Week 6 Section

Zichen Chen

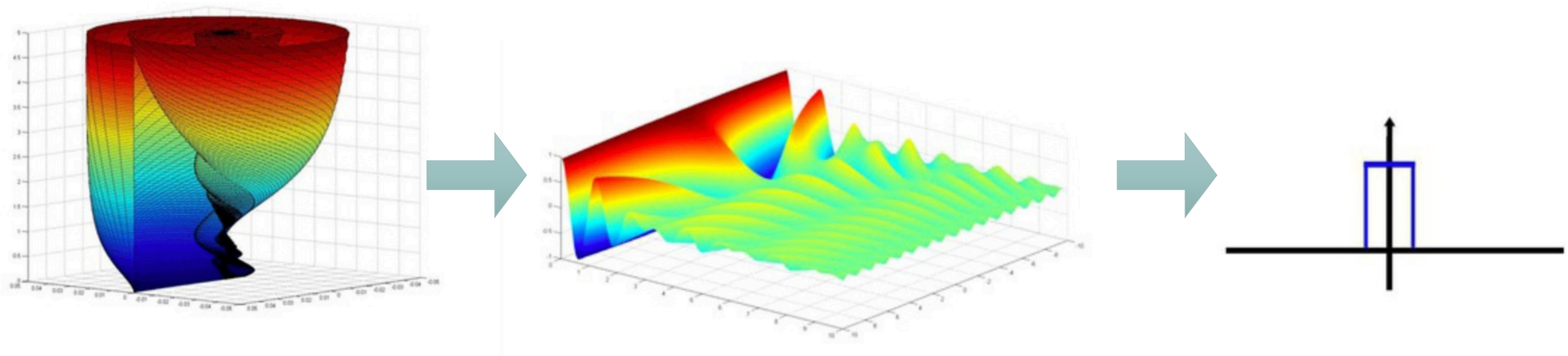
Content

- Use Jacobi’s method to solve Laplace 's Equation
- Floating Point Representation

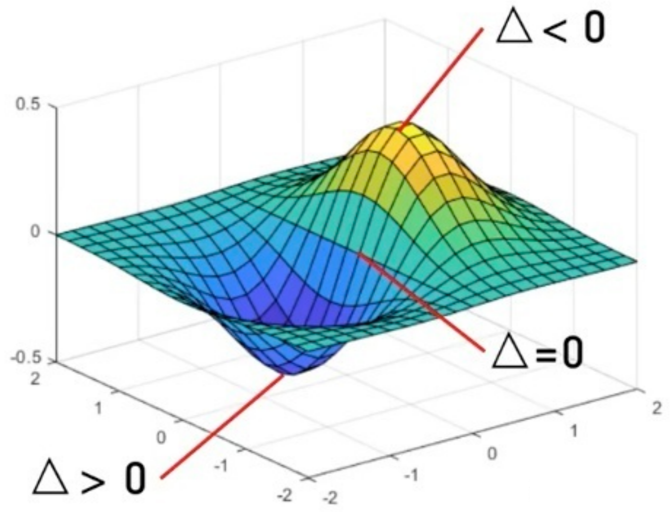
Use Jacobi's method to solve Laplace's Equation

Laplace's Equation

- is a second-order partial differential equation



Laplace's Equation



- is a second-order partial differential equation

$$\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}=0$$

- **f** is solved over a 2D space having coordinates x and y
- If the distance between points (D) is small enough, **f** can be approximated by

$$\frac{\partial^2 f}{\partial x^2}=\frac{1}{\Delta^2}\left[f(x+\Delta,y)-2f(x,y)+f(x-\Delta,y)\right]$$

$$\frac{\partial^2 f}{\partial y^2}=\frac{1}{\Delta^2}\left[f(x,y+\Delta)-2f(x,y)+f(x,y-\Delta)\right]$$

