Carnegie Mellon

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

- **Random Walk**
  - 3-D heat equation
  - Random walk game
  - Randomized PDE solver

3-D Heat Equation

Density ↓ Laplace operator ↓

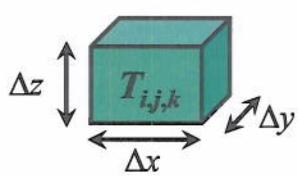
$$\rho \cdot C_p \cdot \frac{\partial T(x,y,z,t)}{\partial t} = \kappa \cdot \nabla^2 T(x,y,z,t) + f(x,y,z,t)$$

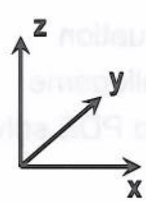
↑ Thermal capacity ↑ Thermal conductivity ↑ Heat source

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Finite Difference

- A control volume





- Discretize PDE at each control volume

$$I_{i,j,k} = C \cdot \frac{\partial T_{i,j,k}}{\partial t} + G_x \cdot \left[T_{i,j,k} - T_{i+1,j,k}\right] + G_x \cdot \left[T_{i,j,k} - T_{i-1,j,k}\right]$$
$$G_y \cdot \left[T_{i,j,k} - T_{i,j+1,k}\right] + G_y \cdot \left[T_{i,j,k} - T_{i,j-1,k}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k+1}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k-1}\right]$$

$$G_x = \frac{\kappa \cdot \Delta y \cdot \Delta z}{\Delta x} \quad G_y = \frac{\kappa \cdot \Delta x \cdot \Delta z}{\Delta y} \quad G_z = \frac{\kappa \cdot \Delta x \cdot \Delta y}{\Delta z}$$

$$C = \rho \cdot C_p \cdot \Delta x \cdot \Delta y \cdot \Delta z \quad I_{i,j,k} = f_{i,j,k} \cdot \Delta x \cdot \Delta y \cdot \Delta z$$

Steady-State Solution

- **Generally interested only in steady state – thermal capacitance is not considered**

$$I_{i,j,k} = C \cdot \frac{\partial T_{i,j,k}}{\partial t} + G_x \cdot \left[T_{i,j,k} - T_{i+1,j,k}\right] + G_x \cdot \left[T_{i,j,k} - T_{i-1,j,k}\right]$$
$$G_y \cdot \left[T_{i,j,k} - T_{i,j+1,k}\right] + G_y \cdot \left[T_{i,j,k} - T_{i,j-1,k}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k+1}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k-1}\right]$$

$$C \cdot \frac{\partial T_{i,j,k}}{\partial t} = 0$$

$$I_{i,j,k} = G_x \cdot \left[T_{i,j,k} - T_{i+1,j,k}\right] + G_x \cdot \left[T_{i,j,k} - T_{i-1,j,k}\right]$$
$$G_y \cdot \left[T_{i,j,k} - T_{i,j+1,k}\right] + G_y \cdot \left[T_{i,j,k} - T_{i,j-1,k}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k+1}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k-1}\right]$$

Steady-State Solution

- **Result in a set of linear equations**

$$I_{i,j,k} = G_x \cdot \left[T_{i,j,k} - T_{i+1,j,k}\right] + G_x \cdot \left[T_{i,j,k} - T_{i-1,j,k}\right]$$
$$G_y \cdot \left[T_{i,j,k} - T_{i,j+1,k}\right] + G_y \cdot \left[T_{i,j,k} - T_{i,j-1,k}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k+1}\right] + G_z \cdot \left[T_{i,j,k} - T_{i,j,k-1}\right]$$

$$I_{i,j,k} = 2 \cdot \left(G_x + G_y + G_z\right) \cdot T_{i,j,k} - G_x \cdot \left(T_{i+1,j,k} + T_{i-1,j,k}\right)$$
$$- G_y \cdot \left[T_{i,j+1,k} + T_{i,j-1,k}\right] - G_z \cdot \left[T_{i,j,k+1} + T_{i,j,k-1}\right]$$

Steady-State Solution

$$I_{i,j,k} = 2\cdot\left(G_x + G_y + G_z\right)\cdot T_{i,j,k} - G_x\cdot\left(T_{i+1,j,k} + T_{i-1,j,k}\right)$$
$$- G_y\cdot\left[T_{i,j+1,k} + T_{i,j-1,k}\right] - G_z\cdot\left[T_{i,j,k+1} + T_{i,j,k-1}\right]$$

$$\Downarrow$$

$$\begin{aligned} T_{i,j,k} = {} & \frac{G_x}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i+1,j,k} + \frac{G_x}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i-1,j,k} \\ & + \frac{G_y}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j+1,k} + \frac{G_y}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j-1,k} \\ & + \frac{G_z}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j,k+1} + \frac{G_z}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j,k-1} \\ & + \frac{1}{2\cdot\left(G_x+G_y+G_z\right)}\cdot I_{i,j,k} \end{aligned}$$

