CS6068 Lecture 8
Applications:
Numerical Methods

- Solving ordinary and partial differential equations
- Finite difference methods (FDM)
- Wave equation: vibrating string problem
- Heat equation: Steady state heat distribution problem

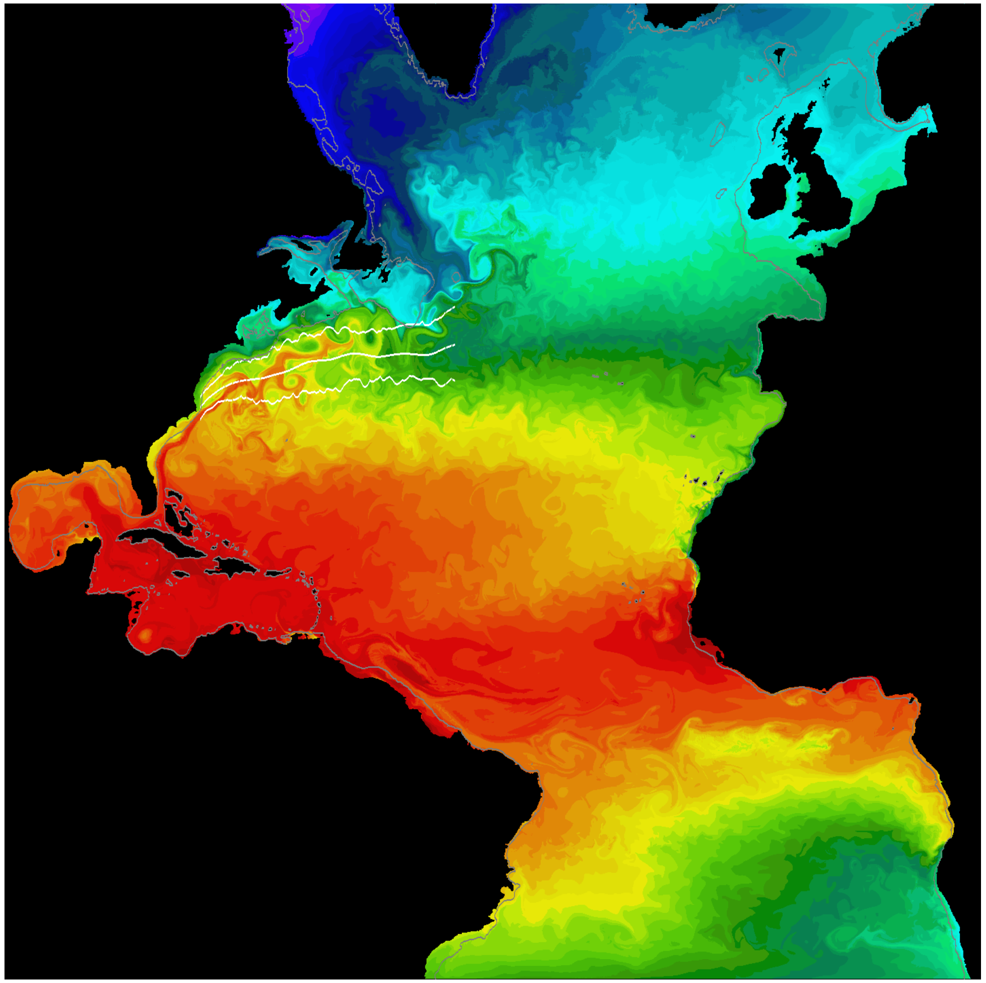
PDEs and Examples of Phenomena Modeled

- Ordinary differential equation: equation containing derivatives of a function of one variable
- Partial differential equation: equation containing derivatives of a function of two or more variables

Models:

- Air flow over an aircraft wing
- Blood circulation in human body
- Water circulation in an ocean
- Bridge deformations as its carries traffic
- Evolution of a thunderstorm
- Oscillations of a skyscraper hit by earthquake
- Strength of a toy
- Financial Markets

Example: Model of Sea Surface Temperature in Atlantic Ocean



…there is a physical problem that is common to many fields, that is very old, and that has not been solved. Nobody in physics has really been able to analyze it mathematically satisfactorily in spite of its importance to the sister sciences. It is the analysis of circulating or turbulent fluids.

- *Richard Feynman's Lectures on Physics*

Courtesy MICOM group
at the Rosenstiel School
of Marine and Atmospheric
Science, University of Miami

Solving PDEs

- Finite element method
- Finite difference method (our focus)
  - Converts PDE into matrix equation
    - Linear system over discrete basis elements
  - Result is usually a sparse matrix
  - Matrix-based algorithms represent matrices explicitly
  - Matrix-free algorithms represent matrix values implicitly (our focus)

