Matrix Condition & Introducing the The Temperature Problem

CS 111: Introduction to Computational Science

Spring 2019 Lecture #6

Ziad Matni, Ph.D.

Administrative

- Homework #3 due next Monday

Lecture Outline

- More on Numerical Stability
  - The Matrix Condition Number
- The Temperature Problem and Sparse Matrices

Numerical Stability

- Forward and backward error are related by the **condition number**
  - We'll examine its calculation another time
  - You can use **numpy** to calculate it!

- Large condition number = the matrix is not numerically stable
  - Or, "ill-conditioned"
- Small condition number (closer to 1) = the matrix **is** numerically stable
  - Or, "well-conditioned"

- Usually, symmetrical and/or is normal matrices are "well-conditioned"

Condition Number

The condition number of matrix **M** is defined as:

$$||M|| \times ||M^{-1}||$$

i.e. norm of M multiplied by the norm of the inverse of M

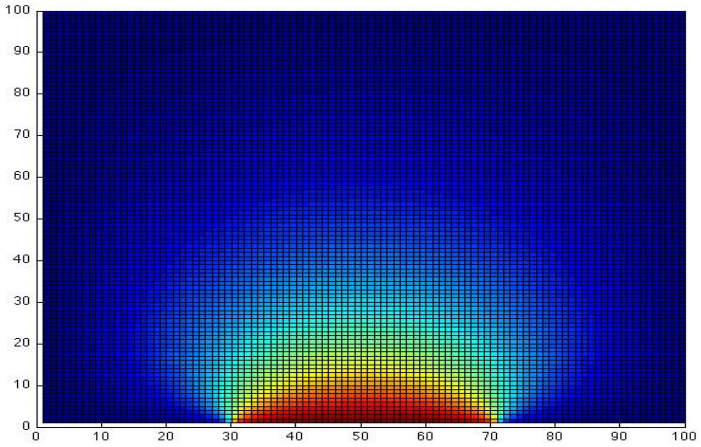
In Python, you can use the `.cond()` function in `linalg`:

```
numpy.lingalg.cond(M, 'fro')
```

Where **'fro'** indicates the Frobenius Norm, that is the norm defined as the square root of the sum of the absolute squares of its elements.

The Temperature Problem

- A cabin in the snow
- Wall temperature is 0°, except for a radiator at 100°
- What is the temperature at every point in the interior?



