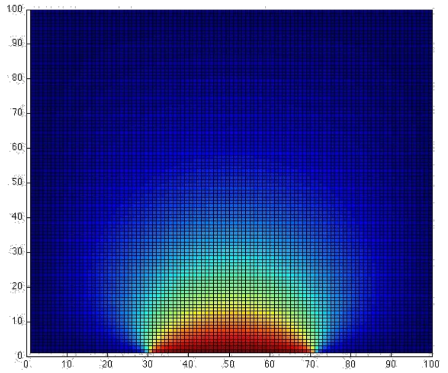


Lecture 07:
The Temperature Problem

CS 111: Intro to Computational Science
Spring 2023

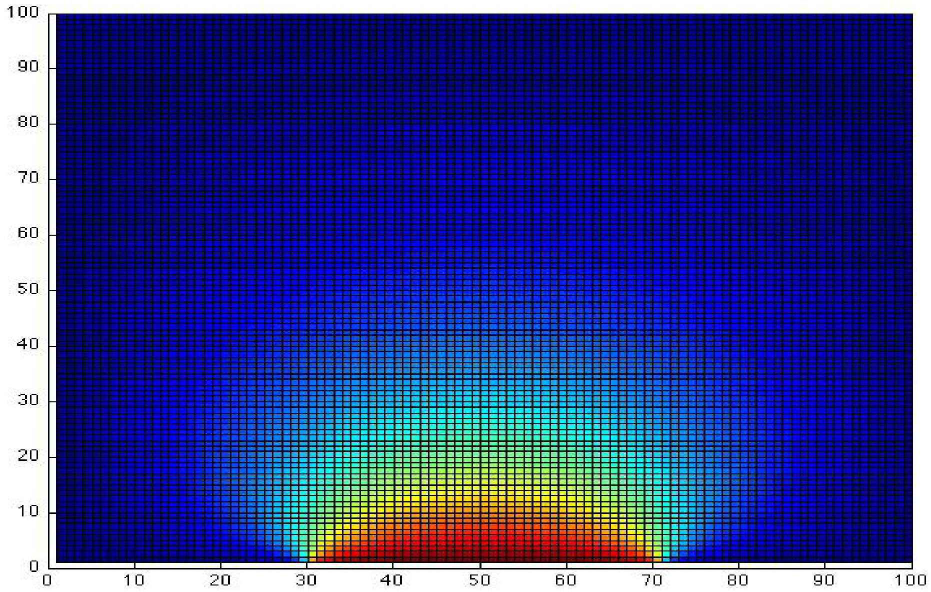
Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

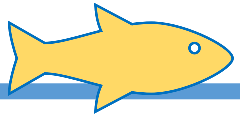
- Turn in current homework
- New homework is out today
  - It's a tougher one, so start it early!
- Preliminary slides for this lecture available!
- Note:
  I'm pushing the continuation of "Numerical Stability" for a future lecture…
- **No quiz this week!**

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?***



The Physics behind the Problem: **Poisson's Equation**



<u>For a 2-dimensional situation:</u>

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

for $(x, y) \in R = \{ (x, y) \mid a < x < b, \ c < y < d \}$, 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* (for Computation)?

Discrete Poisson Equation

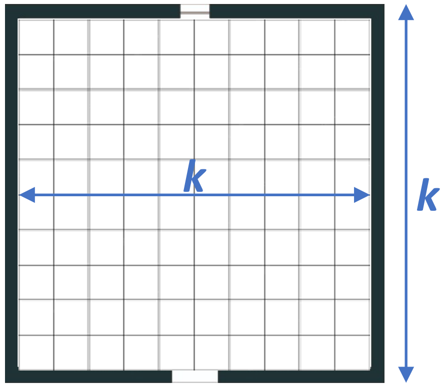
$$(\nabla^2 u)_{ij} = \frac{1}{\Delta x^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij})$$

- This translates to a linear system that we can write as a matrix expression similar to **Ax = b** *(!!! How convenient!!!)*