Class of Linear Second-order PDEs

- Linear second-order PDEs are of the form

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Eu_x + Fu_y + Gu = H$$

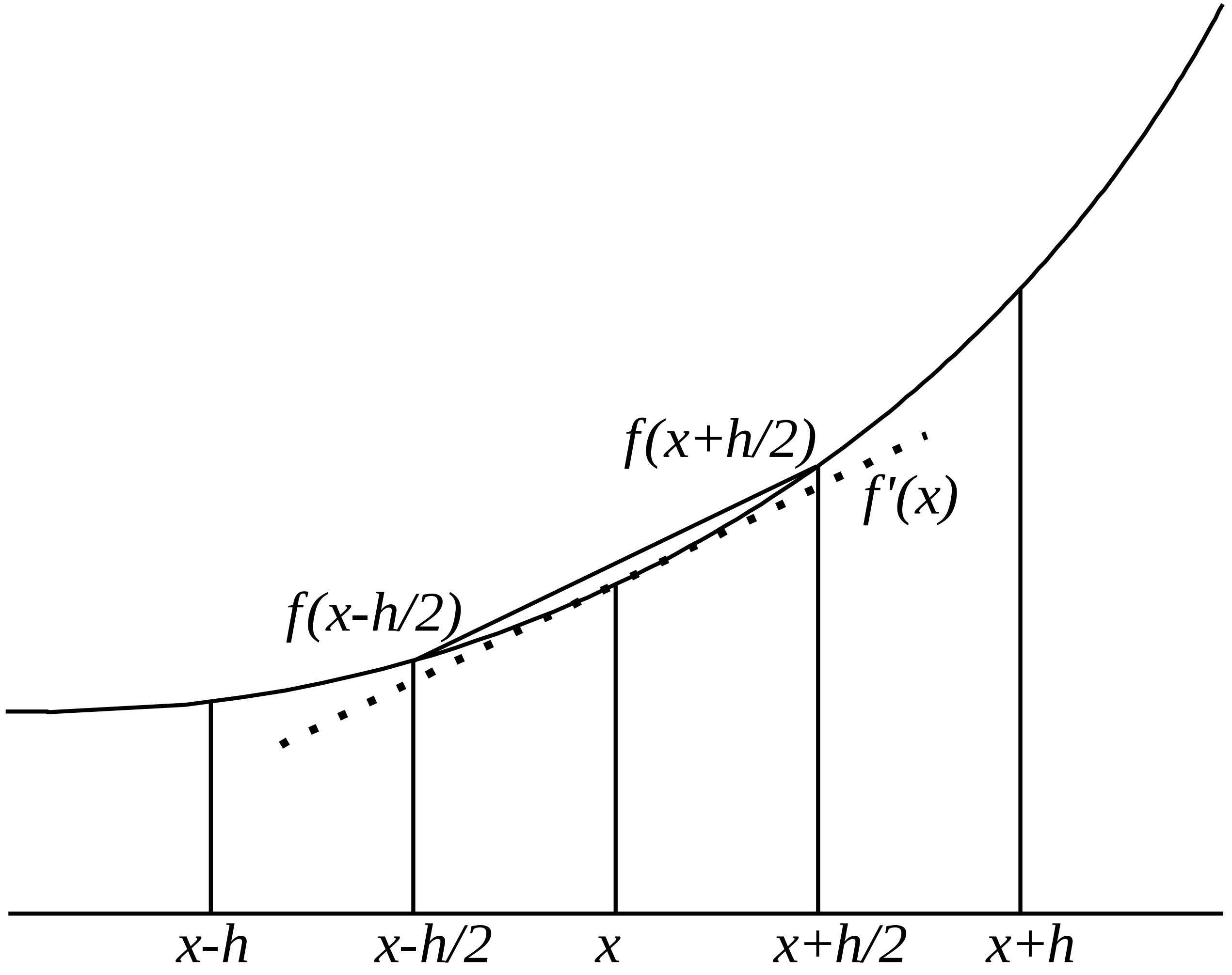
  where *A* - *H* are functions of *x* and *y* only
- Elliptic PDEs: $B^2 - AC < 0$

  (steady state heat equations)
- Parabolic PDEs: $B^2 - AC = 0$

  (heat transfer equations)
- Hyperbolic PDEs: $B^2 - AC > 0$

  (wave equations)

Difference Quotients



Formulas for 1st, 2d Derivatives

$$f'(x) \approx \frac{f(x+h/2) - f(x-h/2)}{h}$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Example of Finite Differences:

