2 [3] represents the 2D system. Our interval width is h, and i and j are our choices for the counters in x and y direction respectively. Our system is node-based.

2.3 Linear Systems

2.3.1 Second Order, 1D case

Let

[1] https://www.mech.kth.se/ ardeshir/courses/literature/fd.pdf
[2] https://www.hindawi.com/journals/ana/2016/8376061/fig1/
[3] http://people.eecs.berkeley.edu/ demmel/cs267/lecture17/lecture17.html

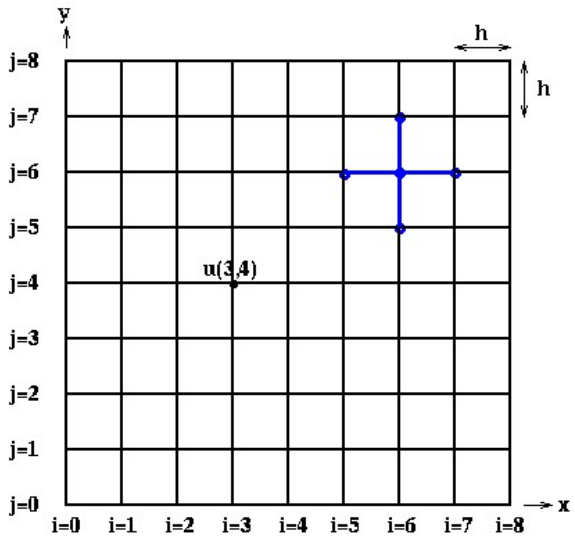

Figure 2: 2D Finite difference representative figure

$$\vec{\alpha} = \begin{bmatrix} T_1 \\ \vdots \\ T_{N-1} \end{bmatrix}$$

where $T_i = T(x_i)$ corresponds to the approximated value of T - the unknown - at the point x_i where $x_i = i * h$, and $h = 1/N$ (or $\delta x = \delta y$ to match the previous' section notation) and N is the number of sample points in the domain. Our goal is to find $\vec{\alpha}$.

To do this, we approximate the second derivative with a second order finite difference formula in the previous section to obtain

$$-(k/h^2)(T_{i-1} - 2T_i + T_{i+1}) = q(x_i)$$

Repeating this for $i = 1 \dots N-1$ leads to $N-1$ independent equations that can be vectorized in the following linear system

$$\mathbf{M_2}\vec{\alpha} = \vec{q}$$

where

$$\mathbf{M_2} = -(k/h^2)\ \text{tridiag}(1, -2, 1)$$

That is, $\mathbf{M_2}$ is the matrix that contains a -2 on the diagonal, and 1 on the first upper and sub diagonal. Moreover, we define

$$\vec{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_{N-1} \end{bmatrix}$$

where $q_i = q(x_i)$. Note that in this case each row of the matrix has $N-1$ entries, but on the interior of the matrix only 3 are non-zero.

2.3.2 Fourth Order, 1D case

Allowing $\vec{\alpha}$ and $\vec{q}$ to be defined as before, we arrive at a similar system. However, in this case we have

$$\mathbf{M_4} = -(k/12h^2)\ \text{sixdiag}(-1, 16, -30, 16, -1)$$

Therefore, in each interior row there are only 6 non-zero entries.

2.3.3 Second Order, 2D case

Let

$$\vec{\alpha_i} = \begin{bmatrix} T(x_i, y_1) \\ \vdots \\ T(x_i, y_{N-1}) \end{bmatrix}$$

As before, we let $x_i = y_i = ih$ with $h = 1/N$ and N is the number of sample points in the domain. These is a vector containing all the y points associated with the x_i point in the domain. Then, applying the finite difference formula, we obtain

$$\left(\frac{1}{h^2}\begin{bmatrix}\mathbf{I} & -2\mathbf{I} & \mathbf{I}\end{bmatrix} + \left(\begin{bmatrix}\mathbf{0} & \mathbf{M_2} & \mathbf{0}\end{bmatrix}\right)\begin{bmatrix}\vec{\alpha}_{i-1} \\ \vec{\alpha}_i \\ \vec{\alpha}_{i+1}\end{bmatrix} = \vec{q_i}$$

Where $\mathbf{I}$ is the $(N-1)$ identity matrix, $\mathbf{M_2}$ is defined as before, and

$$\vec{q_i} = \begin{bmatrix} q(x_i, y_1) \\ \vdots \\ q(x_i, y_{N-1}) \end{bmatrix}$$

Repeating for each $i = 1, \ldots, N_1$ we get a system that can be conveniently written with Kronecker products as follows:

$$\big(\mathbf{M_2} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{M_2}\big)\vec{\alpha} = \vec{q}$$

where we solve for

$$\vec{\alpha} = \begin{bmatrix} \vec{\alpha}_1 \\ \vdots \\ \vec{\alpha}_{N-1} \end{bmatrix}$$

and $\vec{q}$ is defined in a similar fashion.