- *How?... Let's See…*

Discrete Poisson Equation

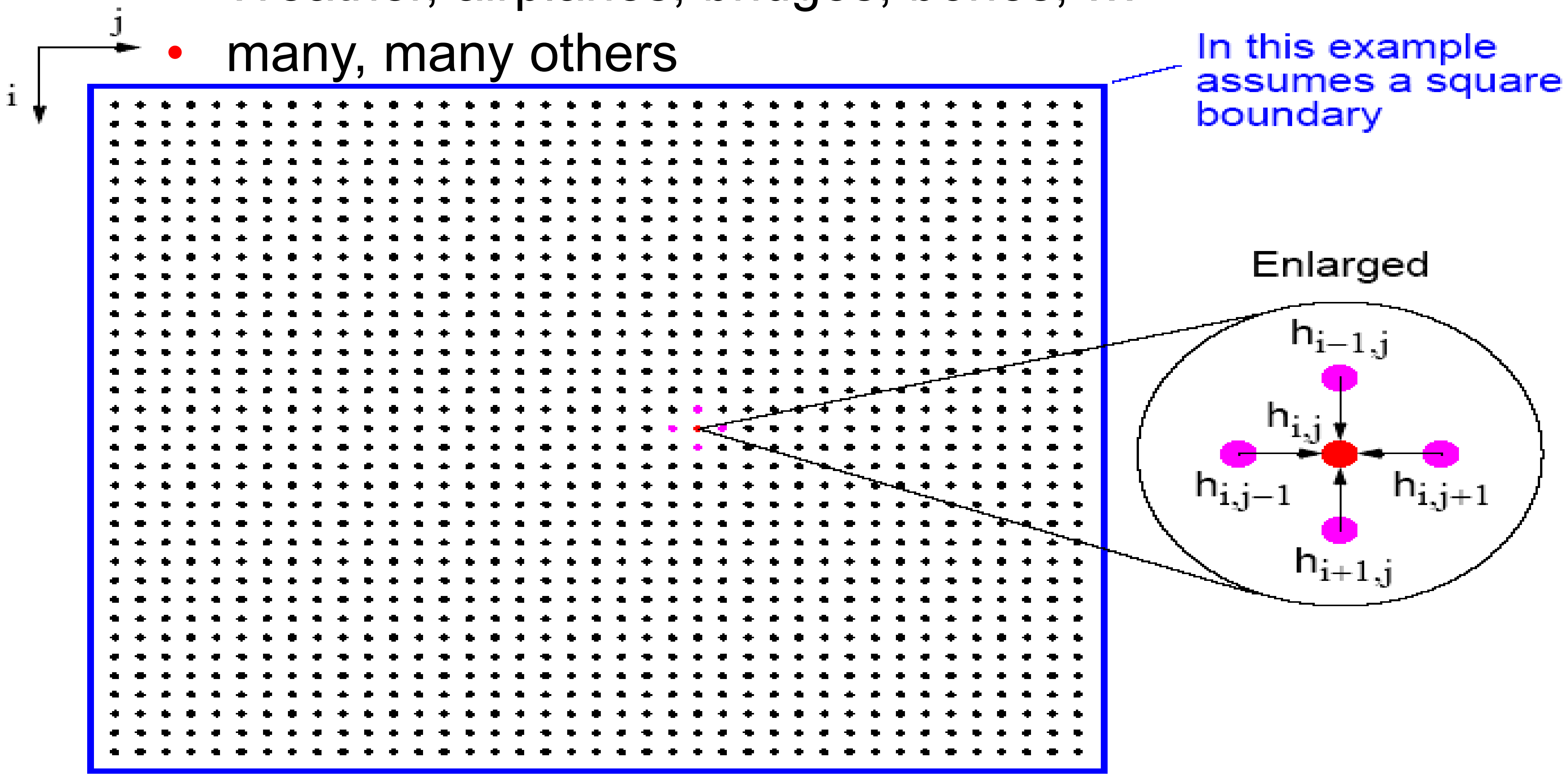
- Think about this in the case of examining a *square* room…



- The room has sides that are **k** units long each
  - We'll divide up the room in **k** discrete steps in each direction
  - Imagine it like a square room broken up into a grid
  - End up with a discrete **"stencil" space**
  - This is NOT a matrix! (… yet…)