Matni,

The Physics behind the Problem: Poisson's Equation

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y)$$

for $(x,y) \in R = \{ (x,y) \mid a < x < b, \quad c < y < d \}$, and

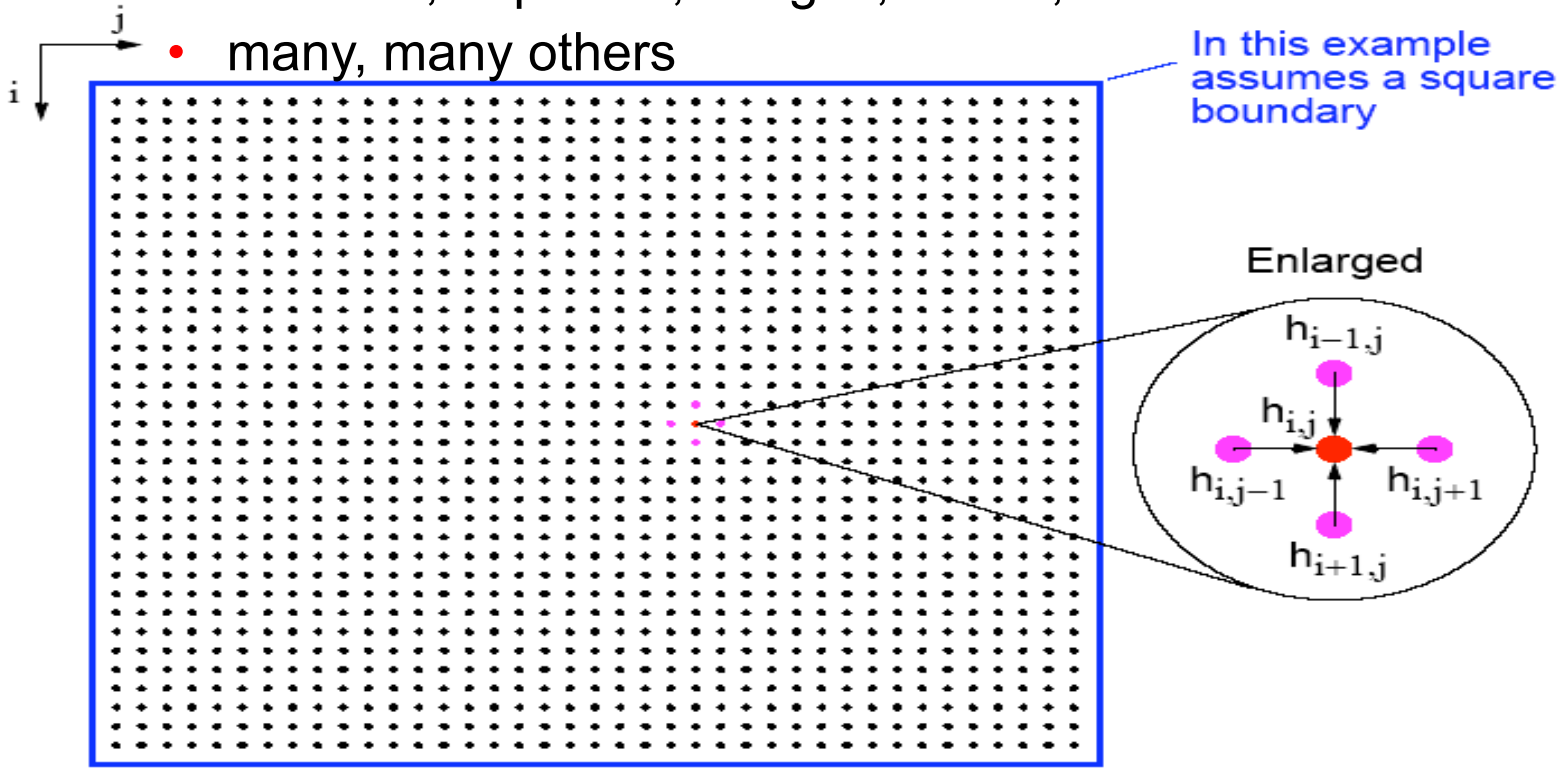
$$u(x,y) = g(x,y)$$

for (x,y) on the boundary of R.

But, this is a continuous math (Calculus) problem – how can we "translate" it into discrete math (Computation)?

Many Physical Models Use Stencil Computations

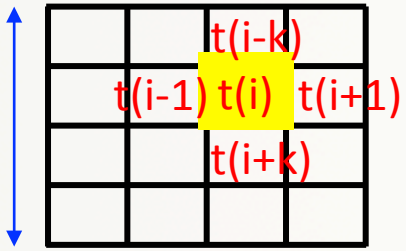
- PDE models of heat, fluids, structures, …
- Weather, airplanes, bridges, bones, …
- many, many others



Model Problem:
Solving Poisson's equation for temperature

- Discrete approximation to Poisson's equation (without going into details):

$k = n^{1/2}$

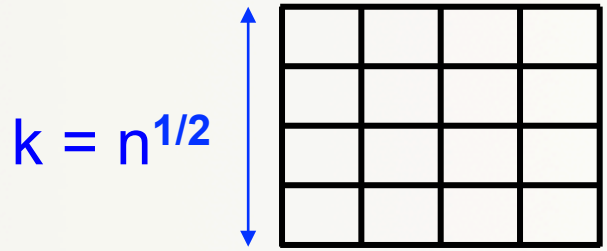


$$t(i) = \frac{1}{4} (t(i-k) + t(i-1) + t(i+1) + t(i+k))$$

- Intuitively:

Temperature at any point is
the average of the temperatures at its surrounding points

