Numerical Optimization

David Levin

Plan for Today

- A fast and furious tour through numerical optimization
  - Unconstrained Optimization
    - Gradient Descent
    - Newton's Method
  - Constrained Optimization
    - Newton's Method
    - Quadratic Programming

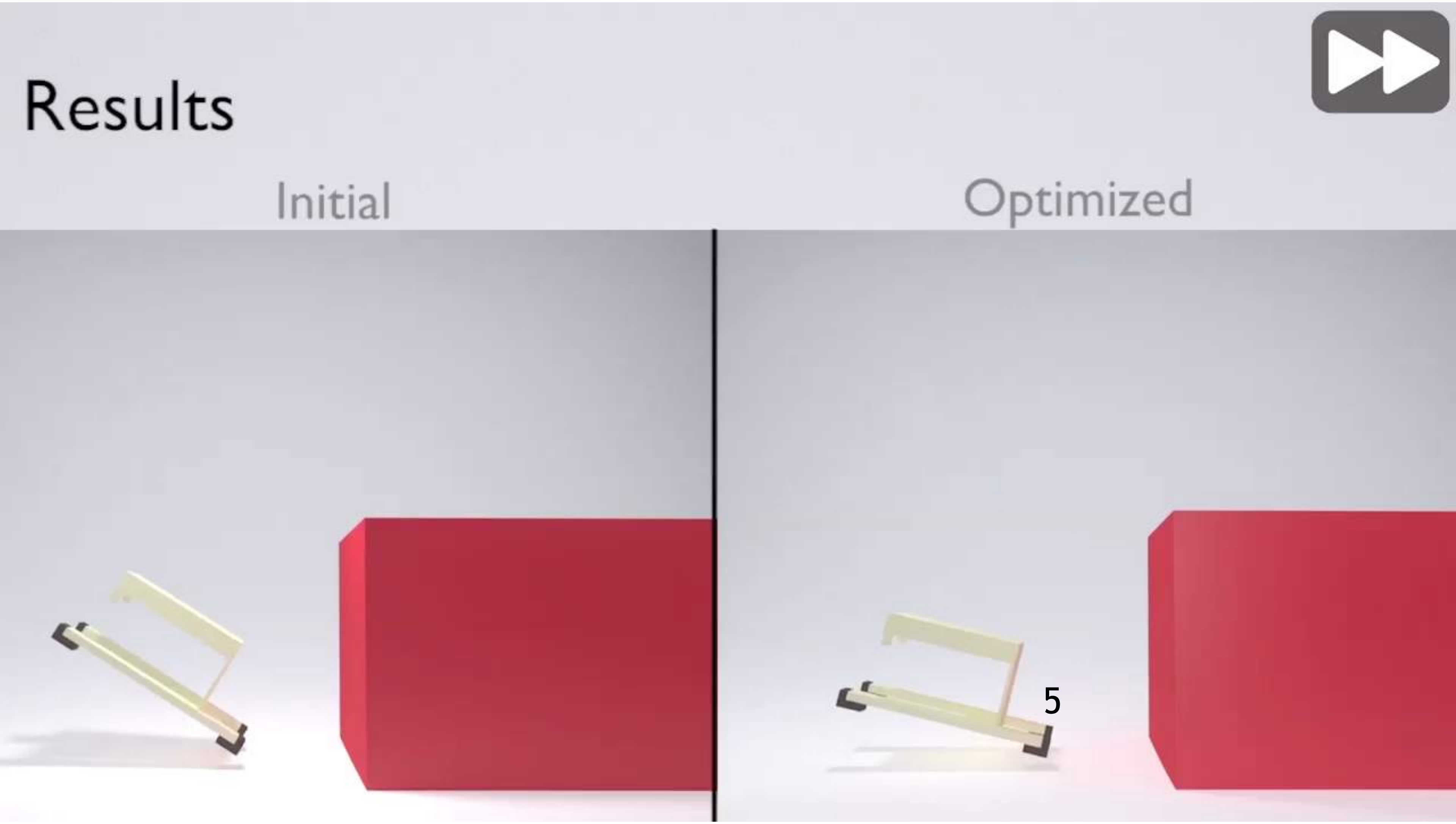
Plan for Today

- A fast and furious tour through numerical optimization
  - Discrete Optimization
    - Simulated Annealing
    - Branch and Bound

Warning

- Learning about optimization is a contact sport
- There will be math than (not too hard though!)

Example of a Design Optimization



Introduction to Optimization

- Optimization involves finding an “optimal value”

- i.e. Maximizing a profit, minimizing an area etc…

$$\underbrace{\boxed{\min}}_{\text{minimize}}\ \overbrace{\boxed{f\left(x\right)}}^{\text{Cost Function}}$$

Introduction to Optimization

- Optimization involves finding an "optimal value"
- i.e. Maximizing a profit, minimizing an area etc…