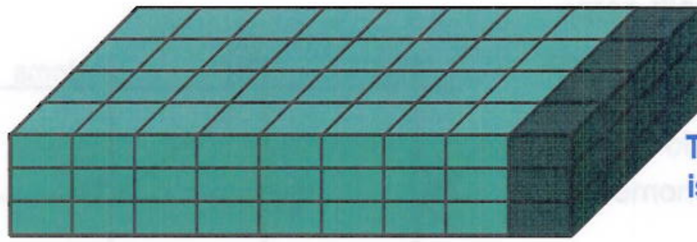
Steady-State Solution

$$\begin{aligned} T_{i,j,k} = {} & \underbrace{\boxed{\frac{G_x}{2\cdot\left(G_x+G_y+G_z\right)}}}_{g_x}\cdot T_{i+1,j,k} + \frac{G_x}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i-1,j,k} \\ & + \underbrace{\boxed{\frac{G_y}{2\cdot\left(G_x+G_y+G_z\right)}}}_{g_y}\cdot T_{i,j+1,k} + \frac{G_y}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j-1,k} \\ & + \underbrace{\boxed{\frac{G_z}{2\cdot\left(G_x+G_y+G_z\right)}}}_{g_z}\cdot T_{i,j,k+1} + \frac{G_z}{2\cdot\left(G_x+G_y+G_z\right)}\cdot T_{i,j,k-1} \\ & + \underbrace{\boxed{\frac{1}{2\cdot\left(G_x+G_y+G_z\right)}}}_{g_I}\cdot I_{i,j,k} \end{aligned}$$

$$\Downarrow$$

$$T_{i,j,k} = g_x T_{i+1,j,k} + g_x T_{i-1,j,k} + g_y T_{i,j+1,k} + g_y T_{i,j-1,k}$$
$$+ g_z T_{i,j,k+1} + g_z T_{i,j,k-1} + g_I I_{i,j,k}$$

Boundary Conditions



Temperature is fixed at T_0

$$T_{i,j,k} = T_0$$ for all {i,j,k} at boundary

Thermal Equation

$$T_{i,j,k} = g_x T_{i+1,j,k} + g_x T_{i-1,j,k} + g_y T_{i,j+1,k} + g_y T_{i,j-1,k} + g_z T_{i,j,k+1} + g_z T_{i,j,k-1} + g_I I_{i,j,k}$$

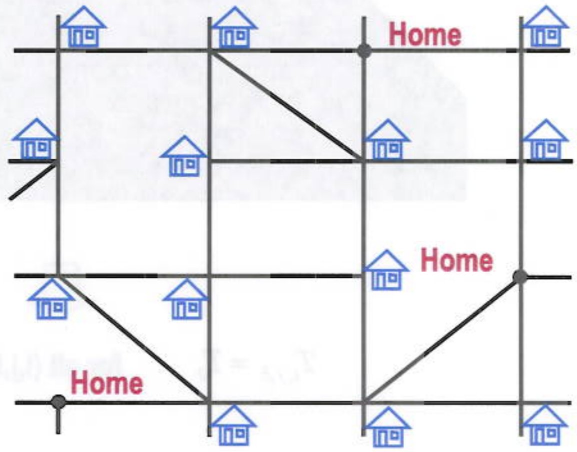
$$T_{i,j,k} = T_0 \quad @ \quad Boundary$$

- **Linear thermal equation can be solved by many techniques**
  - Gaussian elimination
  - Conjugate gradient method
  - Etc.

- **In this lecture, we will explore a new "random" technique to solve discretized PDE**

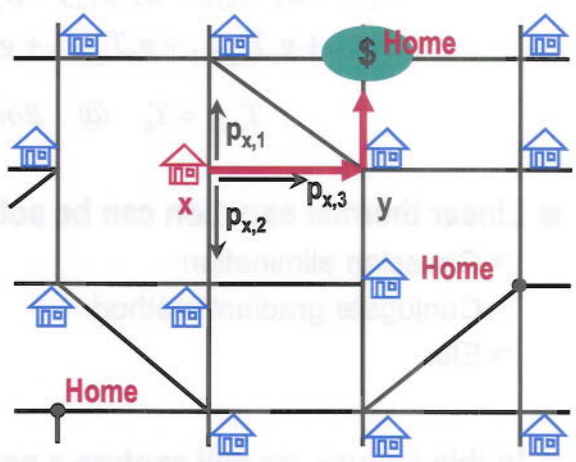
Random Walk Game

- Random walk game
  - A network of roads
  - A motel at each intersection
  - A set of homes



Random Walk Game

- Start from node x
- Walk one (randomly chosen) road every day
  - Each direction is associated with probability $p_{i,j}$
- Stay the night at motel y
- Motel charges m_y
- Keep going until home
- Get reward m_0 at each home