Let $u(x) = \sin x$. We will approximate

$u'(1) = \cos 1 = .5403023$

Let us use finite difference quotients

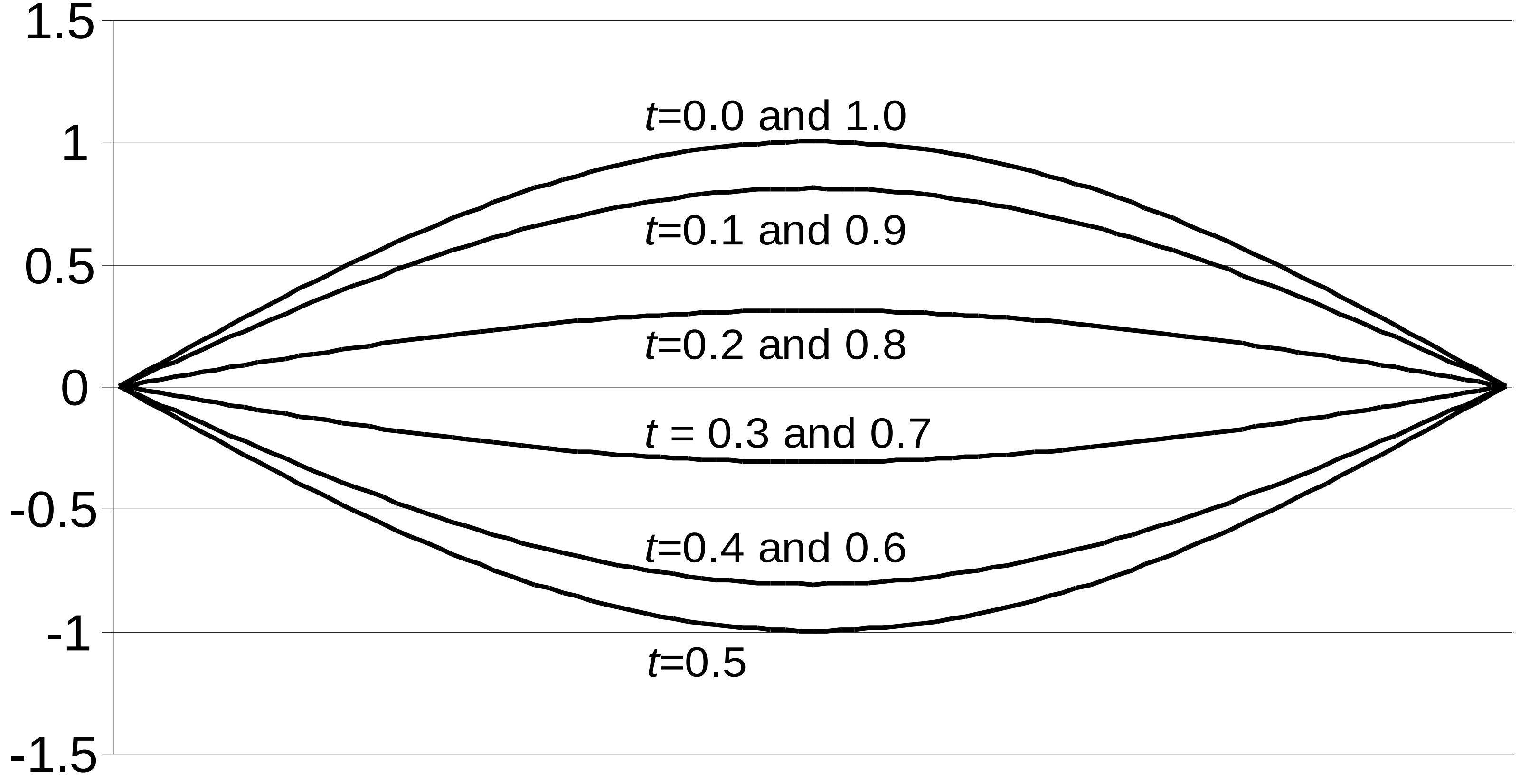
$\cos(1) \approx (\sin(1+h) - \sin(1-h))/2h$

$h = .1, .01, .001\ .0001$

approx= .539402, .540293,.540302,.540302

error=.0009,.000009,.00000009,.0000000009

Vibrating String Problem



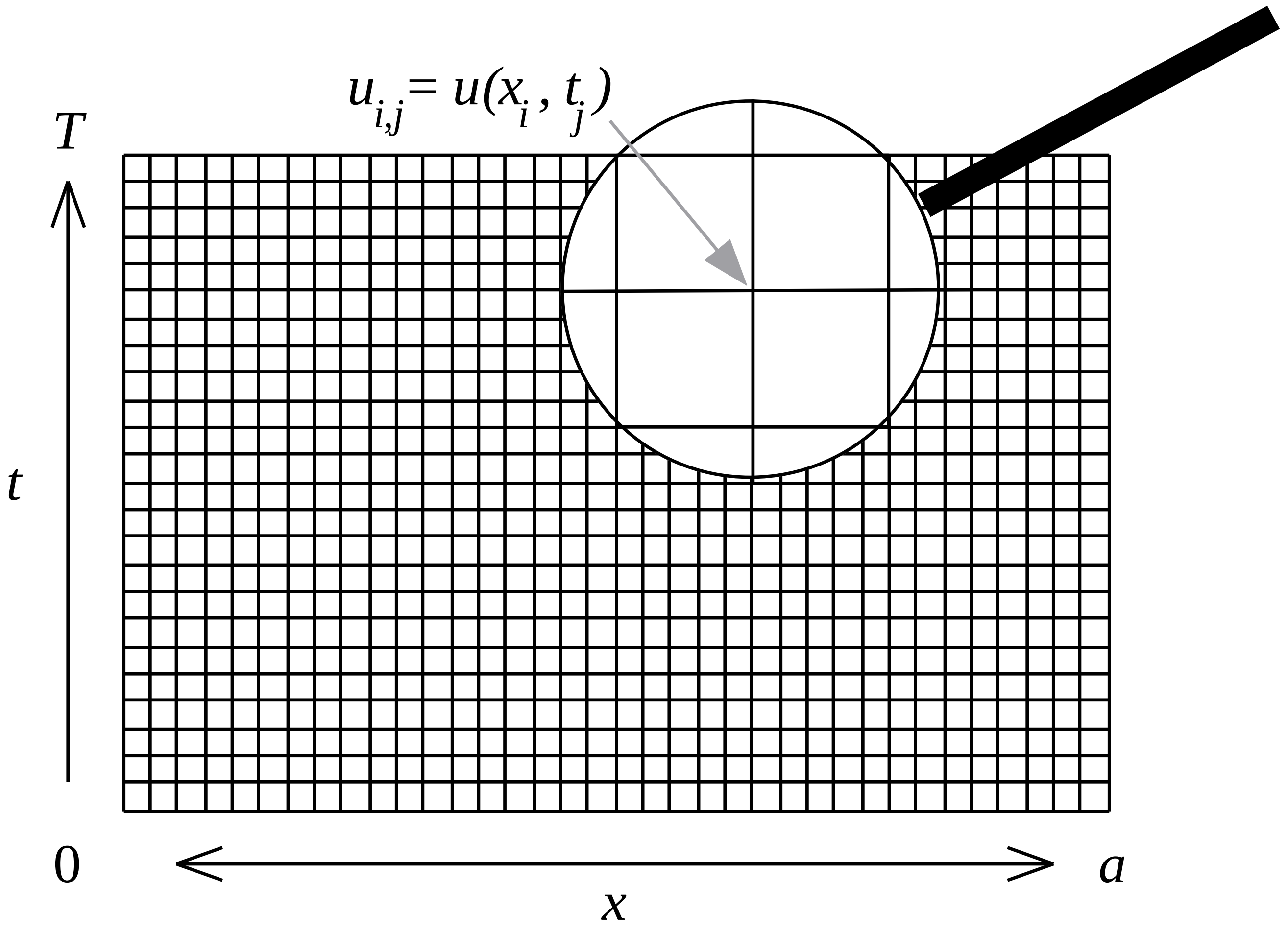
Vibrating string modeled by a hyperbolic PDE
Derived from Newton's and Hooke's Law

