Lesson 4.1 Numerical 1-D Conduction with Matrices [ME516 F19 4.1]

* Show Blackboard system & HW#1

We initially presented content in this class through the Engineering Equation Solver (EES) software program. This was done because EES is the eesyist solver there is. It is an implicit equation solver ⇒ show EES implicitly solving a system of equations.

Notice that the order of variables & equations, or whether linear or non-linear is irrelevent ⇒ I can program the equations exactly as ~~the~~ derived with minimal algebra. EES does this via implementation of Jacobi's method.

Jacobi's method builds a system of matrices $Ax=b$

where

$$A=\begin{bmatrix} a_{11} & a_{22} & \cdots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \\ a_{n1} & \cdots & & \end{bmatrix},\quad x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},\ \&\ b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

A is then decomposed into diagonal & non-diagonal elements

$$A=D+R \text{ where } D=\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix},\ \&\ R=\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}$$

& the solution is then obtained by iteration for

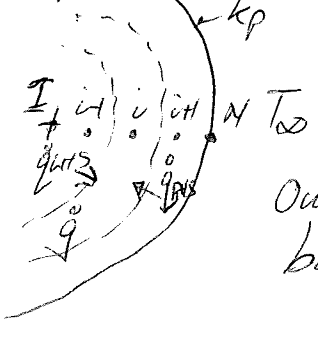
$$x^{(k+1)}=D^{-1}(b-Rx^{(k)}) \text{ or } x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j\neq i} a_{ij}x_j^{(k)}\right)$$

4.2 ME 576 F19

Because of the way EES parses & builds these mattricees the order is irrelevant However considerable ~~time~~ computational time is required to iterate continuously on variables that may have minimal change. So you sacrifice speed for ease with EES, With numerical heat transfer analysees we'll often push EES to the limits. This is where more formal programming languages like Matlab or C+ have an advantage.

Numerical Solutions in Matrix Format

Matlab is an explicit equation solver ⇒ everything on one side of an equation must be known in advance & must equal a single unknown on the other side of the equation. No iteration required. But it requires some additional work on our part. Let's reconsider the Hay Bale problem from last time:



Where the nodes were spaced $r_i = \frac{(i-1)}{(N-1)} R_{Bale}$ & $\Delta r = \frac{R_{Bale}}{(N-1)}$

Our energy balance was $\dot{q}_{LHS} + \dot{q}_{RHS} + \dot{g} = 0$ Where $\dot{g} = \dot{g}''' 2\pi r_i L \Delta r$,

$$\dot{q}_{LHS} = \frac{2\pi L k}{\Delta r}\left(r_i - \frac{\Delta r}{2}\right)(T_{i-1} - T_i) \;\&\; \dot{q}_{RHS} = \frac{2\pi L k}{\Delta r}\left(r_i + \frac{\Delta r}{2}\right)(T_{i+1} - T_i)$$

Which are combined into:

$$\frac{2\pi L k}{\Delta r}\left(r_i - \frac{\Delta r}{2}\right)(T_{i-1} - T_i) + \frac{2\pi L k}{\Delta r}\left(r_i + \frac{\Delta r}{2}\right)(T_{i+1} - T_i) + \dot{g}''' 2\pi r_i L \Delta r = 0$$

Similarly Node 1: $2\pi L \left(\frac{\Delta r}{2}\right) \frac{k}{\Delta r}(T_2 - T_1) + \pi L \left(\frac{\Delta r}{2}\right)^2 \dot{g}''' = 0$

& Node N: $\frac{2\pi L k}{\Delta r}\left(r_N - \frac{\Delta r}{2}\right)(T_{N-1} - T_N) + \frac{(T_\infty - T_N)}{R_{cond,p} + R_{conv}} + \pi L r_N \Delta r \dot{g}''' = 0$

This linear system of N equations with N unknowns can be easily placed into matrix format: $\underline{\underline{A}}\,\underline{x} = \underline{b}$ where $\underline{\underline{A}}$ is a matrix, $\underline{x}$ & $\underline{b}$ are vectors

$$\underline{\underline{A}} = \begin{bmatrix} \text{row } 1 = \text{control volume equation } 1 \\ \text{row } 2 = \text{control volume equation } 2 \\ \vdots \\ \text{row } N = \text{control volume equation } N \end{bmatrix}, \quad \underline{x} = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{bmatrix}$$

to do this we need to manipulate our equations

For Node 1

$$\underbrace{T_1}_{\underline{x}_1}\underbrace{\left[-2\pi L\left(\frac{\Delta r}{2}\right)\frac{k}{\Delta r}\right]}_{\underline{\underline{A}}_{1,1}} + \underbrace{T_2}_{\underline{x}_2}\underbrace{\left[2\pi L\left(\frac{\Delta r}{2}\right)\frac{k}{\Delta r}\right]}_{\underline{\underline{A}}_{1,2}} = \underbrace{-\dot{q}'''2\pi r_i L\Delta r}_{\underline{b}_1}$$

For Internal Nodes

$$\underbrace{T_i}_{\underline{x}_i}\underbrace{\left[-\frac{2\pi Lk}{\Delta r}\left(r_i-\frac{\Delta r}{2}\right)-\frac{2\pi Lk}{\Delta r}\left(r_i+\frac{\Delta r}{2}\right)\right]}_{\underline{\underline{A}}_{i,i}} + \underbrace{T_{i-1}}_{\underline{x}_{i-1}}\underbrace{\left[\frac{2\pi Lk}{\Delta r}\left(r_i-\frac{\Delta r}{2}\right)\right]}_{\underline{\underline{A}}_{i,i-1}}$$

$$+\underbrace{T_{i+1}}_{\underline{x}_{i+1}}\underbrace{\left[\frac{2\pi Lk}{\Delta r}\left(r_i+\frac{\Delta r}{2}\right)\right]}_{\underline{\underline{A}}_{i,i+1}} = \underbrace{-\dot{q}'''2\pi r_i L\Delta r}_{b_i}$$