Random Walk Game

- **Problem: find the average amount of earned money in the end as a function of the starting node x**

  f(x) = E[money earned in the end | from node x]

- **f(x) can be estimated by Monte Carlo analysis**
  - Estimate the expected value from a number of random sampling points

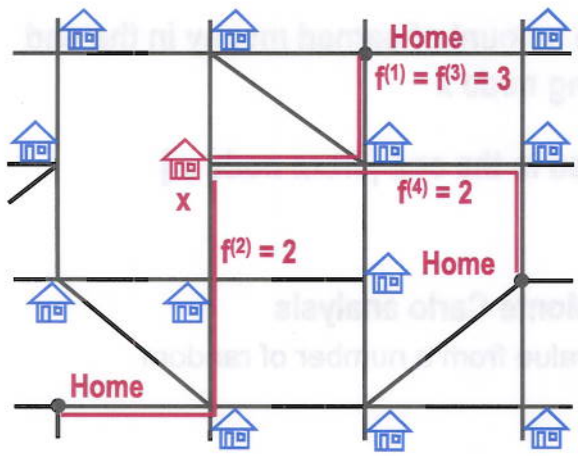
Monte Carlo Analysis

f(x) = E[money earned in the end | from node x]

- For i = 1,2,...,M
  - Start from node x
  - Perform random walk to reach home
  - Calculate the total money earned during this walk: $f^{(i)}$
- End For
- f(x) is estimated by:

$$f(x) \approx \frac{1}{M} \cdot \sum_{i=1}^{M} f^{(i)}$$

Monte Carlo Analysis

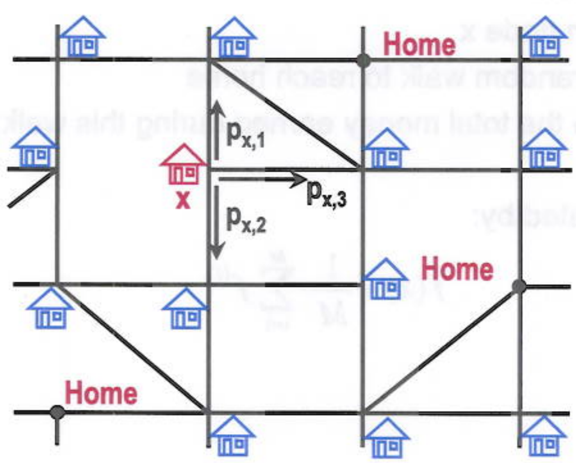


$f(x) = \frac{1}{4}(3+2+3+2)$
$= 10/4$
$= 2.5$

- **Why is random walk game related to thermal analysis?**
  - To understand the connection, we need to analytically model the game as a Markov chain

Markov Chain Model

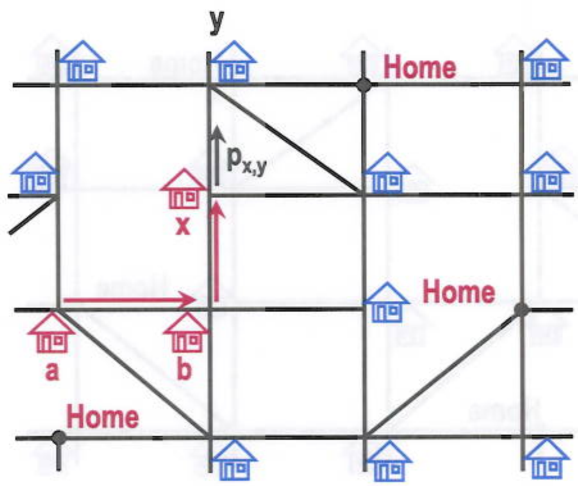
- **The random walk problem can be modeled as a Markov chain**
  - The transition probability from x to y is uniquely determined by x and y only
  - It is independent of any previous locations



Markov Chain Model

- **Mathematically, it means:**

$$p(3/x, b, a) = p(3/x) = p_{x,3}$$



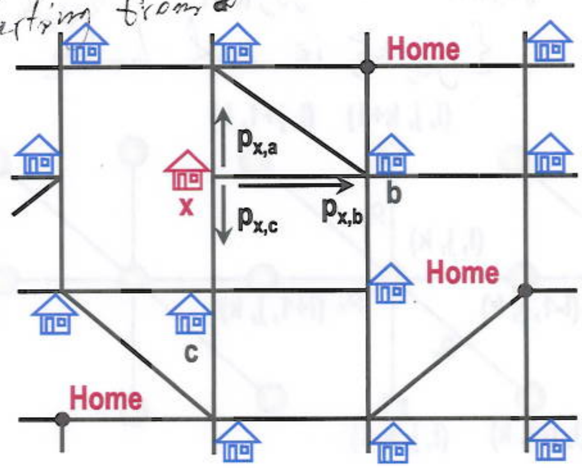
Markov Chain Model

- **Linear equation for f(x)** (x is not home)