$$F = m\,a = -k\,y$$

Solution Stored in 2-D Array

- Each row represents state of string at some point in time
- Each column shows how position of string at a particular point changes with time

Discrete Space, Time Intervals Lead to 2-D Array



Finite Difference Approximation

Central difference approximations

$$\frac{\partial^2 u}{\partial x^2}(x_i,t_\ell) \approx \frac{u_{i-1}^\ell - 2u_i^\ell + u_{i+1}^\ell}{\Delta x^2},$$
$$\frac{\partial^2 u}{\partial t^2}(x_i,t_\ell) \approx \frac{u_i^{\ell-1} - 2u_i^\ell + u_i^{\ell+1}}{\Delta t^2}$$

Inserted into the equation:

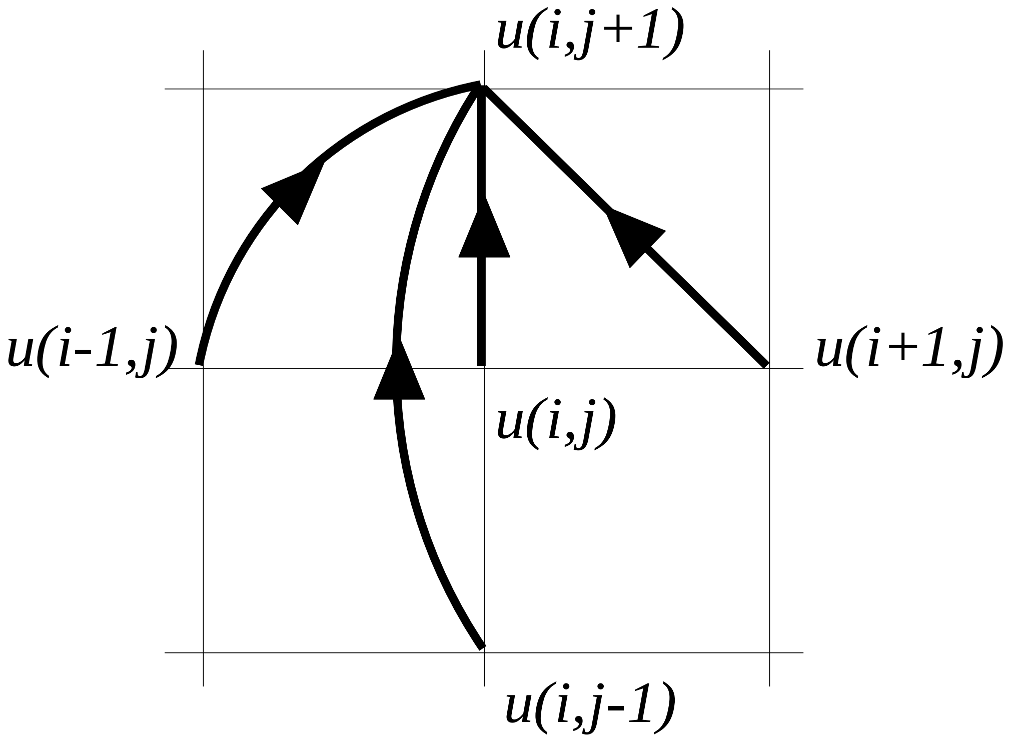
$$\frac{u_i^{\ell-1} - 2u_i^\ell + u_i^{\ell+1}}{\Delta t^2} = \gamma^2 \frac{u_{i-1}^\ell - 2u_i^\ell + u_{i+1}^\ell}{\Delta x^2}$$

Solve for $u_i^{\ell+1}$. Then the difference equation reads

$$u_i^{\ell+1} = 2u_i^\ell - u_i^{\ell-1} + C^2\left(u_{i-1}^\ell - 2u_i^\ell + u_{i+1}^\ell\right)$$