Many Physical Models Use *Stencil* Computations

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



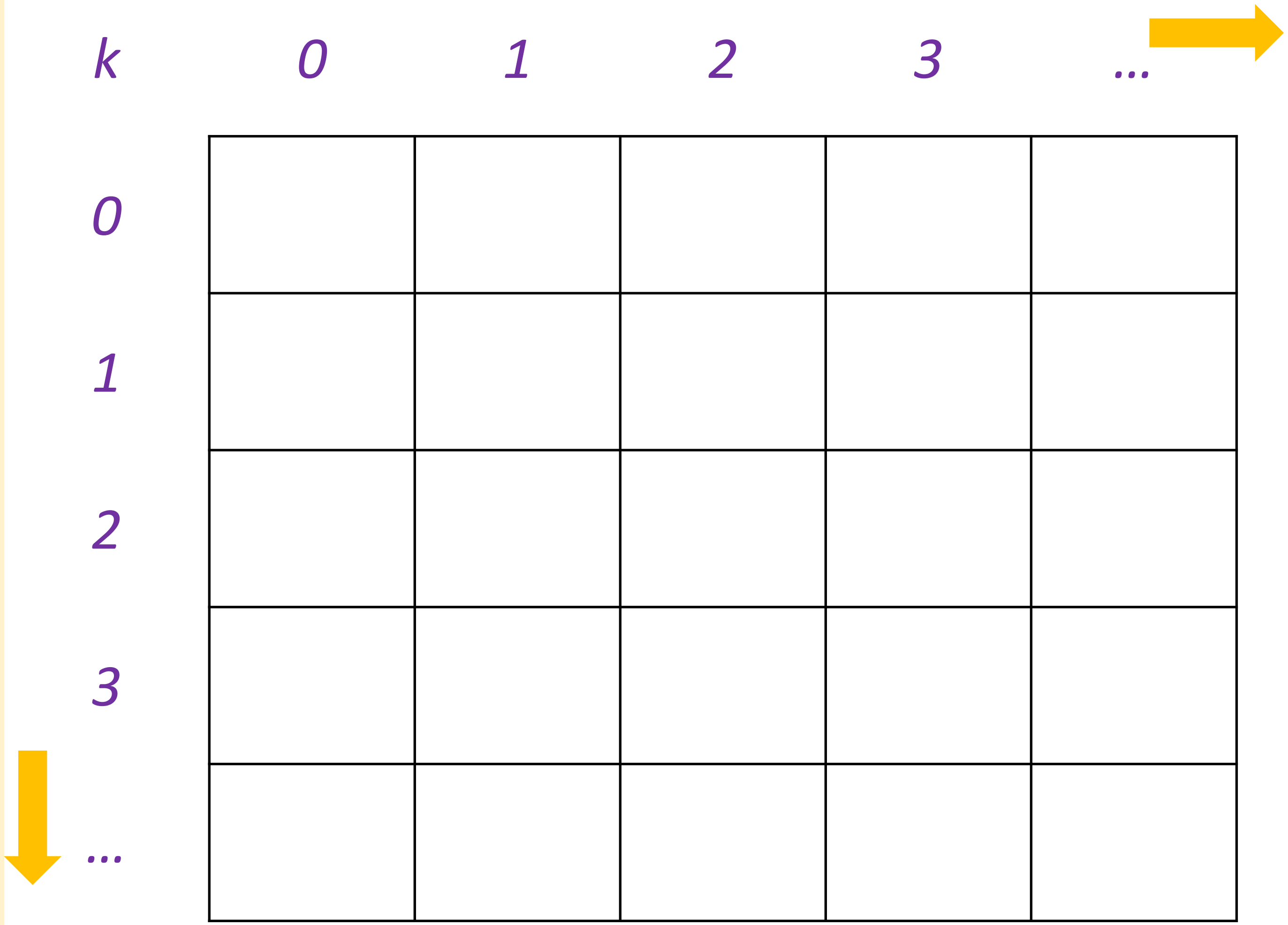
Modeling the Room

Discrete approximation to Poisson's equation: Divide up the room into **k**x**k** spaces or "cells".