$$f(x) = P_{x,a} f(a) + P_{x,b} f(b) + P_{x,c} f(c) - M_x$$

money earned by starting from x

Payment made at x



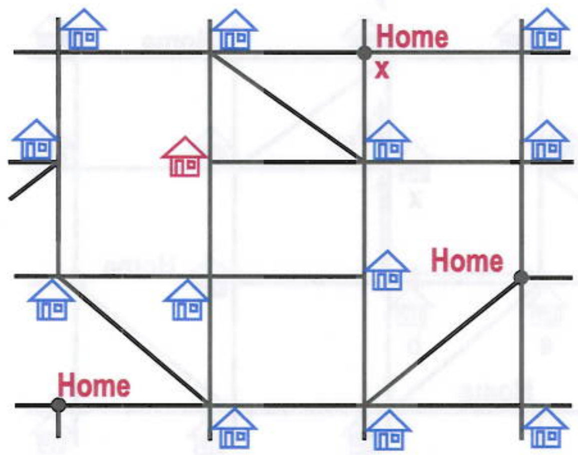
Markov Chain Model

- **Linear equation for f(x)**

(x is home)

$f(x) = m_0$

reward at home

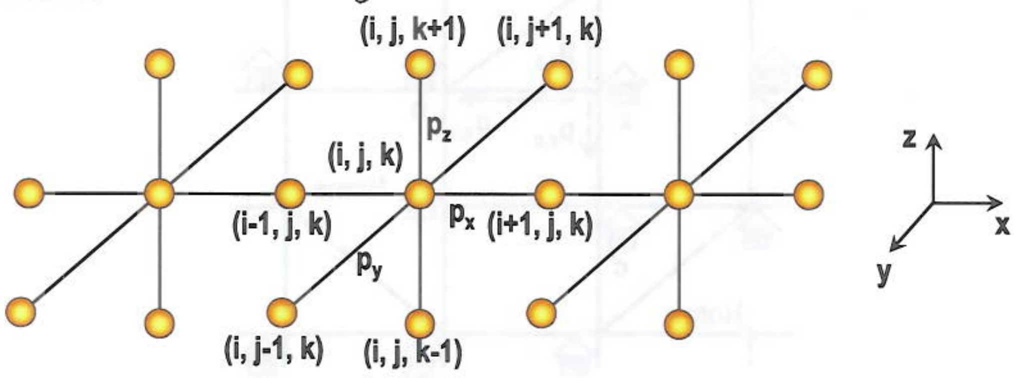


Random Walk Game for 3-D Grid

- **If the road network is a 3-D grid**

$$f_{i,j,k} = P_x f_{i+1,j,k} + P_x f_{i-1,j,k} + P_y f_{i,j+1,k} + P_y f_{i,j-1,k} + P_z f_{i,j,k+1} + P_z f_{i,j,k-1} - m_{i,j,k}$$

$\{i,j,k\}$ is not home

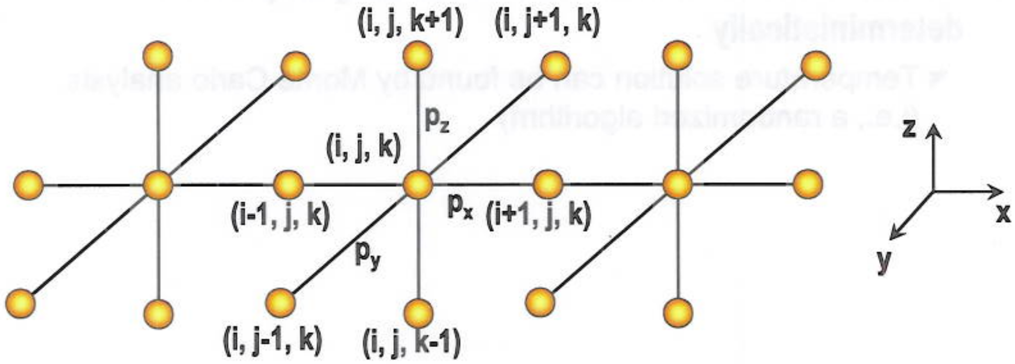


Random Walk Game for 3-D Grid

- If the road network is a 3-D grid

$\{i, j, k\}$ is home

$f_{i,j,k} = m_0$



Thermal Analysis vs. Random Walk Game

- Both thermal analysis and random walk game can be modeled by similar linear equations

$$T_{i,j,k} = g_x T_{i+1,j,k} + g_x T_{i-1,j,k} + g_y T_{i,j+1,k} + g_y T_{i,j-1,k} + g_z T_{i,j,k+1} + g_z T_{i,j,k-1} + g_I I_{i,j,k}$$