Heart of Sequential C Program

```
u[j+1][i] = 2.0*u[j][i] - u[j-1][i]
   + C2*(u[j][i+1] - 2.0*u[j][i] + u[j][i-1])
```

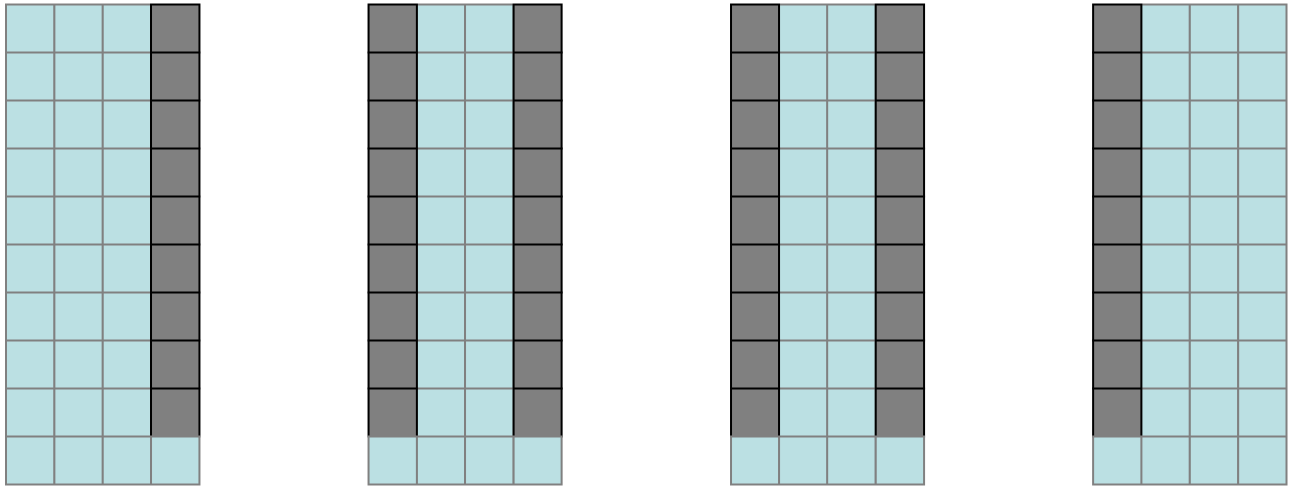


Parallel Program Design

- Associate primitive task with each element of matrix
- Examine communication pattern
- Agglomerate tasks: thread per column
- Static number of identical tasks
- Regular communication pattern
- Strategy: agglomerate columns into blocks
- Difficulty: sharing information across blocks

Communication Still Needed

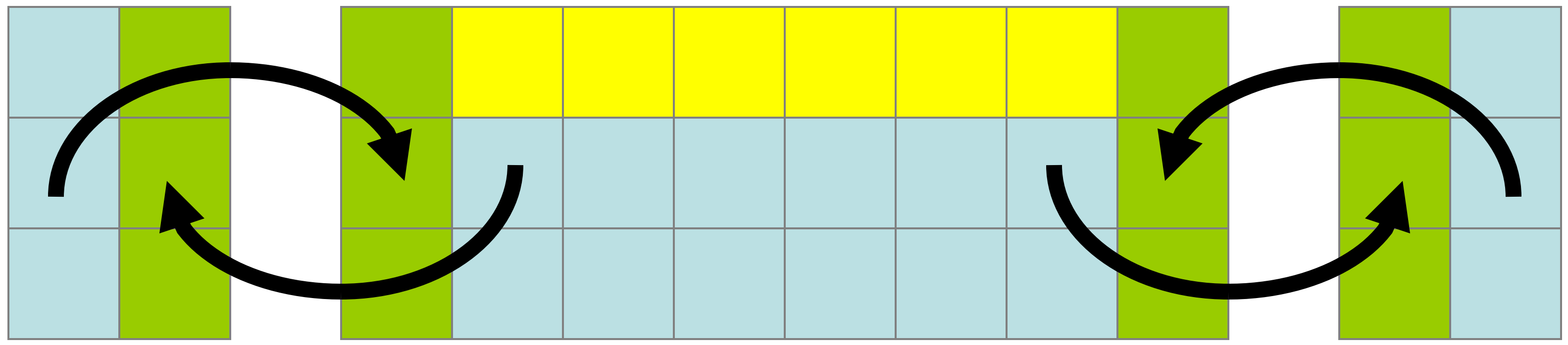
- Initial values (in lowest row) are computed without communication
- Values in black cells cannot be computed without access to values held by other tasks