Model Problem:
Solving Poisson's equation for temperature



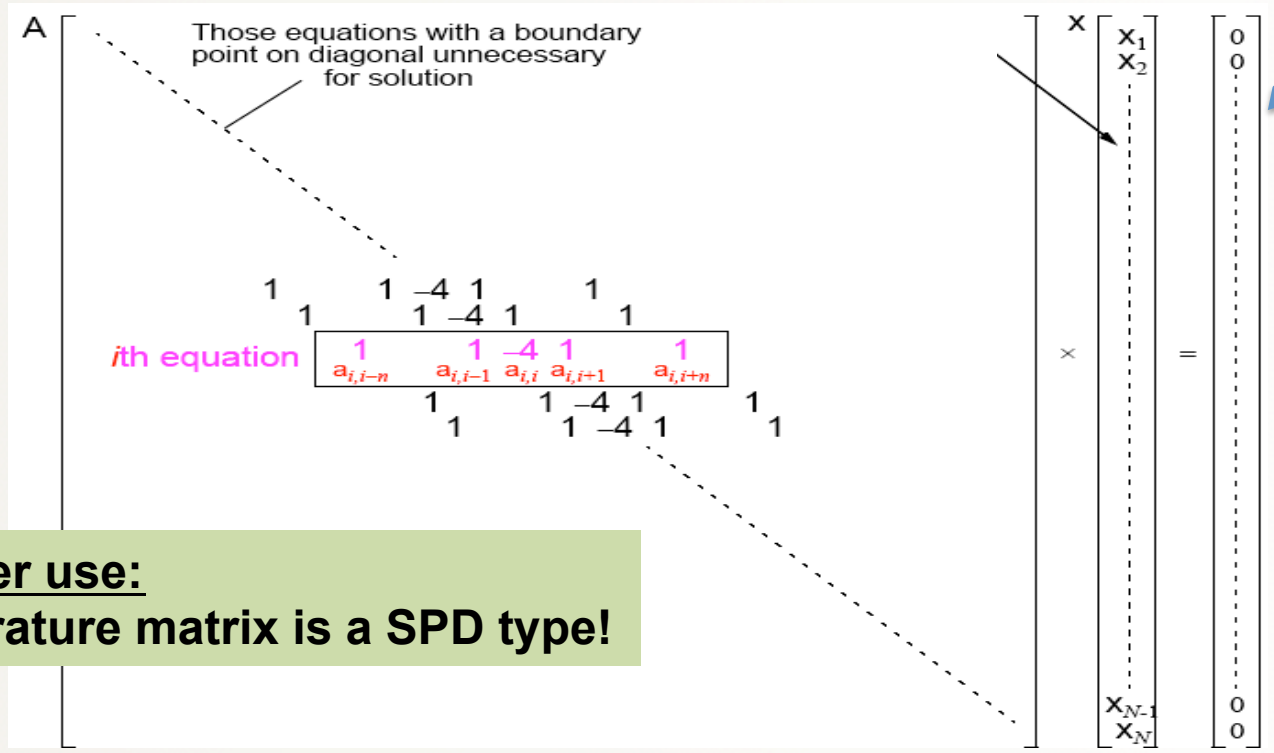
- For each i from 1 to n, **except** on the boundaries:

$$-t(i-k) - t(i-1) + 4*t(i) - t(i+1) - t(i+k) = 0$$

- n equations in n unknowns: A***t** = b
- Each row of A has at most 5 non-zeros (the rest are zeros!)
- In three dimensions, $k = n^{1/3}$ and each row has at most 7 non-zeros

A Stencil Computation Solves a System of Linear Equations

- Solve A**x** = b for **x**
- Matrix A, right-hand side vector b, unknown vector **x**
- A is *sparse*: most of the entries are 0



b is where we would express boundary conditions

Note for later use:
This temperature matrix is a SPD type!

To Python We Go!

- Sparse vs Dense Matrix Representation
- Solving the temperature problem!
- Graphing it too!!

Your To-Dos

- Homework #3 – due **Monday, April 22nd**

</LECTURE>