$$T_{i,j,k} = T_0$$

@ boundary

Thermal analysis

$$f_{i,j,k} = p_x f_{i+1,j,k} + p_x f_{i-1,j,k} + p_y f_{i,j+1,k} + p_y f_{i,j-1,k} + p_z f_{i,j,k+1} + p_z f_{i,j,k-1} - m_{i,j,k}$$

$$f_{i,j,k} = m_0$$

@ home

Random walk game

Thermal Analysis vs. Random Walk Game

- **This observation has a two-fold meaning**

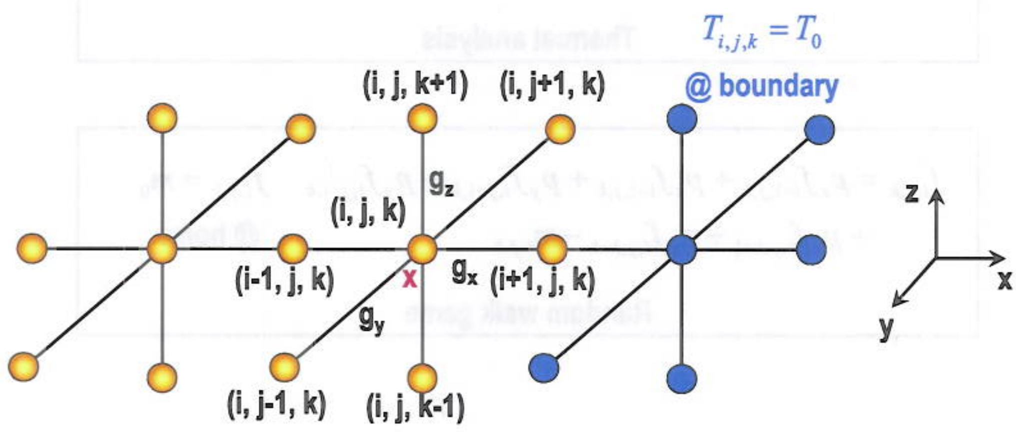
- **#1: We do not have to play random walk game by Monte Carlo**
  - It can be solved deterministically based on linear equation

- **#2: We do not have to solve thermal analysis problem deterministically**
  - Temperature solution can be found by Monte Carlo analysis (i.e., a randomized algorithm)

Random Walk for Thermal Analysis

- **Problem: find temperature at x by random walk**

$$T_{i,j,k} = g_x T_{i+1,j,k} + g_x T_{i-1,j,k} + g_y T_{i,j+1,k} + g_y T_{i,j-1,k}$$
$$+ g_z T_{i,j,k+1} + g_z T_{i,j,k-1} + g_I I_{i,j,k}$$



Random Walk for Thermal Analysis

- **Random walk**
  - Start from node x with reward $g_I I_x$
  - Walk across one (randomly chosen) edge where each walking direction is associated with a probability g_x, g_y or g_z
  - Reach node {i,j,k} and get reward $g_I I_{i,j,k}$
  - Keep going until reaching boundary
  - Get reward T_0 at boundary
  - Calculate the total reward earned during this walk

$$T_{i,j,k} = g_x T_{i+1,j,k} + g_x T_{i-1,j,k} + g_y T_{i,j+1,k} + g_y T_{i,j-1,k} + g_z T_{i,j,k+1} + g_z T_{i,j,k-1} + g_I I_{i,j,k}$$

Random Walk for Thermal Analysis

- **Monte Carlo analysis**

T_x = E[reward earned in the end | from node x]

- For i = 1,2,...,M
  - Start from node x
  - Perform random walk to reach boundary
  - Calculate the total reward earned during this walk: $f^{(i)}$
- End For
- T_x is estimated by:

$$T_x \approx \frac{1}{M} \cdot \sum_{i=1}^{M} f^{(i)}$$

Deterministic Solver vs. Random Walk

- **The efficacy of both algorithms is problem-dependent**
- **In general, random walk is preferable if we are only interested in local temperature**

  $$T_x = E[\text{ reward earned in the end} \mid \text{from node } x\ ]$$

  - We do not have to solve the complete linear equation
  - Random walk quickly tells us the temperature at "a" location x
- **Random walk can also be used to generate "good" pre-conditioner for conjugate gradient method**

Summary

- **Random walk**
  - 3-D heat equation
  - Random walk game
  - Randomized PDE solver