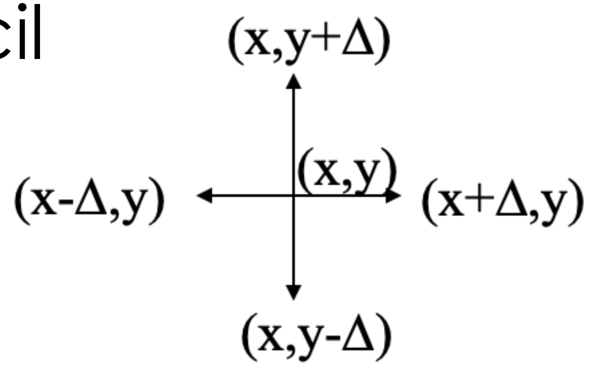
- These equations reduce to $$f(x,y)=\frac{\left[f(x-\Delta,y)+f(x,y-\Delta)+f(x+\Delta,y)+f(x,y+\Delta)\right]}{4}$$

Laplace's Equation

- Note the relationship between the parameters

$$f(x,y) = \frac{\left[f(x-\Delta,y)+f(x,y-\Delta)+f(x+\Delta,y)+f(x,y+\Delta)\right]}{4}$$

- This forms a 4 point stencil



- Any update will involve only **local communication**

Using Jacobi strategy

- Note that in Laplace's equation, we want to solve for all f(**x,y**) which has **2** parameters
- In Jacobi, we want to solve for **x_i** which has only **1** index

- How do we convert f(x,y) into x_i ?
- Associate x_i's with the f(x,y)'s by distributing them in the f 2D matrix in **row-major** (natural) order
- For an *nxn* matrix, there are then *nxn x_i's*, so the A matrix will need to be *(nxn)X(nxn)*

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$

Using Jacobi strategy

- $$f(x,y)=\frac{[f(x-\Delta,y)+f(x,y-\Delta)+f(x+\Delta,y)+f(x,y+\Delta)]}{4}$$
- $$x_i=\frac{[x_{i-n}+x_{i-1}+x_{i+1}+x_{i+n}]}{4}$$
- Rewriting $x_{i-n}+x_{i-1}-4x_i+x_{i+1}+x_{i+n}=0$
- So the **b_i** are 0, what is the A matrix

Finding the A matrix

- Each row only at most 5 non-zero entries
- All entries on the diagonal are 4
- $$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$
- N=9 n=3

Implementation Strategy

- An initial guess is made for all the unknowns, typically x_i = b_i
- New values for the x_i's are calculated using the iteration equations

$$x^t_i = \frac{\left[x^{t-1}_{i-n} + x^{t-1}_{i-1} + x^{t-1}_{i+1} + x^{t-1}_{i+n}\right]}{4}$$

code hint:
new x[i][j] = ¼ (x[i-1][j] + x[I][j+1] + x[i+1][j] + x[i][j-1])

- The updated values are substituted in the iteration equations and the process repeats again
- The user provides a "termination condition" to end the iteration
- $\left|x^t_i - x^{t-1}_i\right| <$ error threshold

Floating Point Representation

Scientific Notation in Base 10

- In base-10, expressing 12,450,000 can be written in a standard way as:
  - 1.245 x 10^7 -> 1.245: mantissa/7: exponent
- It's a given that the base is 10 in this example