Ghost Points

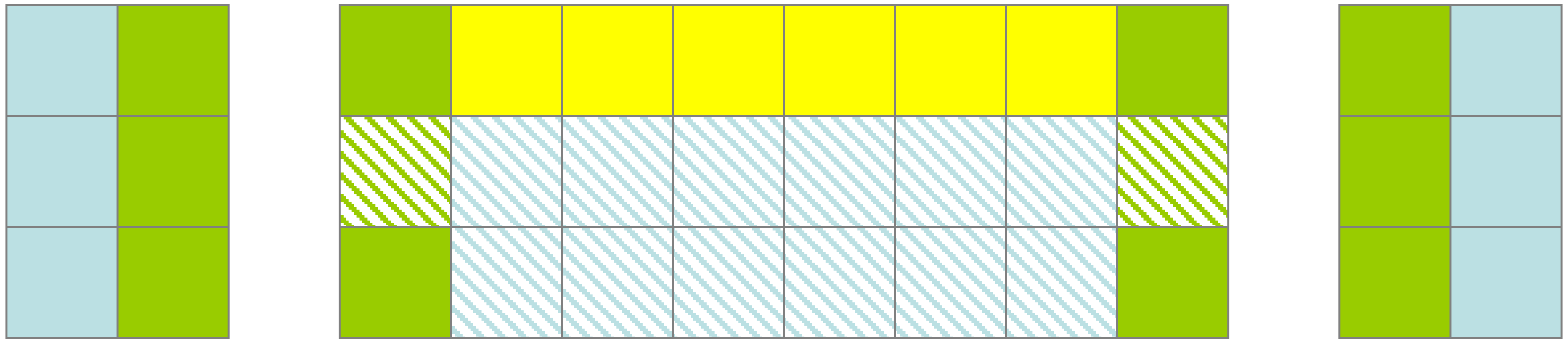
- Ghost points: memory locations used to store redundant copies of data held by neighboring processes
- Allocating ghost points as extra columns simplifies parallel algorithm by allowing same loop to update all cells

Communication in an Iteration



This iteration the process is responsible for computing the values of the yellow cells.

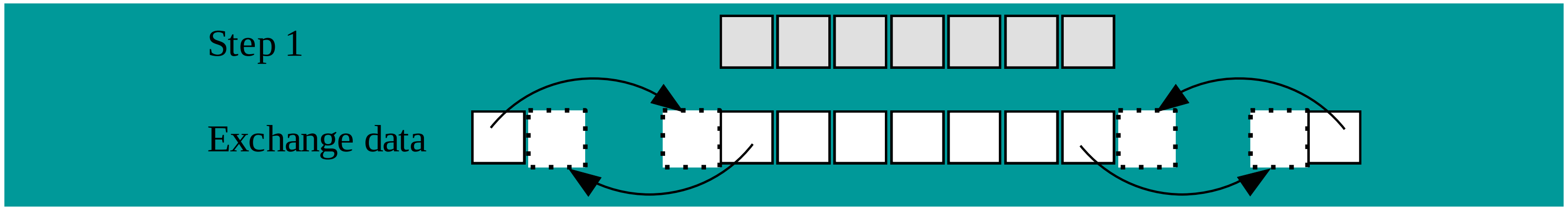
Computation in an Iteration



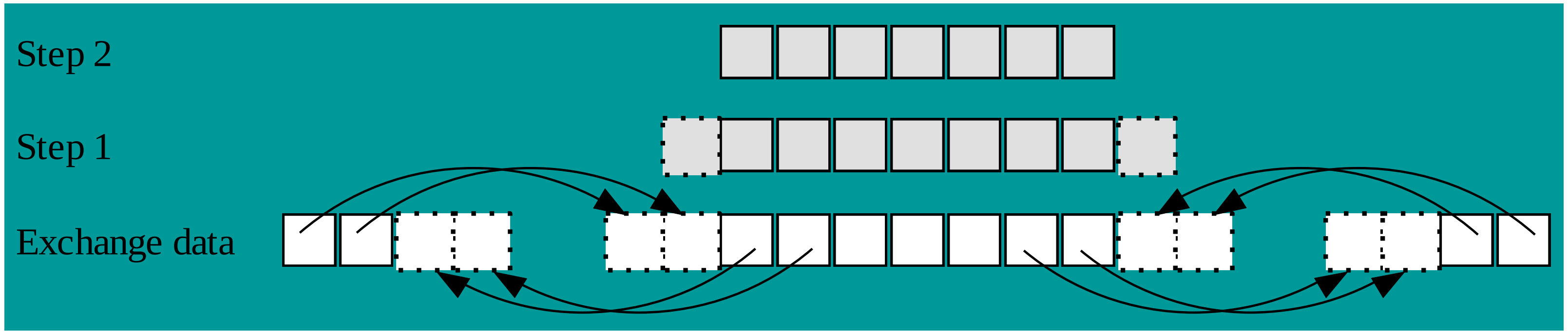
This iteration the process is responsible for computing the values of the yellow cells. The striped cells are the ones accessed as the yellow cell values are computed.