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

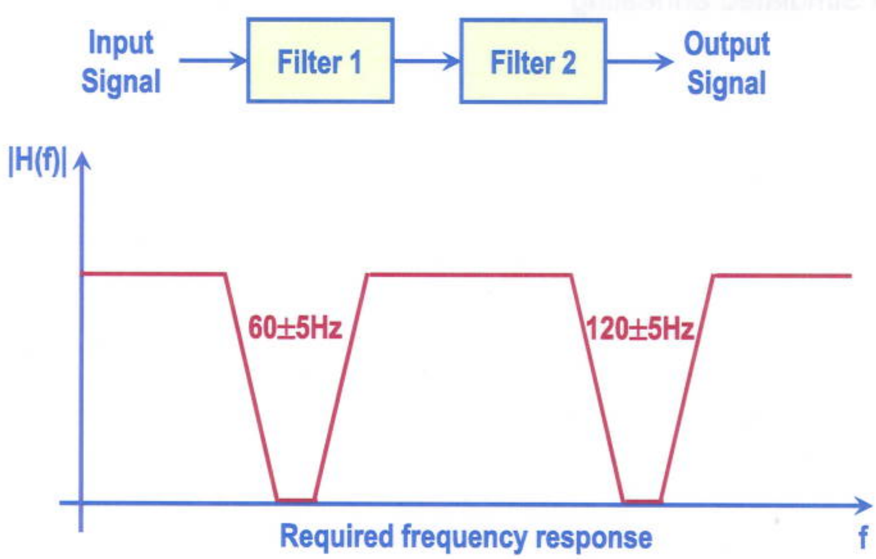
- **Stochastic Optimization**
  - Simulated annealing

Local Optimization

- **All optimization algorithms in early lectures assumes “local convexity” for cost function and constraint set**
  - Gradient method
  - Newton method
  - Conjugate gradient method
  - Interior point method
- **Global convergence cannot be guaranteed if the actual cost function or constraint set is non-convex**

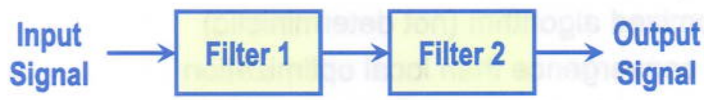
Filter Design Example

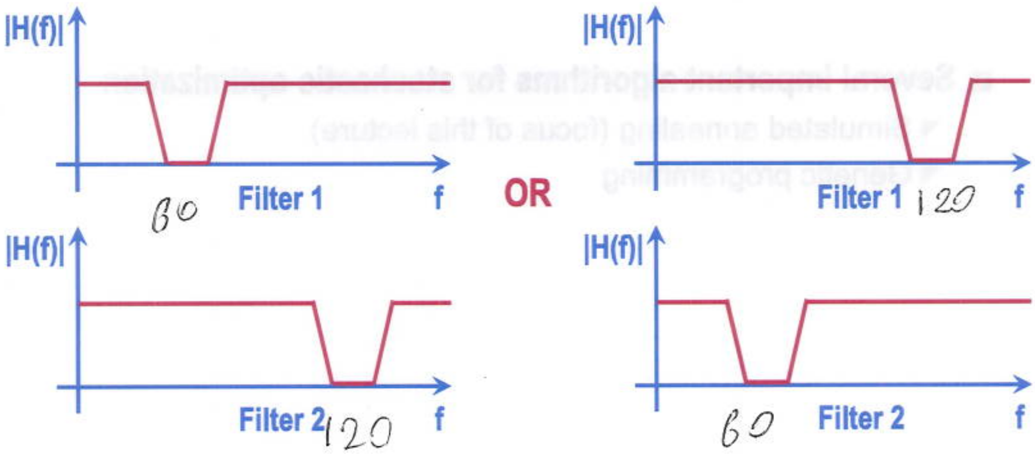
- **Design a band-stop filter to remove power supply noise**



Filter Design Example

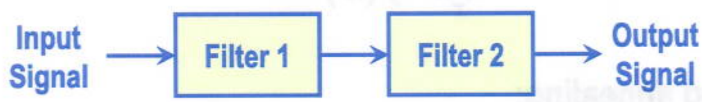
- Design a band-stop filter to remove power supply noise

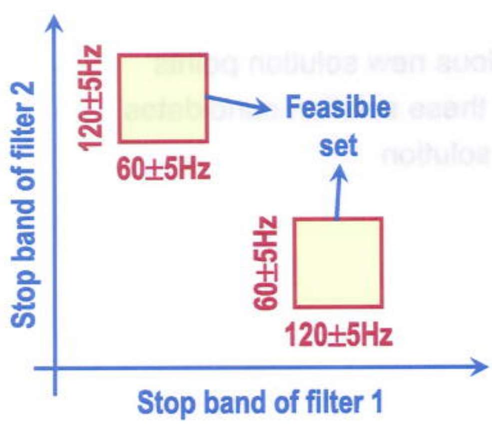




Filter Design Example

- Design a band-stop filter to remove power supply noise





Feasible set is not continuous in this example!