$$\boxed{x^*} = \arg\min f(x)$$

Optimal Solution

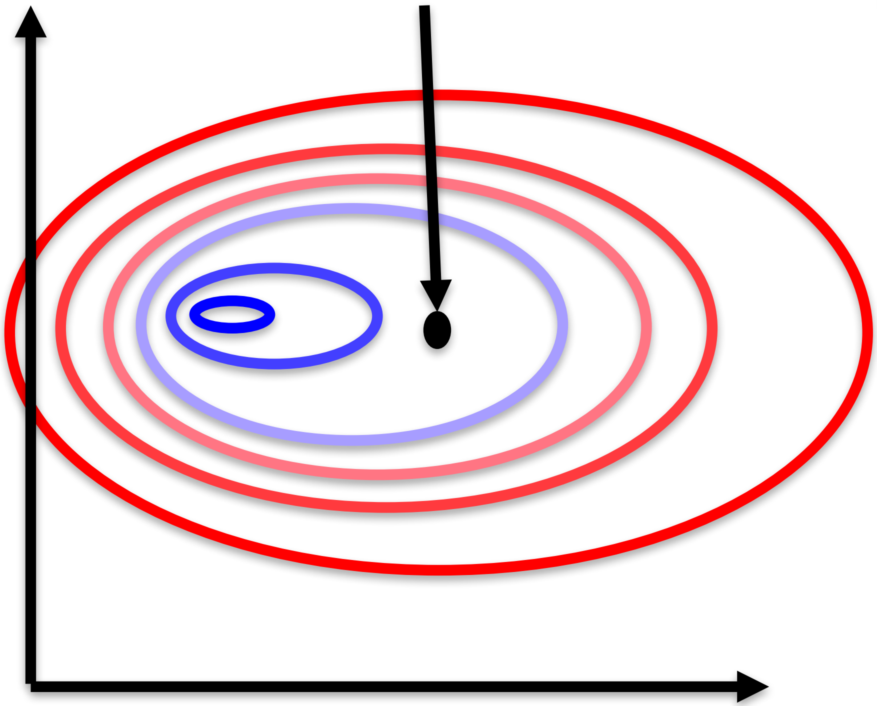
Types of Optimization

- Continuous vs. Discrete
- Constrained vs. Unconstrained

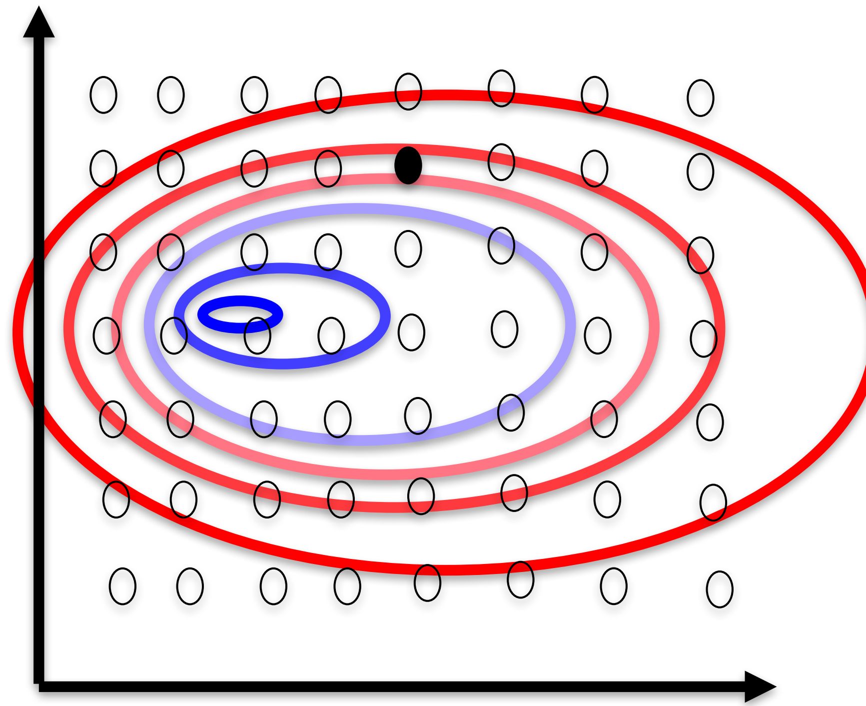
Continuous

Discrete

This point can move smoothly

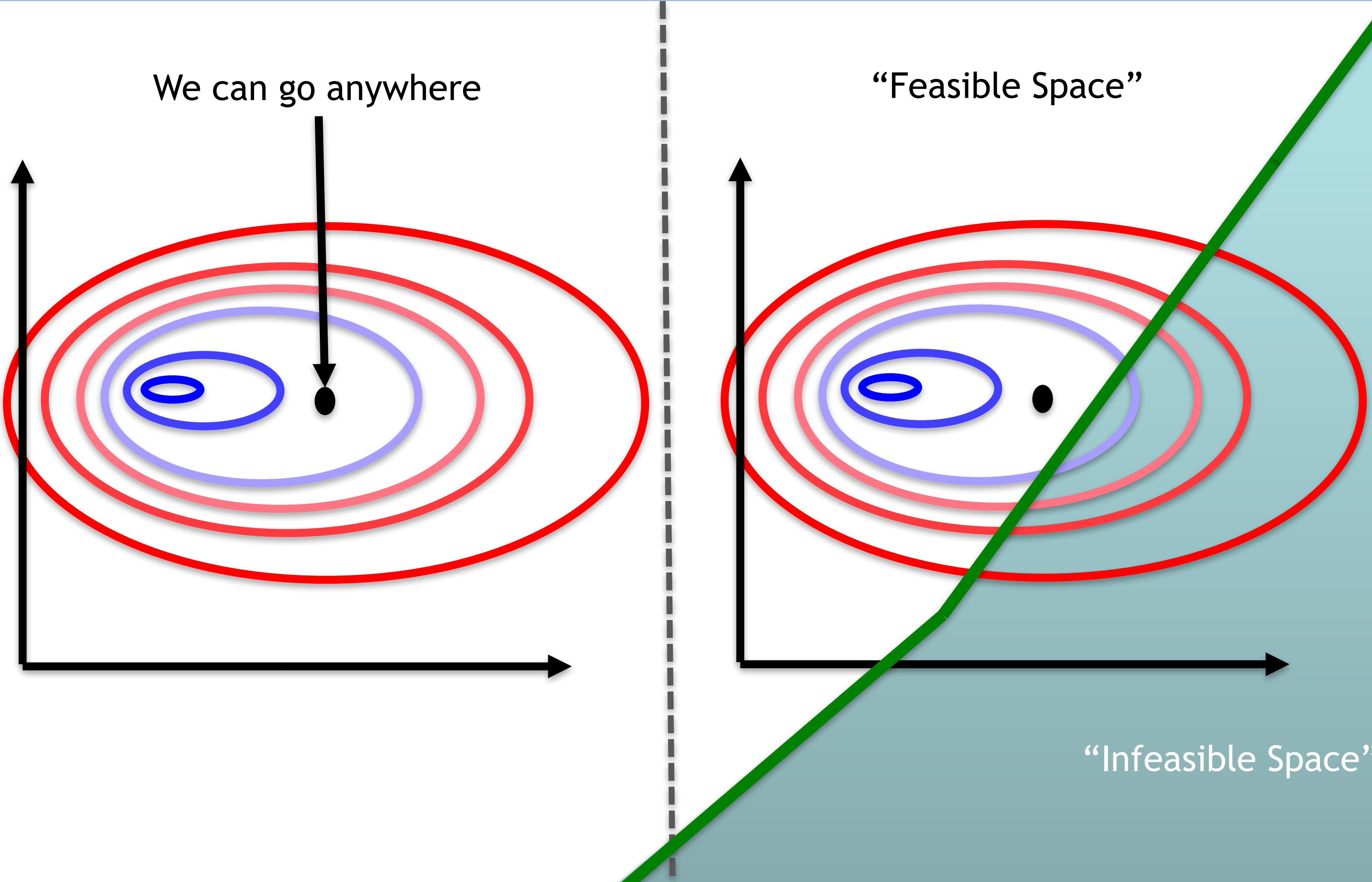


Choose from discrete points in parameter space



Unconstrained

Constrained



Types of Optimization

- $\boxed{\text{Continuous}}$ vs. Discrete
- Constrained vs. $\boxed{\text{Unconstrained}}$