Replicating Computations

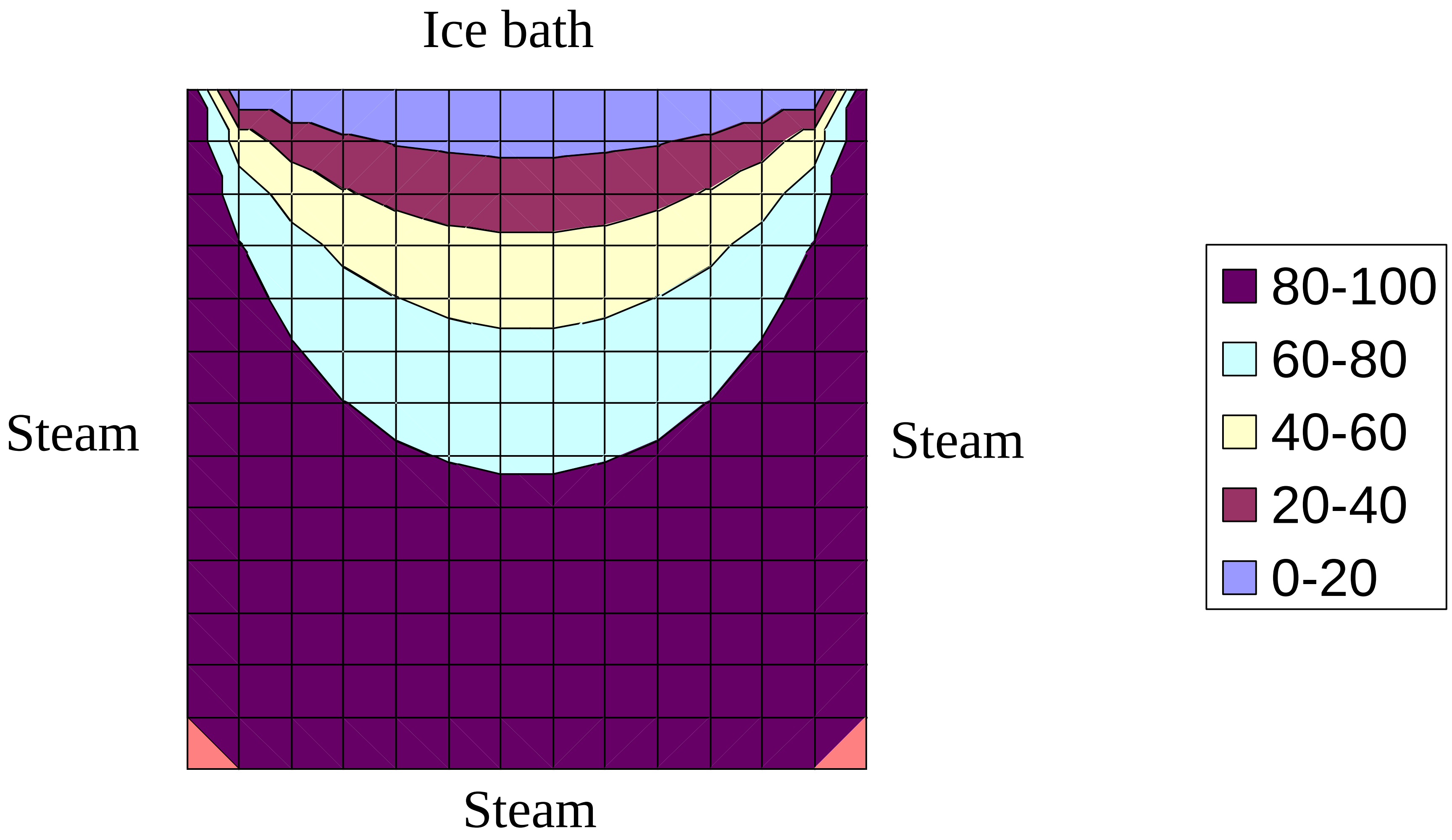
Without replication:



With replication:



Next Case: Steady State Heat Distribution Problem



Solving the Problem

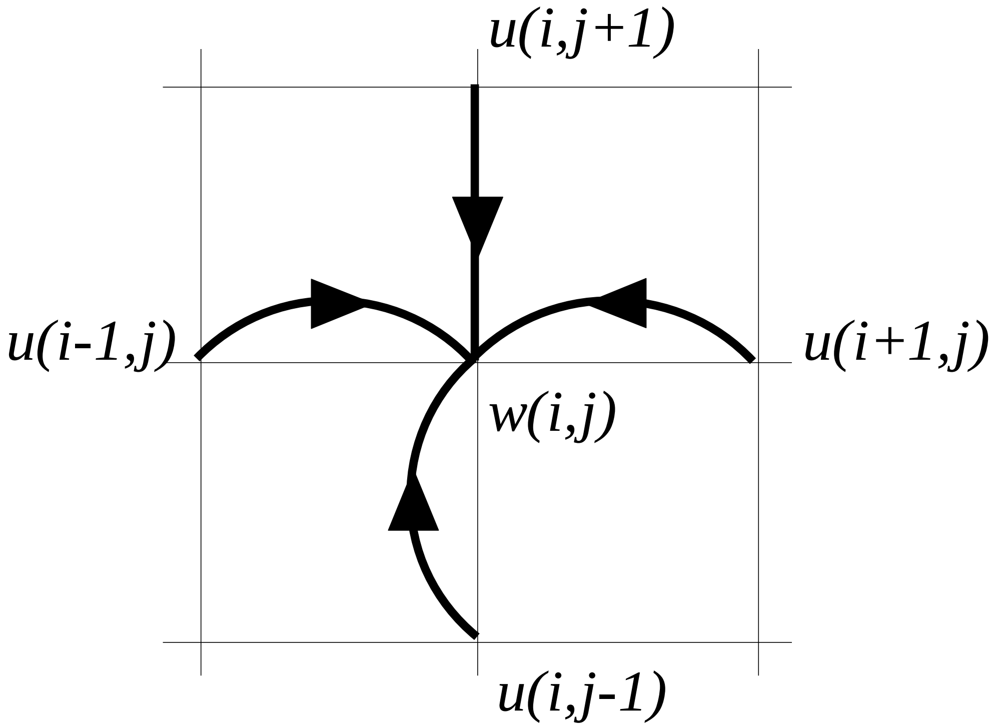
- Underlying PDE is the Poisson equation

$$u_{xx} + u_{yy} = f(x, y)$$

- When f = 0 called Laplace equation
- This is an example of an elliptical PDE
- Will create a 2-D grid
- Each grid point represents value of state state solution at particular (*x*, *y*) location in plate