2.3.4 Fourth Order, 2D case

Allowing $\vec{\alpha}$ and $\vec{q}$ to be defined as before, we arrive at a similar system. However, in this case we use M_4. In other words, we solve

$$\big(\mathbf{M_4} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{M_4}\big)\vec{\alpha} = \vec{q}$$

2.4 Iterative Solution Mechanisms

2.4.1 Jacobi Method

The jacobi method can be used to solve a general linear system $Ax = b$ where $A \in \mathbb{R}^n$ is our coefficient matrix, $b \in \mathbb{R}$, and $x \in \mathbb{R}^n$ is our vector of unknowns. With our linear system, $\mathbf{M}\vec{\alpha} = \vec{q}$, these correspond to $A, x,$ and b. We will use an initial guess or the 0 vector for `x_0` i.e. $x^{(0)}$ in both Jacobi and Gauss-Seidel (in the next section)

The jacobi method assumes that the system has a unique solution and that A has no zeros on its main diagonal; $a_{ii} \neq 0 \, \forall \, i = 1, .., n$.

$$x_{k+1} = D^{-1}(E + F) * x_k + D^{-1} * b$$

is the jacobi method in vector form [1].

A sample implementation of the Jacobi Method, in psuedocode, is shown below:

```
Given: A, b, tolerance tol, and max its N, and initial guess x_0
x_k = x_0
k = 1
while (k ≤ N)
    x_k_1 = D^-1(E + F)x_k + D^-1b
    if ||x_k_1 - x_k|| < tol
        break
    x_k = x_k_1
return x_k_1
```

Listing 1: Jacobi Method Psuedocode [1]

where D is the diagonal of A, and $-E$ and $-F$ are its strictly lower and upper diagonals, respectively.

2.4.2 Gauss-Seidel

Gauss-Seidel is a similar iterative solution scheme. However, it updates the approximate solution immediately after the new component is determined.

The vector form of the Gauss-Seidel method is

$$x_{k+1} = (D-E)^{-1}Fx_k + (D-E)^{-1}b$$

and pseudocode for the Gauss-Seidel method is found below:

```
Given: A, b, tolerance tol, and max its N, and initial guess x_0
x_k = x_0
k = 1
while (k ≤ N)
    x_k_1 = (D − E)^−1 F*x_k + (D − E)^−1 *b
    if ||x_k_1 − x_k|| < tol
        break
    x_k = x_k_1
return x_k_1
```

Listing 2: Gauss-Seidel Psuedocode [1]

Note that * denotes matrix-vector multiplication.

2.5 Memory Requirements

If our matrices were dense, both Jacobi and Gauss-Seidel would have memory requirements of $\mathcal{O}(n^2)$ as $A \in \mathbb{R}^{nxn}$. However, the matrices derived from the finite-difference schemes will be sparse in nature. For the Jacobi Method, $\mathcal{O}(nm + 3n)$ memory will be used (nm is the storage of D^{-1}, E, F matrices, and $3n$ to store $x^{(k)}$, $x^{(k+1)}$, and b). For Gauss-Seidel, $\mathcal{O}(\frac{1}{2}n^2 + \frac{1}{2}mn + 3n)$ memory will be used to store $(D-E)^{-1}, F$, and $x^{(k)}, x^{(k+1)}$, and b respectively. Note that m is the maximum number of non-zero entries in a row of A, and a sparse structure is lost when computing $(D-E)^{-1}$.

References

[1] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second. Other Titles in Applied Mathematics. SIAM, 2003. ISBN: 978-0-89871-534-7. DOI: 10.1137/1.9780898718003. URL: http://www-users.cs.umn.edu/%5C~%7B%7Dsaad/IterMethBook%5C_2ndEd.pdf.