Again, we assume the room is square with an area of **n**

So, $\mathbf{k = n^{1/2}}$

Temperature at any point is the <u>average</u> of the temperatures at its surrounding points



Simple Example: A 5x5 Room (i.e. k=5)

<u>Example – Let's look at space number 13</u>:

The temperature in space number 13 (T_{13}) is the average of the temperatures in the surrounding spaces. That is:

$$T_{13} = ¼ (T_8 + T_{12} + T_{14} + T_{18})$$

or

$$4.T_{13} = T_8 + T_{12} + T_{14} + T_{18}$$

or

$$\mathbf{T_8 + T_{12} - 4.T_{13} + T_{14} + T_{18} = 0}$$

k	*0*	*1*	*2*	*3*	*4*
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

Simple Example: A 5x5 Room (i.e. k=5)

Example – Let's look at space number 13:

The temperature in space number 13 (T_{13}) is the average of the temperatures in the surrounding spaces. That is:

$$\mathbf{T_8 + T_{12} - 4.T_{13} + T_{14} + T_{18} = 0}$$

*Note that the area **n** = 25 in this example (i.e. $n = k^2$)*

Now apply this to ***all*** the **n** "cells" and you can get a relationship "heat" map of the room!

You need 1 more thing: a location defined for the "**heat source**". This is akin to an initial condition for a diff. equation problem.

k	*0*	*1*	*2*	*3*	*4*
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

Simple Example: A 5x5 Room

So:

For each ***i*** from 1 to **n**

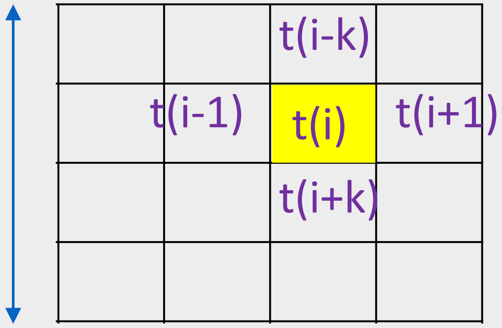
(except on the boundaries), we see that

the *generalized relationship* is:

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

(this is an implementation of the discrete Poisson equation!)

$$k = n^{1/2}$$



k	*0*	*1*	*2*	*3*	*4*
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

REMEMBER: There's a Transformation Going On!

- This **5x5** ***grid***:

k	*0*	*1*	*2*	*3*	*4*
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

Is **NOT** the matrix that

we plug into Python!

It has to be first **transformed** into a **25x25** ***matrix***!
One row for each "cell" on the grid,
One column for each "cell" too!

This is so that every row / column tells me something about the *linear temperature relationship* between every "cell" on the grid!

- The numbers in the matrix will thus be
  the ***coefficients*** of the elements *in the discrete Poisson equation to solve the temperature*
- NOTE: They are **NOT** the temperature values themselves!!

The Transformation! 1

- We are expressing: *mathematical linear relationships*

  i.e. **n**-equations w/ **n**-unknowns

- The ***unknowns*** are the ***temperatures*** in each of the **n** number of "cells"
- These temperatures are all related to all the other cells in the room
  - In actuality, it's not exactly *all* the other cells, just the ones adjacent to the cell in question
- This matrix multiplication can express it:

$$A.t = \begin{bmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n0} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} t_0 \\ \vdots \\ t_n \end{bmatrix}$$

The Transformation! 2

- This matrix multiplication can express it: $A.t = \begin{bmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n0} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} t_0 \\ \vdots \\ t_n \end{bmatrix}$

- We expect this to give us an ***n*** number of equations of the form:

t(i-k) + t(i-1) – 4.t(i) + t(i+1) + t(i+k) = 0

- $\boldsymbol{A.t}$ should give us an **n**x1 vector that's mostly zeros
  - Except where the heat-source is!

- Note also that a LOT of values in matrix ***A*** (a_{ij}) are going to be zeros…!

Using the Transformation for a Solution!!

- What we end up with, then is something like this:

Mostly 0, except for where the heat source is!

$$A.t = \begin{bmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n0} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} t_0 \\ \vdots \\ t_n \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}$$

The coefficients that describe Poisson's Eqs. Many of these will be 0s.

The temperature at each of the ***n*** *cells on the grid*

- Note that this EXACTLY an **Ax = b** situation!!!
  - Are we going to use **`npla.solve()`** ??? Or something else??? This is exciting!!!!

Translate the Room into a Matrix!