Stochastic Optimization

- **Stochastic optimization is another useful technique for nonlinear programming**
  - Randomized algorithm (not deterministic)
  - Better convergence than local optimization
  - More expensive in computational cost
- **Several important algorithms for stochastic optimization**
  - Simulated annealing (focus of this lecture)
  - Genetic programming

Simulated Annealing

- **Unconstrained optimization**

$$\min_{X} \quad f(X)$$

- **Simulated annealing:**
  - Start from an initial point
  - Repeatedly consider various new solution points
  - Accept or reject some of these solution candidates
  - Converge to the optimal solution

Simulated Annealing

- **Unconstrained optimization**

$$\min_{X} \quad f(X)$$

- **Simulated annealing was introduced by Metropolis in 1953**

- **It is based on "similarities" and "analogies" with the way that alloys manage to find a nearly global minimum energy level when they are cooled slowly**

Simulated Annealing

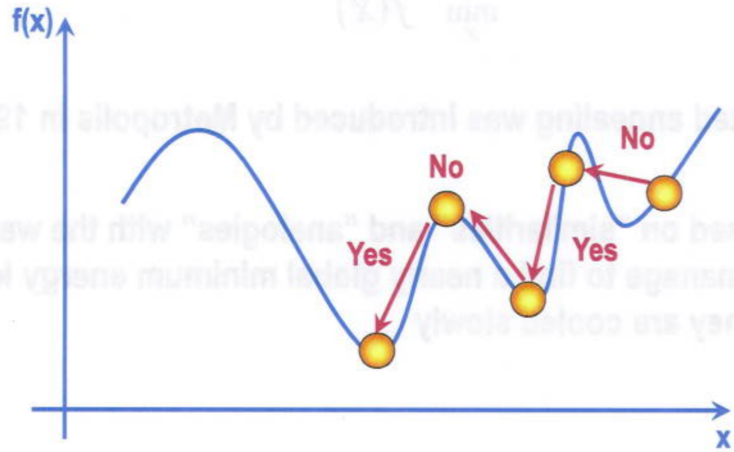
- **Local optimization vs. simulated annealing**

- **Local optimization**
  - Start from an initial point
  - Repeatedly consider various new solution points
  - Reduce cost function at each iteration
  - Converge to optimal solution

- **Simulated annealing**
  - Start from an initial point
  - Repeatedly consider various new solution points
  - Accept/reject new solution using probability at each iteration
  - Converge to optimal solution

Simulated Annealing

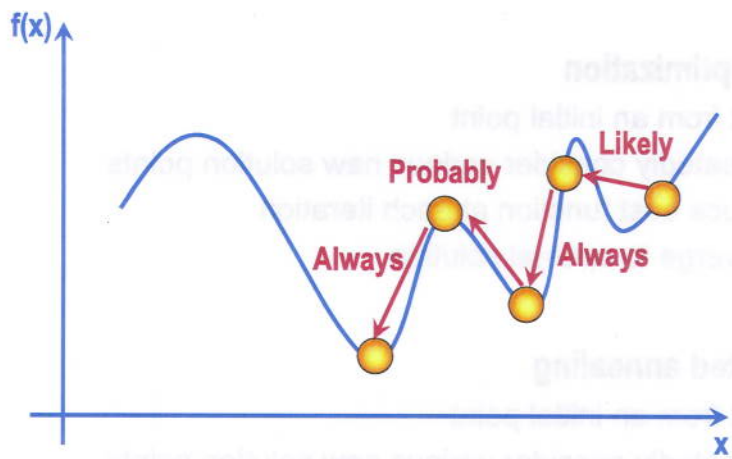
- Local optimization



Local optimization attempts to reduce cost function at each iteration

Simulated Annealing

- Simulated annealing



Simulated annealing accept/reject new solution candidate based on probability

Simulated Annealing

- **Step 1: start from an initial point $X = X_0$ & $K = 0$**
- **Step 2: evaluate cost function $F = f(X)$**
- **Step 3: randomly move from X_K to a new solution X_{K+1}**
- **Step 4: if $f(X_{K+1}) < F$, then**
  - Accept new solution
  - $X = X_{K+1}$ & $F = f(X_{K+1})$
- **End if**

Similar to local optimization

- **Step 5: if $f(X_{K+1}) \geq F$, then**
  - Accept new solution with certain probability
  - $X = X_{K+1}$ & $F = f(X_{K+1})$ iff rand(1) < ε
- **End if**

Help to get out of local minimum

- **Step 6: K = K + 1 & go to Step 3**

Simulated Annealing

- **Accept/reject new solution with the probability ε**
  - If $f(X_{K+1}) \geq F$, then
    - Accept new solution with certain probability
    - $X = X_{K+1}$ & $F = f(X_{K+1})$ iff rand(1) < ε
  - End if

- **Option 1**
  - Constant probability, i.e., $\varepsilon = 0.1$

- **Option 2 (better than Option 1)**
  - Dynamically varying probability, i.e., decreasing over time