- This number in binary:
- 11101

Scientific Notation in Base 10

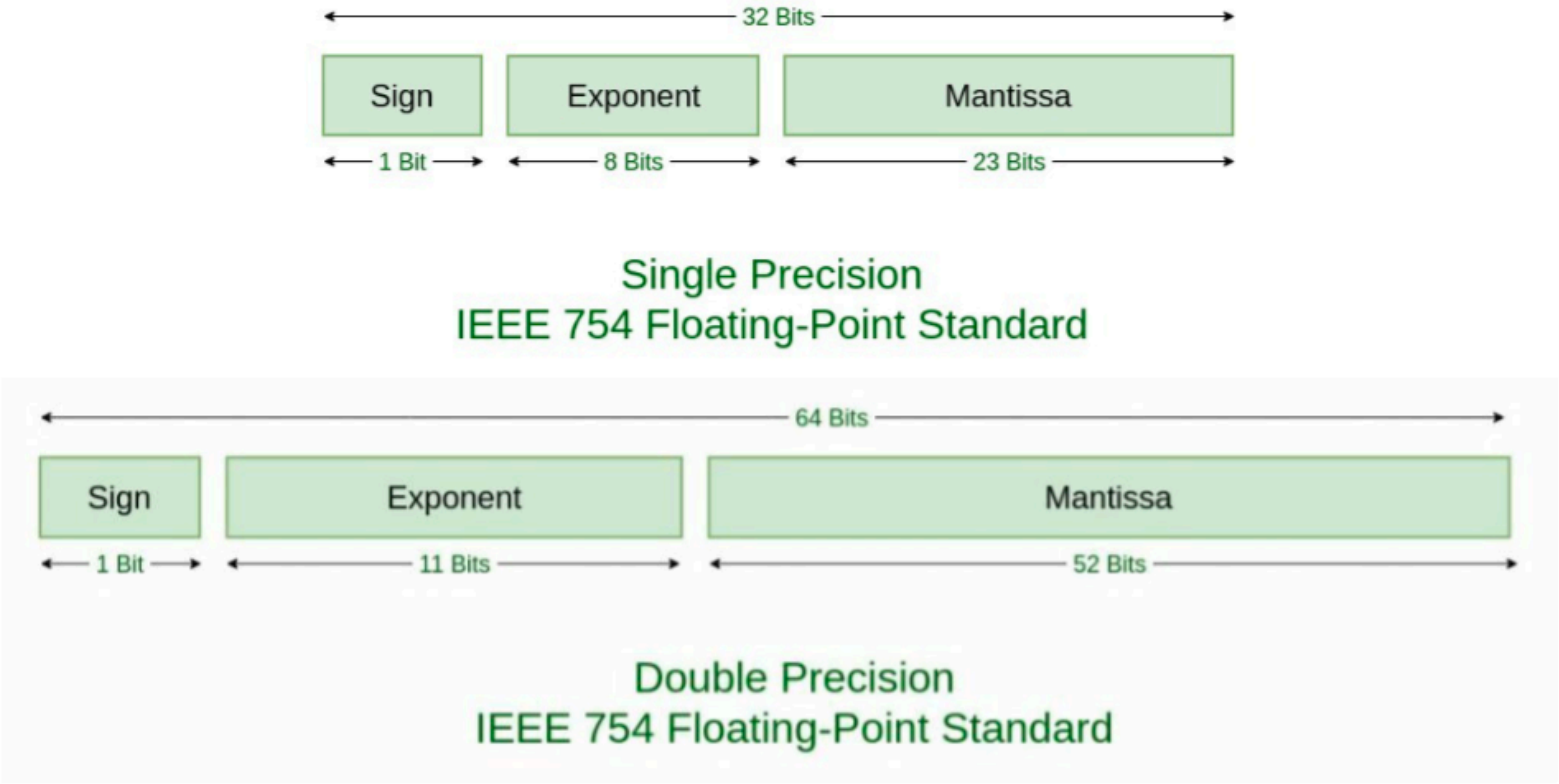
- 1.245 x 10^7

- This number in binary:
- 11101

- 2^4=16 2^3=8 2^2=4   2^1=2   2^0=1
- this is 16+8+4+1 = 29 in decimal

- 29.25 (29 + 1/4)
- 2^-1=½   2^-2=¼
- 1.110101 x 2^4   -> standard scientific notation

Scientific Notation in Base 10



Extend …

- Compression:
  - SignSGD —> using sign to update
- SGD?