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

- Each row of this **nxn** matrix (**A**) will have ***at most*** 5 non-zero elements
  - These represent the cell-in-question + the 4 adjacent cells
  - These are the ones represented in the (red) Poission's equation above…
  - The rest of each row will be filled with **zeros**
  - Note that, if this were in 3-D, **n** would be $\mathbf{k^3}$ and each row will have at most ***7*** non-zeros, etc…

- **This makes for a "sparse matrix"!!!**
  - Def.: a matrix where most of the elements are zero
  - The opposite is called a "**dense matrix**"

A *Very* Simple Example

t1	t2	t3
t4	t5	t6
t7	t8	t9

- Consider a room that's stenciled **3x3** (so: 9 squares in the **grid**)
- So, the **matrix A** will be a 9x9 and *looks like this*:

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} t1 \\ t2 \\ t3 \\ t4 \\ t5 \\ t6 \\ t7 \\ t8 \\ t9 \end{pmatrix} = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

For square **5** in the room, we see that the relationship:
t2 + t4 - 4*t5 + t6 + t8 = 0
is applied by the **At = b** relationship

Same for square **1**:
-4*t1 + t2 + t4 = 100
(note how this suggests that a heat source at 100^{o} is placed next to square 1)

And so on, for each of the other 9 squares in the room.

The vector **t** gives us the temperature in every room!
*So solve for **t**!!*

A *Very* Simple Example

Running **`solve(A, b)`** on this, gives us the 9 values for **`t`**

These are the values in the original physical grid:

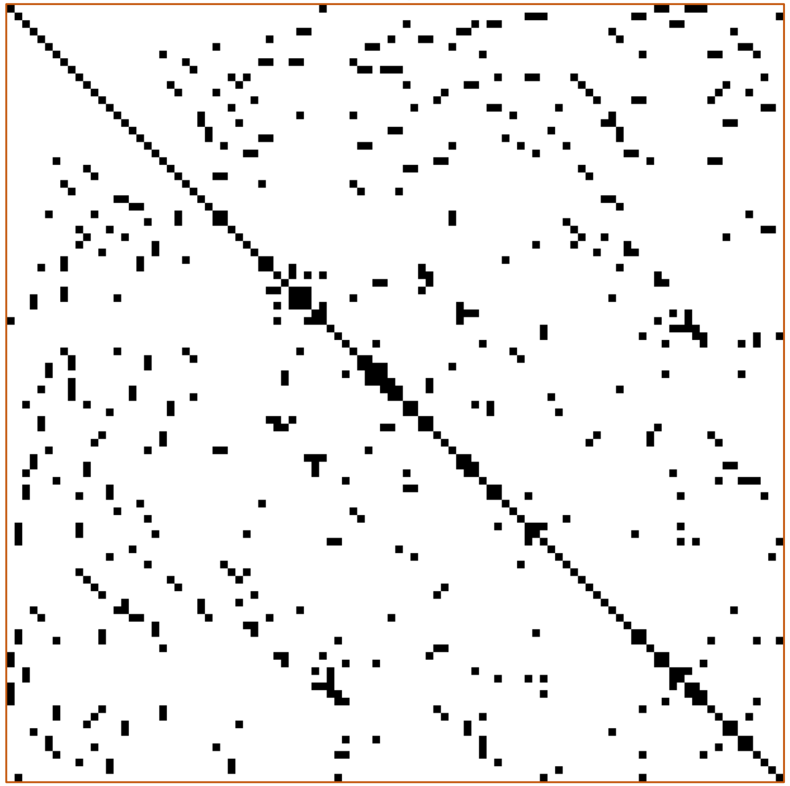
Where the heat source is applied

	100^{o}	*100^{o}*	*0*	
0	39.7^{o}	42.8^{o}	12.9^{o}	*0*
0	16.1^{o}	18.8^{o}	8.9^{o}	*0*
0	5.8^{o}	7.1^{o}	4.0^{o}	*0*
	0	*0*	*0*	