Continuous Optimization

- We're solving

$$x^{*} = \arg\min f\left(x\right)$$

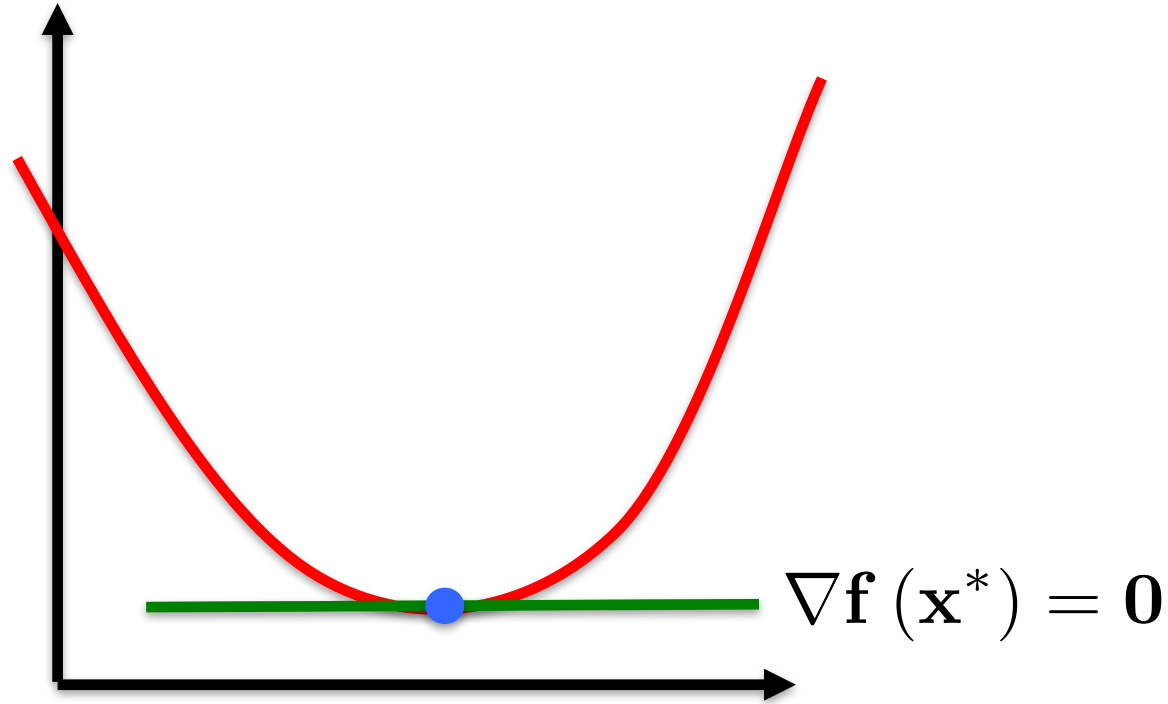
- How do we know we've found a potential solution?

IMPORTANT!!!!!!