A Recipe for Machine Learning

- 1. Given training data:
  - $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$
- 2. Choose each of these:
  - Decision function
    - $\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$
  - Loss function
    - $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$
- 3. Define goal:
  - $$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$
- 4. Train with SGD: (take small steps opposite the gradient)
  - $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$

To Compress

- SignSGD*
  - Basic idea

Algorithm 1 SIGNSGD

Input: learning rate δ, current point x_k

$\tilde{g}_k \leftarrow \text{stochasticGradient}(x_k)$

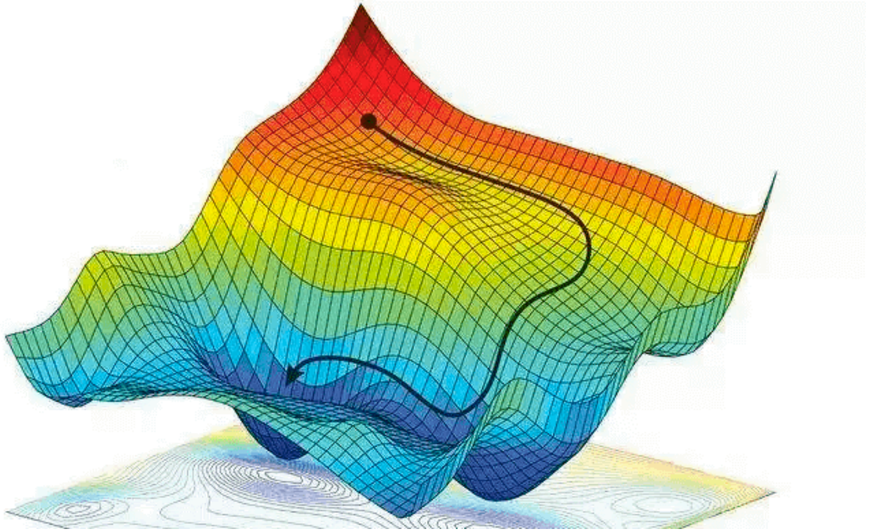
$x_{k+1} \leftarrow x_k - \delta\,\text{sign}(\tilde{g}_k)$

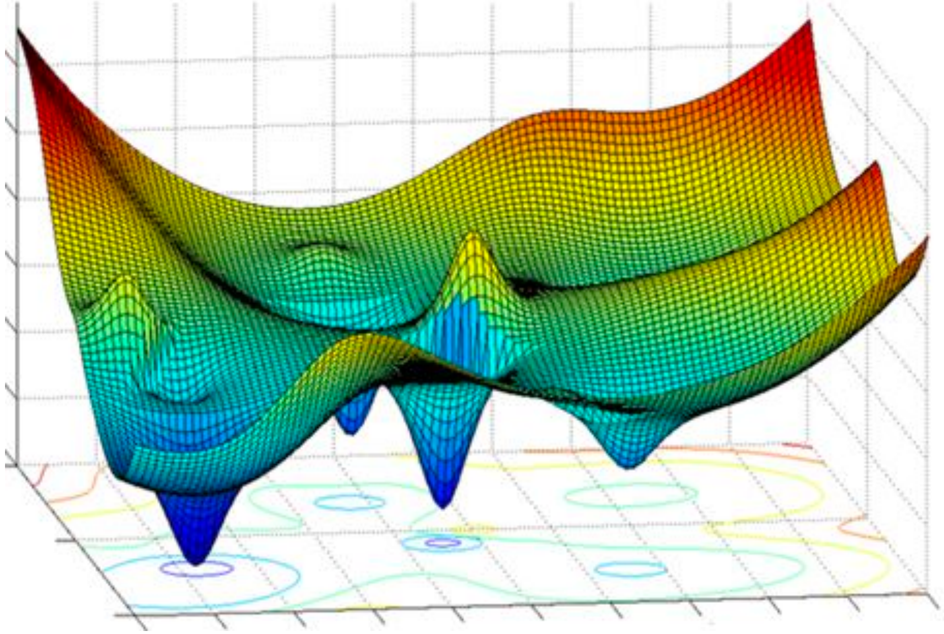
  -
  - +/-1, 0

* Bernstein, Jeremy, et al. "signSGD: Compressed optimisation for non-convex problems." *International Conference on Machine Learning*. PMLR, 2018.

Think …

- Why sign can be used?





Local vs Distributed

- Local:
  - Speed up the convergence
  - Efficient, reduce the computation pressure
  - Reduce the influence from data noise

- Distributed:
  - Communication-efficient
  - Reduce the cost