Simulated Annealing

- **Accept/reject new solution with the probability ε**
  - If $f(X_{K+1}) \geq F$, then
    - Accept new solution with certain probability
    - $X = X_{K+1}$ & $F = f(X_{K+1})$ iff rand(1) < ε
  - End if
- **Use Boltzmann distribution to determine the probability ε**

rand(1) < ε

$$\varepsilon = \exp\left[-\frac{f(X_{K+1}) - F}{T_{K+1}}\right]$$

  - T_{K+1} is a "temperature" parameter that gradually decreases
  - E.g., $T_{K+1} = \alpha \cdot T_K$ where $\alpha < 1$

Simulated Annealing

- **Accept/reject new solution with the probability ε**
  - If $f(X_{K+1}) \geq F$, then
    - Accept new solution with certain probability
    - $X = X_{K+1}$ & $F = f(X_{K+1})$ iff

$$\text{rand}(1) \leq \exp\left[-\frac{f(X_{K+1}) - F}{T_{K+1}}\right]$$

  - End if
- **High temperature**
  - Attempt to accept all new solutions even if $f(X_{K+1}) - F$ is large
- **Low temperature**
  - Only accept the new solutions where $f(X_{K+1}) - F$ is small

Simulated Annealing

- **Simulated annealing is particularly developed for unconstrained optimization**

- **Constrained optimization can be converted to unconstrained optimization using barrier method**

$$\begin{array}{ll} \min\limits_X & f(X) \\ \text{S.T.} & g(X) \le 0 \end{array} \quad \Rightarrow \quad \min_X \; f(X) - \frac{1}{t} \cdot \log[-g(X)]$$

Simulated Annealing

- **Simulated annealing does not guarantee global optimum**
  - However, it tries to avoid a large number of local minima
  - Therefore, it often yields a better solution than local optimization

- **Simulated annealing is not deterministic**
  - Whether accept or reject a new solution is random
  - You can get different answers from multiple runs

- **Simulated annealing is more expensive than local optimization**
  - It is the price you must pay to achieve a better optimal solution

Simulated Annealing

- **Simulated annealing has been used to solve many practical engineering problems**

- **A large number of implementation issues must be considered for practical circuit optimization problems**
  - How to define optimization variable X (continuous vs. discrete)?
  - How to randomly move to a new solution?
  - Etc.

Example: Travelling Salesman Problem (TSP)

- **N cities are located on a 2-D map**

- **One must visit each city once and then return to start city**

- **Find the optimal route with minimum length**
  - If all cities are visited in the order of $R = \{C_1, C_2, ..., C_N\}$, we have

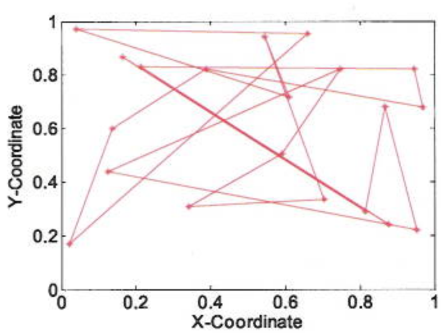
$$f(R) = \underbrace{\|C_1 - C_2\|_2}_{\text{Distance between } C_1 \text{ and } C_2} + \|C_2 - C_3\|_2 + \cdots + \underbrace{\|C_N - C_1\|_2}_{\text{Distance between } C_N \text{ and } C_1}$$

Example: Travelling Salesman Problem (TSP)

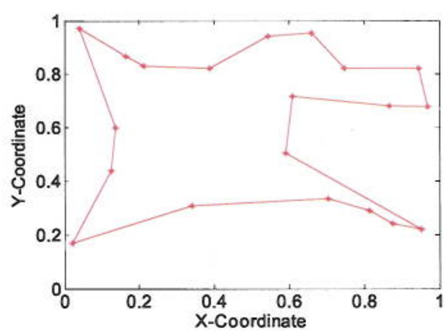
- **Step 1: start from random route R, initial temperature T & K = 1**
- **Step 2: evaluate cost function F = f(R)**
- **Step 3: define new route R_K by randomly swapping two cities**
- **Step 4: if $f(R_K) < F$, then**
  - Accept new route
  - $R = R_K$ & $F = f(R_K)$
- **End if**
- **Step 5: if $f(R_K) \geq F$, then**
  - Accept new solution with certain probability
  - $R = R_K$ & $F = f(R_K)$ iff rand(1) < $\exp\{[F - f(R_K)]/T\}$
- **End if**
- **Step 6: $T = \alpha T$ ($\alpha < 1$), $K = K + 1$, and go to Step 3**

Example: Travelling Salesman Problem (TSP)

- **TSP route optimized by simulated annealing**



Initial route



Optimized route

Summary

- **Stochastic optimization**
  - Simulated annealing