Heart of Sequential C Program

```
w[i][j] = (u[i-1][j] + u[i+1][j] +
           u[i][j-1] + u[i][j+1]) / 4.0;
```



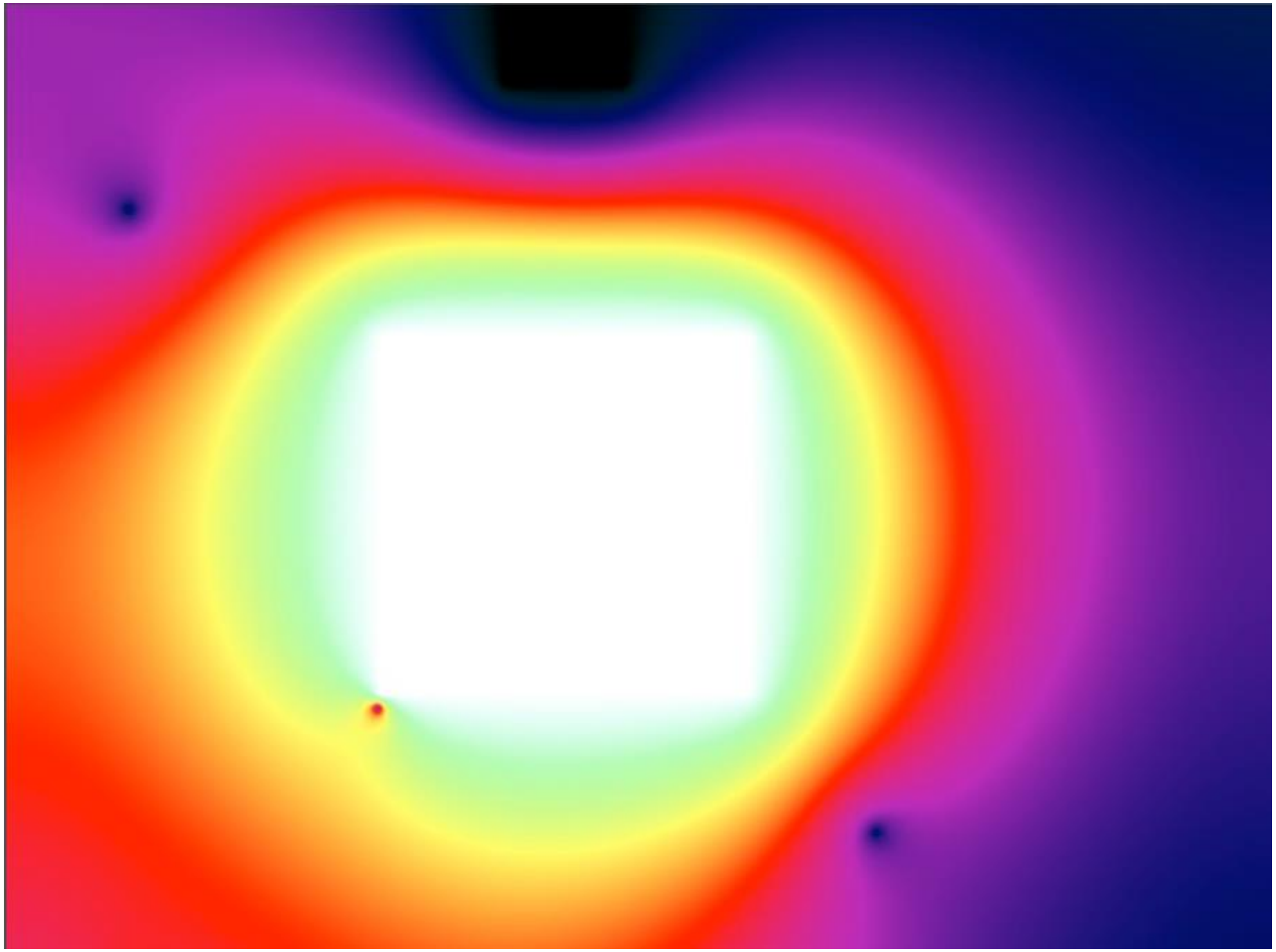
//Cuda Kernel Code that updates temperatures based on difference with average of neighbors
// T-new= T-old + k [Sum(Neighbors) – 4T-old]

```
__global__ void blend_kernel
   ( float *outSrc, const float *inSrc ) {
// map from threadIdx/BlockIdx to pixel position
int x = threadIdx.x + blockIdx.x * blockDim.x;
int y = threadIdx.y + blockIdx.y * blockDim.y;
int offset = x + y * blockDim.x * gridDim.x;

int left = offset - 1;  int right = offset + 1;
if (x == 0) left++;  if (x == DIM-1) right--;

int top = offset - DIM; int bottom = offset + DIM;
if (y == 0) top += DIM; if (y == DIM-1) bottom -=
   DIM;

outSrc[offset] = inSrc[offset] + SPEED * ( inSrc[top] +
```



Optimized Implementation

- Method 1: use ghost points around 2-D blocks, but requires extra copying steps

- Method 2: use “texture memory” in CUDA which provides for fast access – spatially localized cache.