$$\nabla \mathbf{f}\left(\mathbf{x}^{*}\right) = \mathbf{0}$$

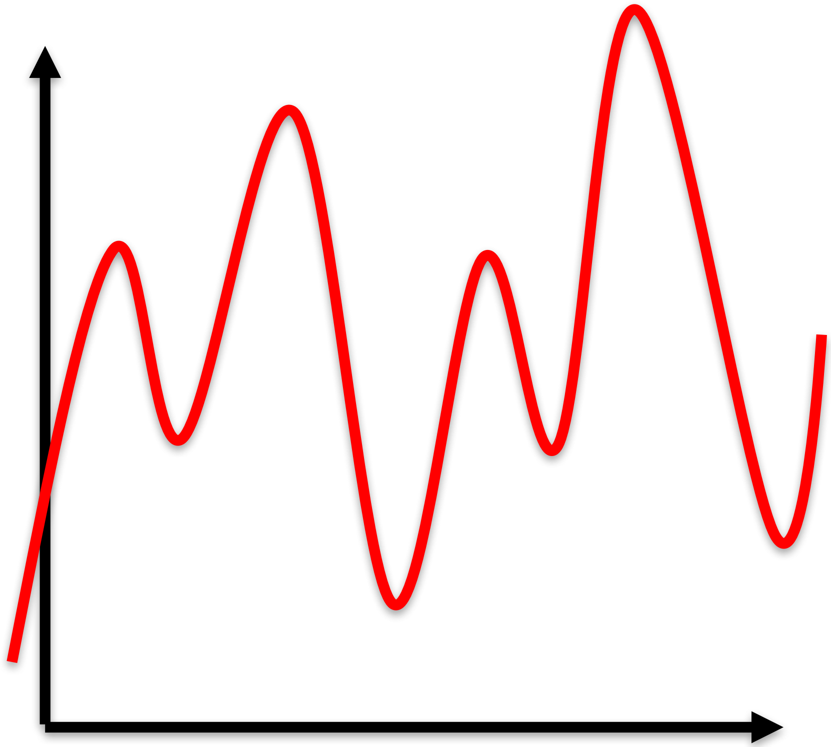
Potential Solutions

- Intuitively we look for a flat point on the cost function



Potential Solutions

- Sometimes that’s easier said than done

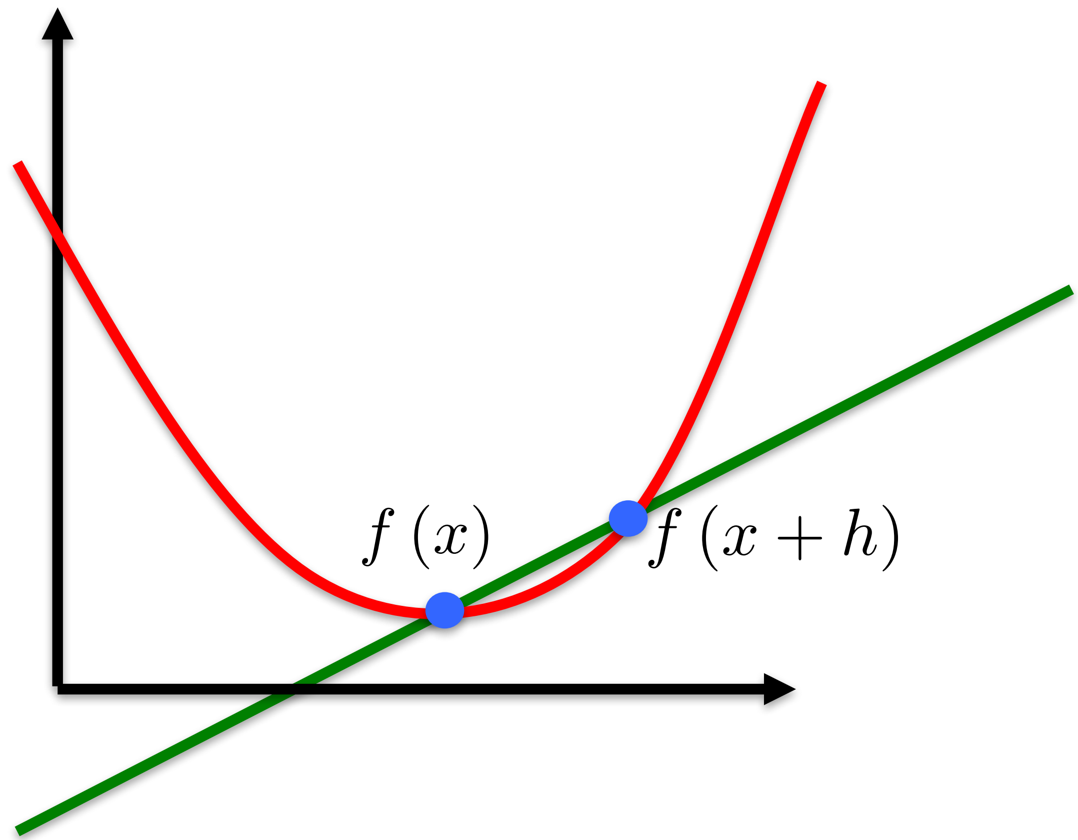


Computing the Gradient

- Analytically (pencil, paper, mathematica)
- Finite Differences

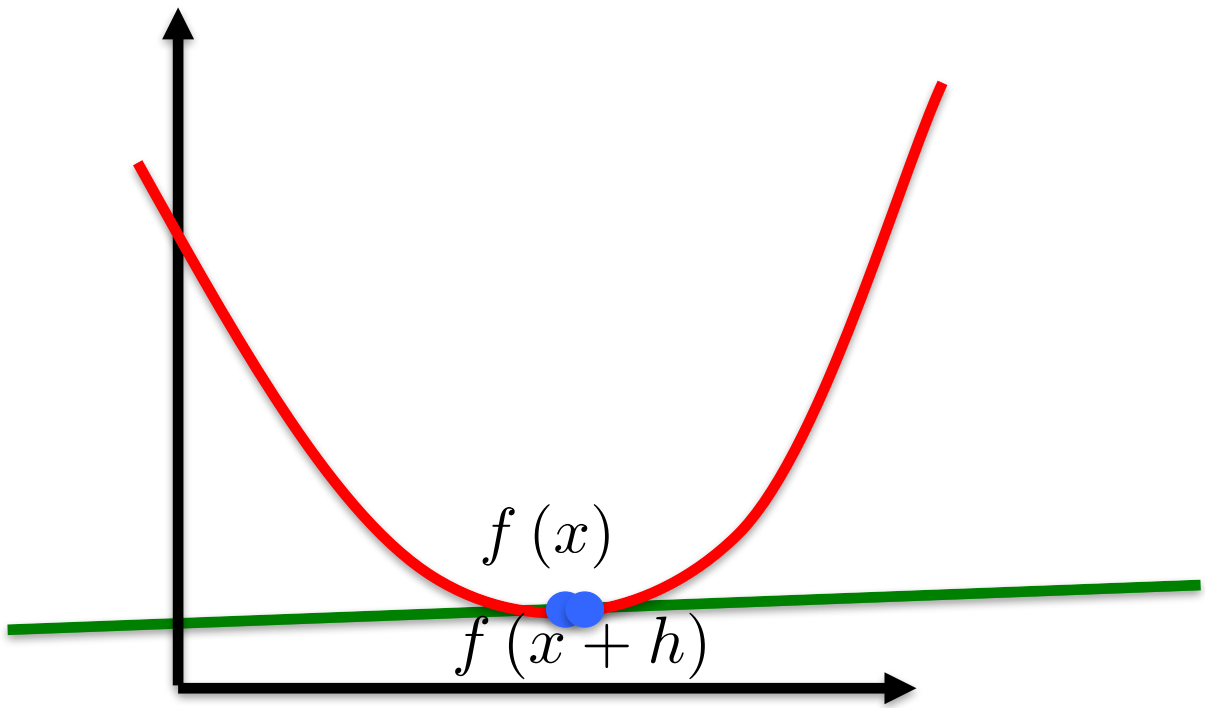
Potential Solutions

- Computing the gradient requires a limit



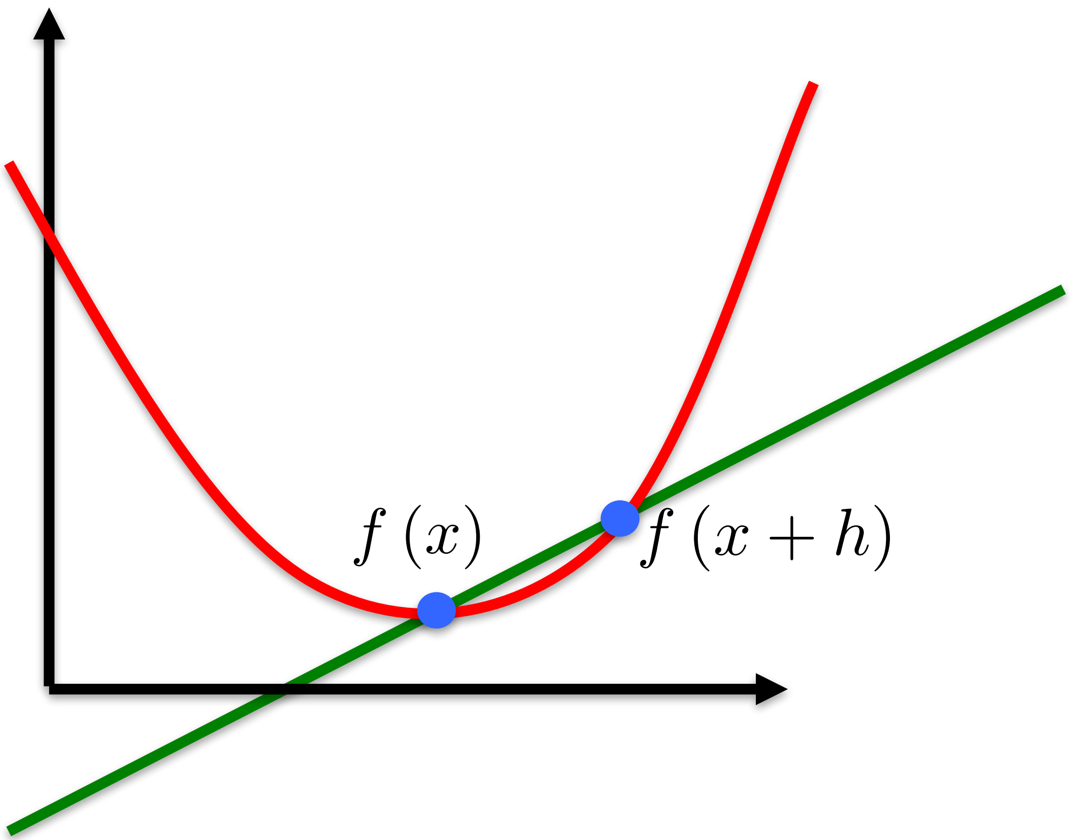