Sparse Matrices

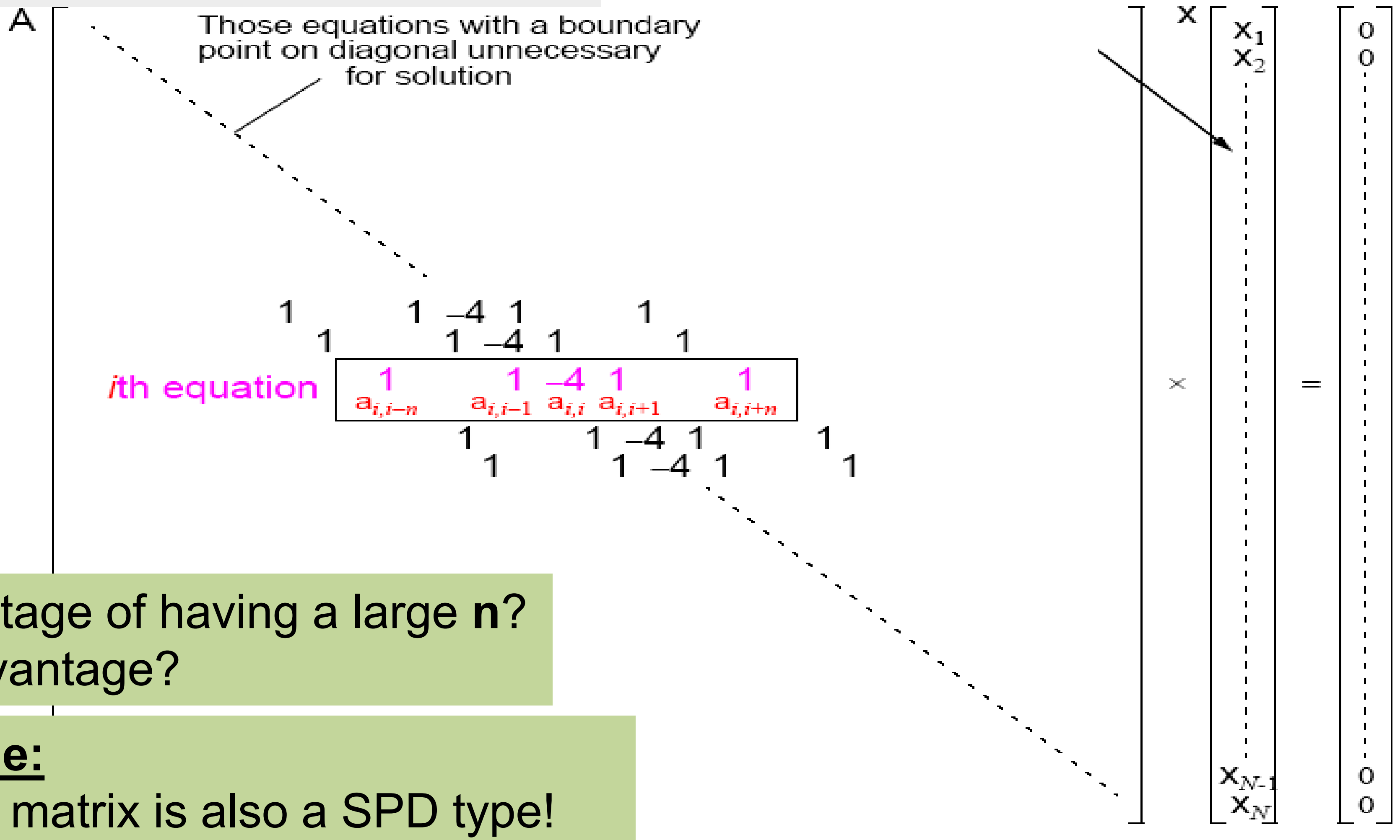
- A **sparse matrix** is a **matrix** which contains *very few non-zero elements*.
- When a **sparse matrix** is represented with a 2-D *array* (i.e. simple 1-to-1 mapping), we waste a lot of computer memory to represent that **matrix**.
- We can represent ***sparse matrices*** in more efficient ways than this!
    - We won't get into *how* this is done, *just how to use them*…

