# ChEn 2450

# Homework 3

# Thomas Algorithm

Later in the semester we will discuss a discrete solution to the second-order differential equation:

$$\frac{\mathrm{d}^2T}{\mathrm{d}x^2} = -\frac{S}{k}, \tag{1}$$

with boundary conditions $T(x=0) = T_o$ and $T(x=L) = T_L$ and a source term $S(x) = \exp\left(-\left(\frac{x-a}{\sigma}\right)^2\right)$. This equation describes the temperature profile through a rod being held at fixed temperature at both ends and heated somewhere along its length.

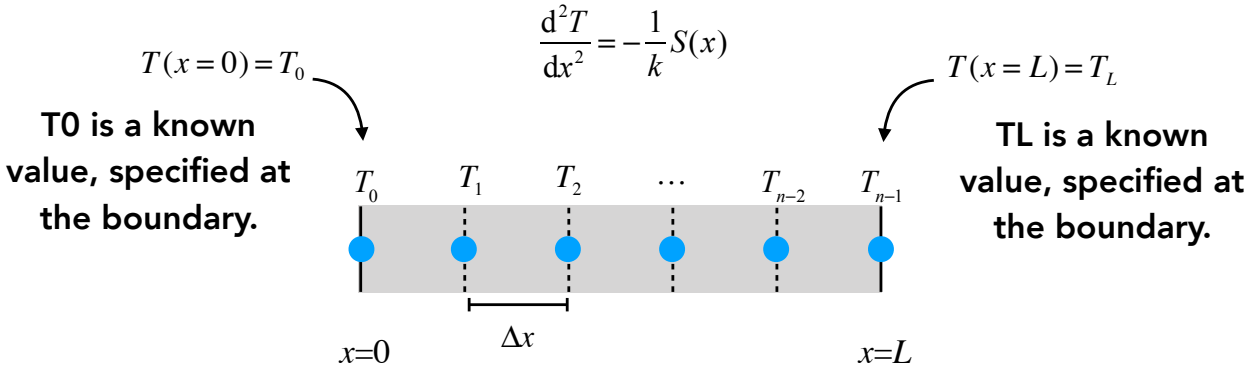


Figure 1: Schematic of heated rod with location of approximate values for the temperature. Blue disks indicate points where we solve for $T$.

A discrete (*approximate*) solution for $n$ points may be written as a system of linear equations for the discrete values of temperature along the rod:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & & 0 \\ 1 & -2 & 1 & 0 & \cdots & & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} T_0 \\ T_1 \\ T_2 \\ \vdots \\ T_{n-3} \\ T_{n-2} \\ T_{n-1} \end{pmatrix} = \begin{pmatrix} T_o \\ -\Delta x^2 \frac{S_1}{\lambda} \\ -\Delta x^2 \frac{S_2}{\lambda} \\ \vdots \\ -\Delta x^2 \frac{S_{n-3}}{\lambda} \\ -\Delta x^2 \frac{S_{n-2}}{\lambda} \\ T_L \end{pmatrix}. \tag{2}$$

Assume $L = 1$ m, $T_o = 300$ K, $T_L = 310$ K, $\lambda = 10^{-5}\,\mathrm{W/m{\cdot}K}$, $a = 2L/3$ and $\sigma = 0.1$ m. Note that units of $S$ are $\mathrm{W/m^3}$.

# Problem 1 (40 pts)

The purpose of this problem is to get you familiar with the Thomas algorithm. This problem should be done **by hand** (show your work). You should typeset these in your Jupyter notebook using markdown and LaTeX for your equations. If you do not know LaTeX you can use an online LaTeX equation editor such as http://www.hostmath.com or download and typset with LyX and then copy the latex math formulas to your jupyter notebook.

1. (8 pts) Define $\Delta x$ and $x_i$ in terms of $i$, $n$ and $L$. *Hint*: carefully study Figure 2.

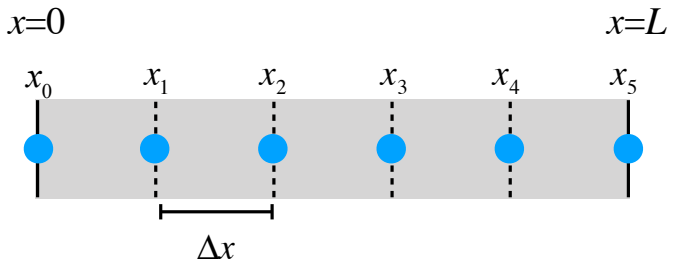


Figure 2: Schematic of domain. Circles indicate points where we solve for $T$.

2. (32 pts) Write down the system of equations for $n = 4$ and then solve that using the Thomas algorithm.

# Problem 2 (60 pts)

The objective of this problem is to compare the cost of using built-in linear solvers in Python to that of using the Thomas algorithm for a tridiagonal system of equations.

Generate a plot showing the time ($t_s$) it takes to solve the system of linear equations from Problem 1 as a function of the number of equations (e.g. grid points) $n$ for $n = [10, 20, 40, 100, 1000, 2000, 3000]$ using:

1. (20 pts) The Thomas algorithm. You should create a function called `thomas` that implements the Thomas algorithm and submit that as part of your solution. (Note that you can test your implementation of the Thomas algorithm by comparing it to Python's `numpy.linalg.solve`).

2. (20 pts) Python's built-in `solve` command (`numpy.linalg.solve`).

3. (20 pts) The inverse (use Python's `inv` command - `Ainv = numpy.linalg.inv(A)`) and then compute your solution as $\mathbf{T} = \mathbf{A}^{-1}\mathbf{b}$. To perform a matrix vector multiplication in Python use the `@` symbol, i.e. `T = Ainv@b`. You should include this last step in the timing for this case.

Prepare a brief report that summarizes your findings.

To measure the timings in Python, you can use:

```
import time
tic = time.time()
your code goes here...  for example call Thomas algorithm
toc = time.time()
ts = toc - tic
```

Your plot should be on a log-log scale with the x-axis corresponding to the number of equations and the y-axis corresponding to the time-to-solution. You should also label axes and add a legend to your plot. Here's an example of creating a plot in Python:

```
%matplotlib inline
import matplotlib.pyplot as plt
n = [10,20,40,100,1e3,2e3,3e3] # x axis data
# you will obtain your timings from your solution of the linear system
# the following values are arbitrary just to illustrate how to make a plot
t0 = [0.1,0.01,0.05,0.1,0.2,0.4,0.6] # timings for thomas
t1 = [0.2,0.01,0.06,0.15,0.33,0.45,0.68] # timings for using Ainv
plt.xlabel('# of equations') # set x label
plt.ylabel('time to solution (s)') # set y label
plt.loglog(n,t0,'k-*',label="Thomas")
plt.loglog(n,t1,'r-o',label="Inverse")
plt.legend() # display legend
plt.grid(True,which="minor") # turn on minor grid axes
```