Potential Solutions

- Computing the gradient requires a limit



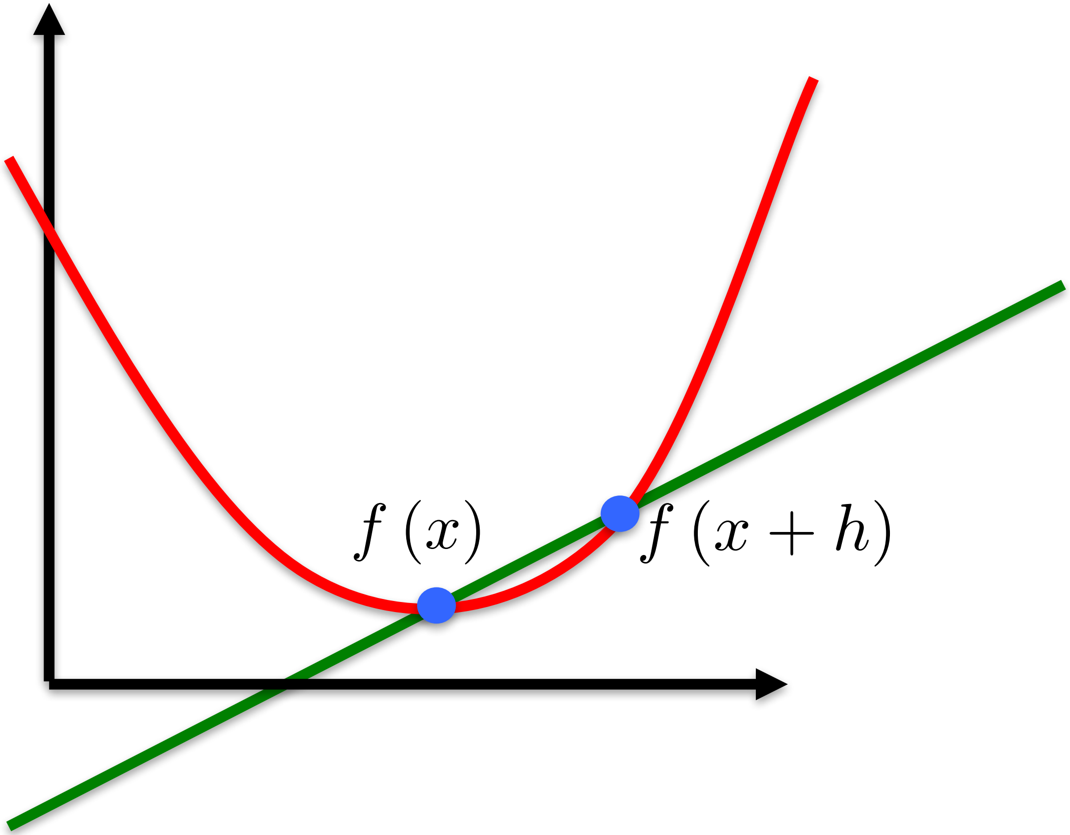
Potential Solutions

- In Finite Differencing we choose h and evaluate $\frac{1}{h} f\left(x+h\right) - f\left(x\right)$ numerically



Potential Solutions

- Matlab does this automatically which makes it easy to test out optimizations



Continuous Optimization

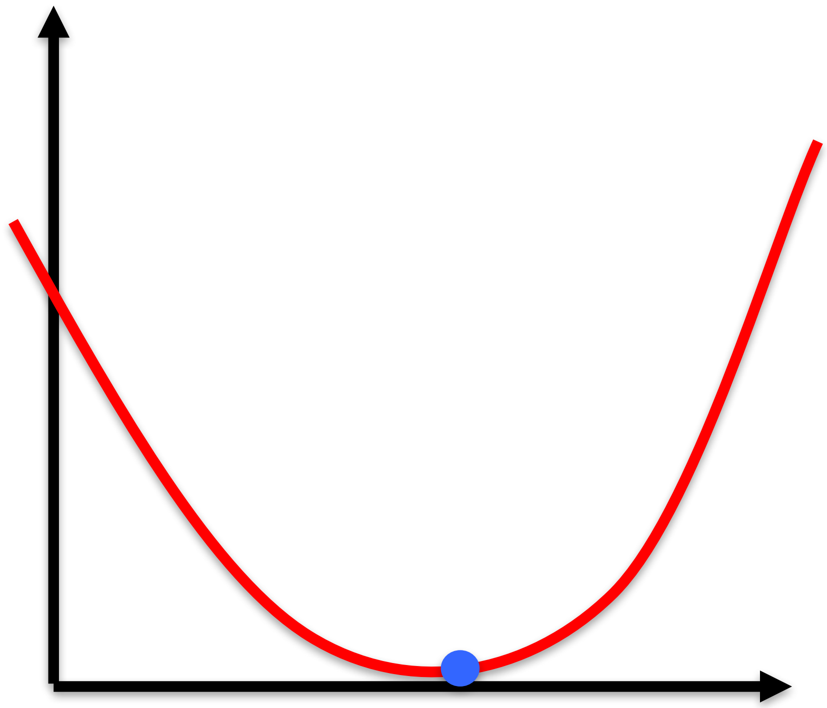
- General, continuous optimizations are difficult to solve
- We focus on certain classes of problems that are solvable