1000x1000 sparse matrix



The Temperature Matrix is a Sparse Matrix…

A is *sparse*: Most of the Entries are 0

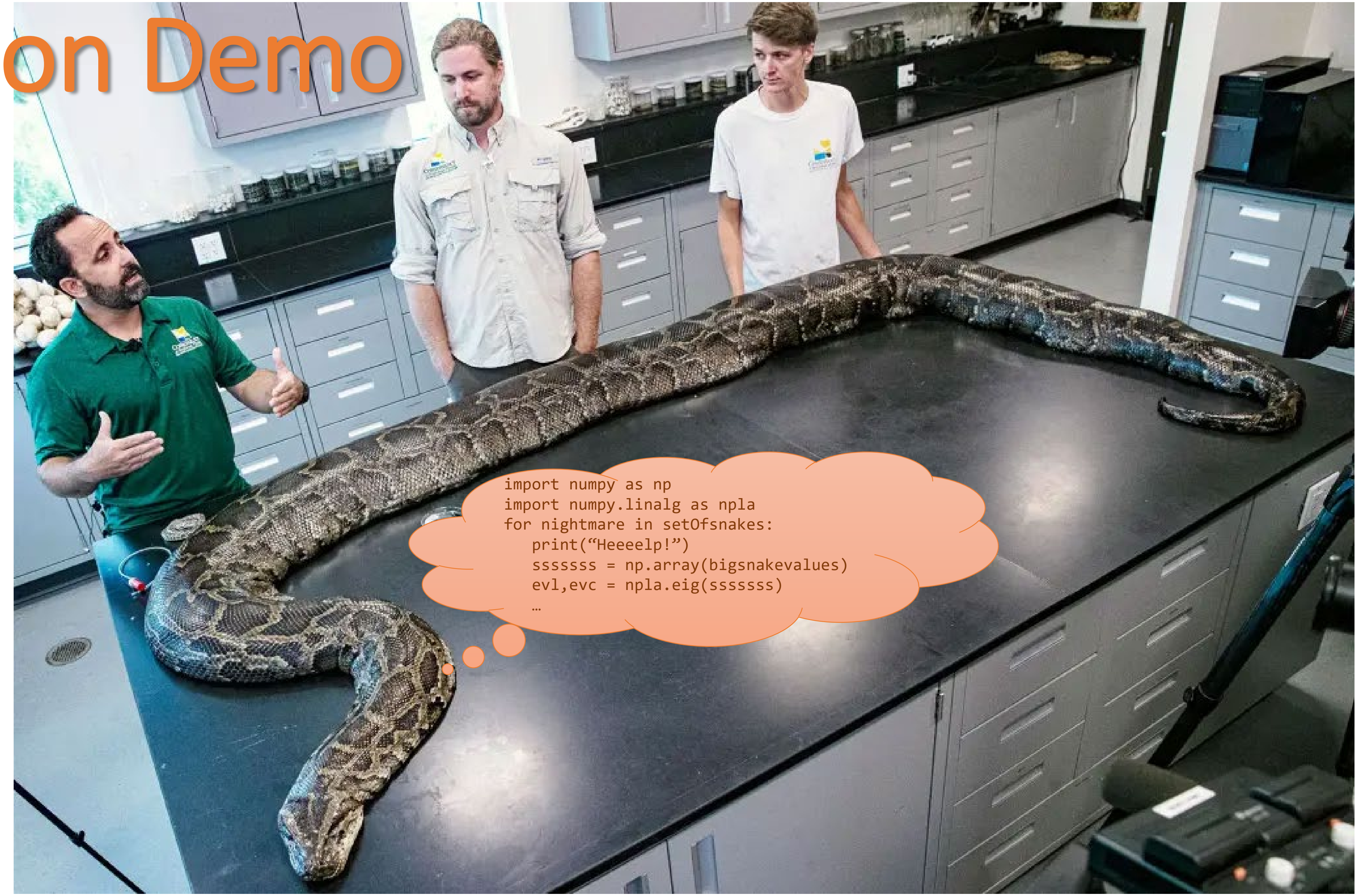


b is where we would express boundary conditions

What's the advantage of having a large **n**? What's the disadvantage?

<u>Note for later use:</u>
This temperature matrix is also a SPD type!

Python Demo



```
import numpy as np
import numpy.linalg as npla
for nightmare in setOfsnakes:
   print(“Heeeelp!”)
   sssssss = np.array(bigsnakevalues)
   evl,evc = npla.eig(sssssss)
   …
```


Credit: The News-Press-USA TODAY NETWORK

Your TO DOs!

- Turn in your homework today!
- Start on your new homework
- Remember: NO QUIZ this week!!!!

</LECTURE>