For Node N

$$\underbrace{T_N}_{\underline{x}_N}\underbrace{\left[-\frac{2\pi Lk}{\Delta r}\left(r_N-\frac{\Delta r}{2}\right)-\frac{1}{R_{cond_p}+R_{conv}}\right]}_{\underline{\underline{A}}_{N,N}} + \underbrace{T_{N-1}}_{\underline{x}_{N-1}}\underbrace{\left[\frac{2\pi Lk}{\Delta r}\left(r_N-\frac{\Delta r}{2}\right)\right]}_{\underline{\underline{A}}_{N,N-1}} = \underbrace{-\frac{T_\infty}{R_{cond_p}+R_{conv}}-\pi L r_N\Delta r\dot{q}'''}_{b_N}$$

Where the solution is then

$$\underline{x} = \underline{\underline{A}}^{-1}\underline{b}$$

Show example code

4.4 ME576 F19

While this code may be fast, it is not quite as flexible as EES. Temperature dependent properties are non-linear, which is easy for EES but harder for explicit solvers like Matlab because the equations can no-longer be solved without iteration. Here's the process:

Step 1: Assume a temperature distribution $\hat{T}_i$ for $i=1...N$

Step 2: Setup $\underline{\underline{A}}$ $\underline{b}$ using $\hat{T}_i$ to solve for an coefficients $f(T)$

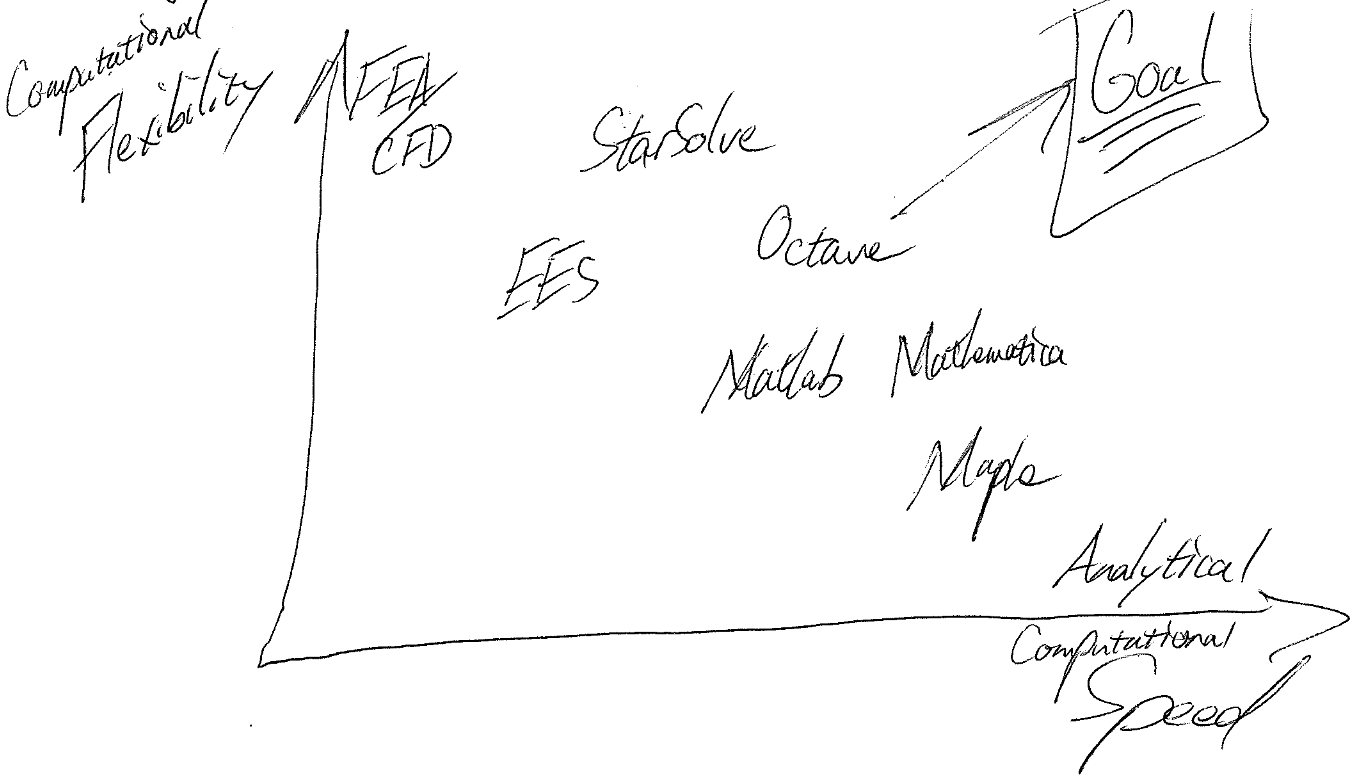
Step 3: Solve $\underline{\underline{A}}\,\underline{X}=\underline{b}$ in order to predict T_i for $i=1...N$

Step 4: Compute convergence error

$$err=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{T}_i-T_i\right)^2}$$

Step 5: Iterate steps 2→4 until convergence tolerance is met

☆ In general you'll have to select your modeling ~~software~~ approach considering:


Computational Flexibility
FEA
CFD
StarSolve
Goal
Octave
EES
Matlab
Mathematica
Maple
Analytical
Computational Speed