Convex Optimization

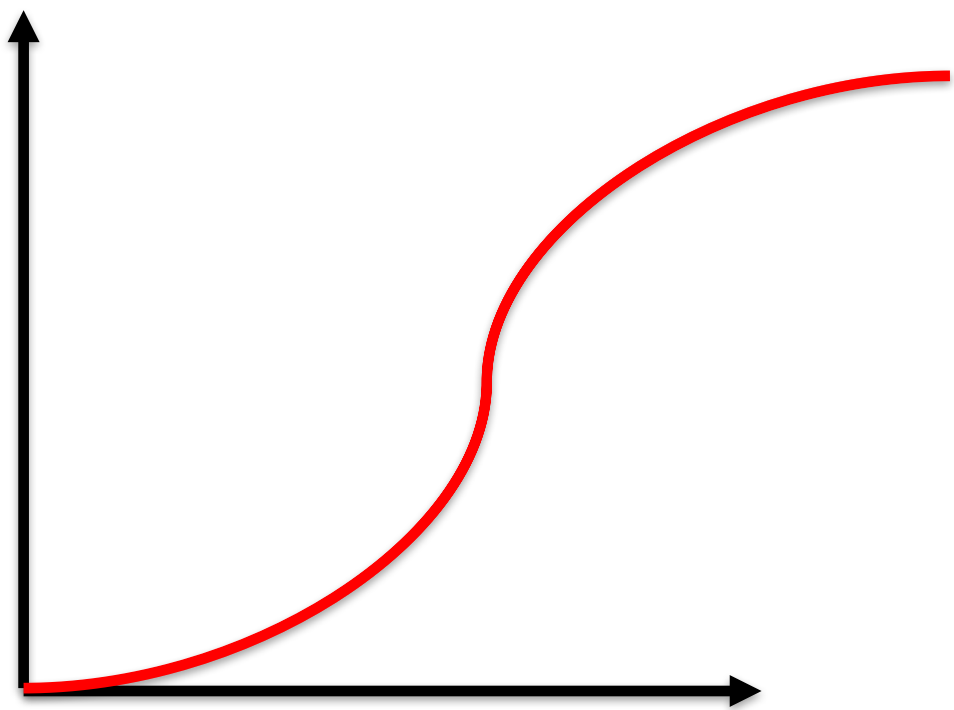
Convex Optimization

- Convex optimizations are ones that have a single minimum
- Let’s look at some examples of convex cost functions

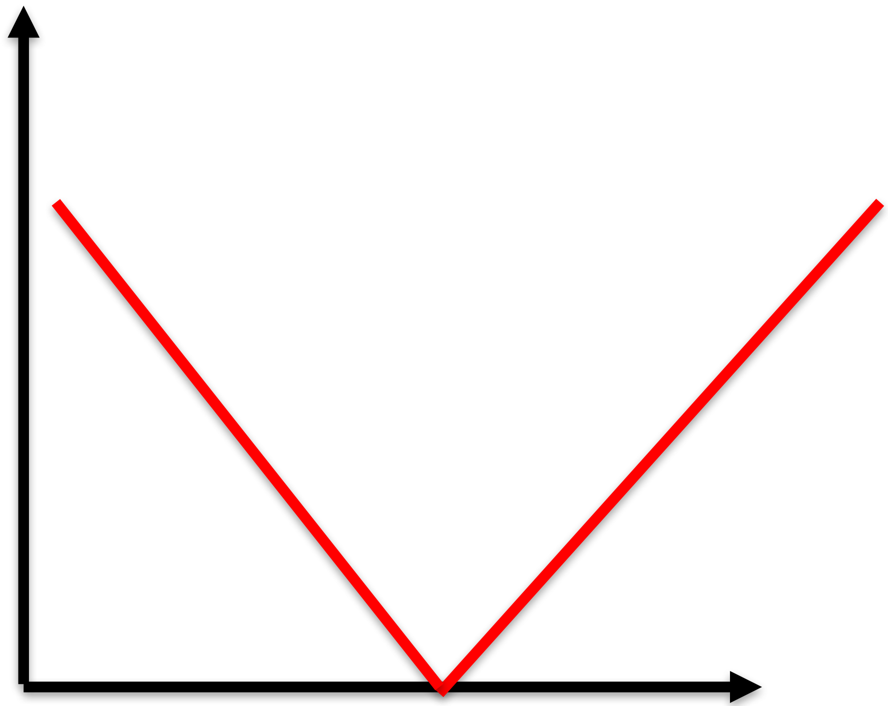
Convex Optimization



Is This Convex ?

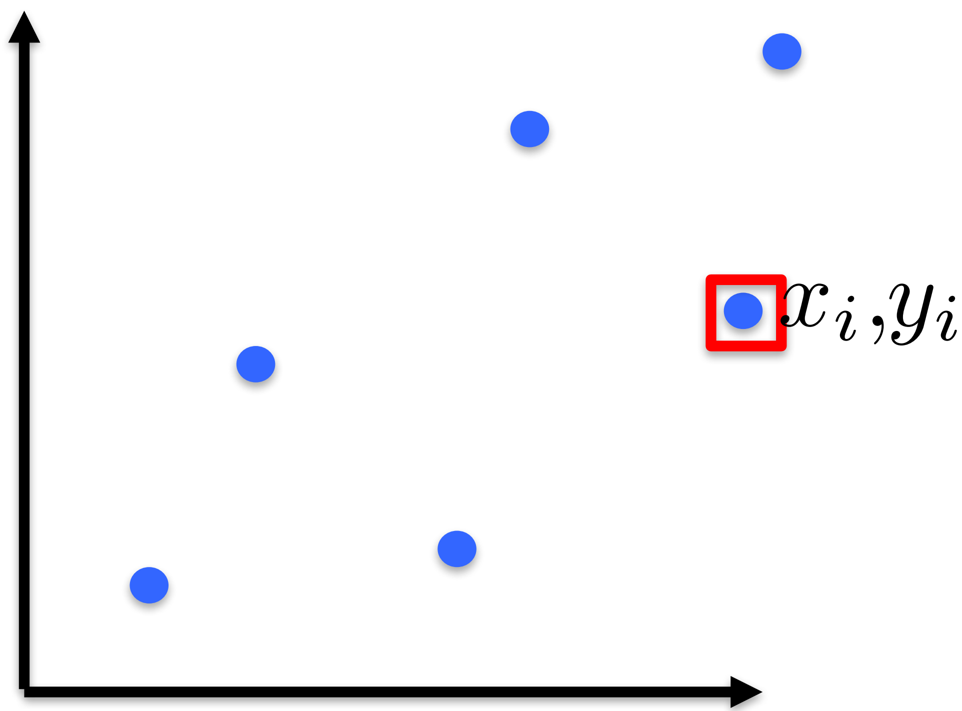


Is This Convex ?



A Simple Example: Least Squares

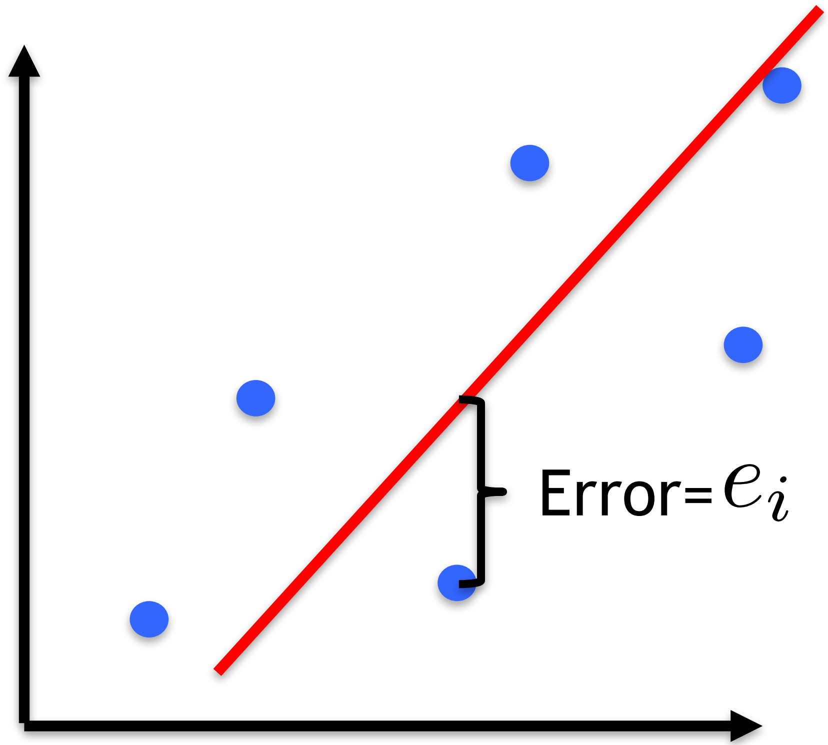
- Least Squares Fitting of a Curve



- Want to find a line, mx + c, that is a “best fit”

A Simple Example: Least Squares

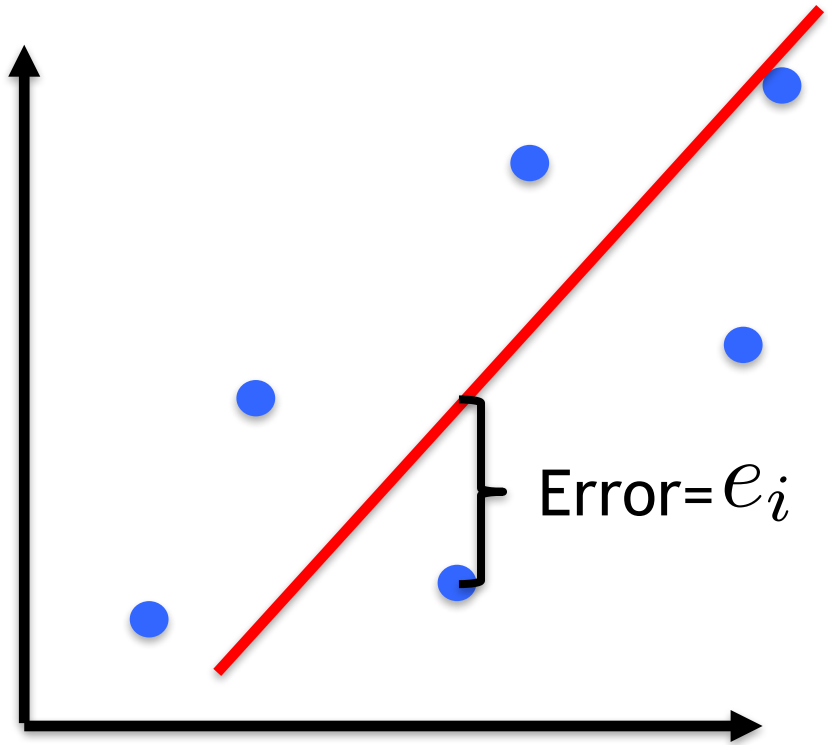
- Least Squares Fitting of a Curve



- Want to find a line, mx + c, that is a “best fit”

A Simple Example: Least Squares

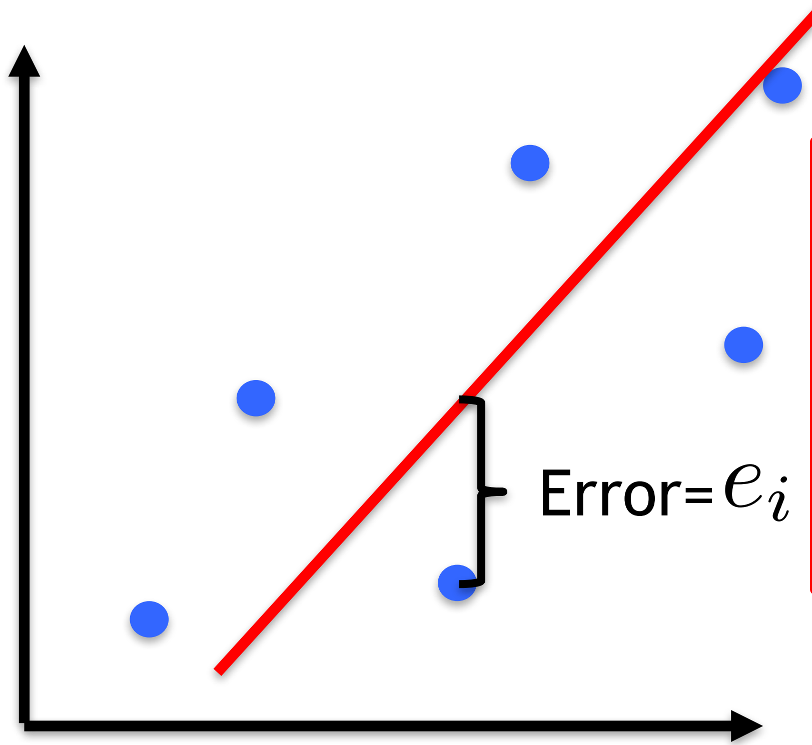
- Least Squares Fitting of a Curve



$$e_i = y_i - mx_i - c$$

A Simple Example: Least Squares

- Minimize sum of squared errors



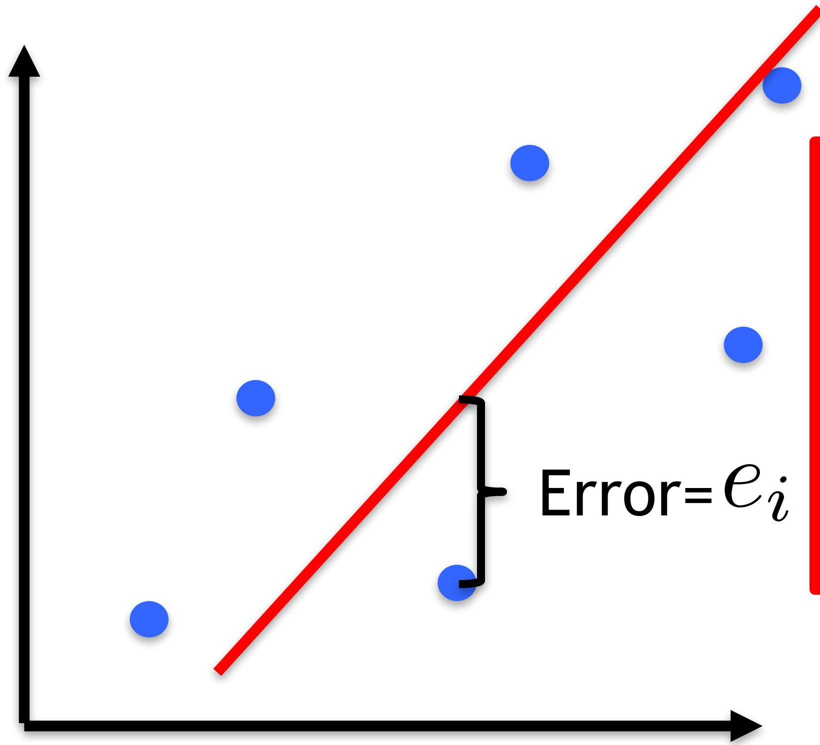
$$\mathbf{A} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix}$$

Total Error = $\|\mathbf{A}\theta - \mathbf{y}\|^2$

A Simple Example: Least Squares

- Minimize sum of squared errors

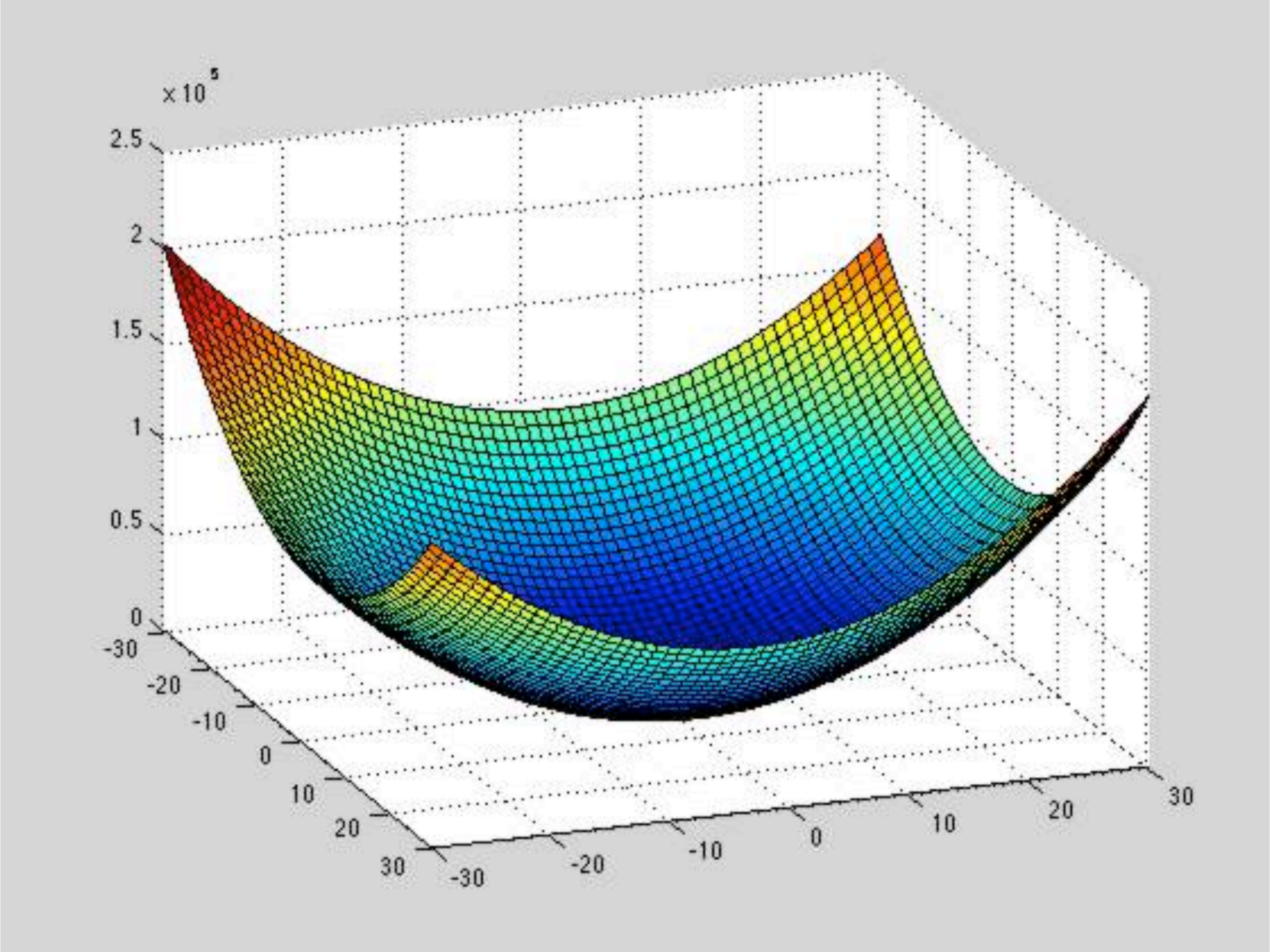
$$\theta = \begin{pmatrix} m \\ c \end{pmatrix}$$



$$\mathbf{A} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix}$$

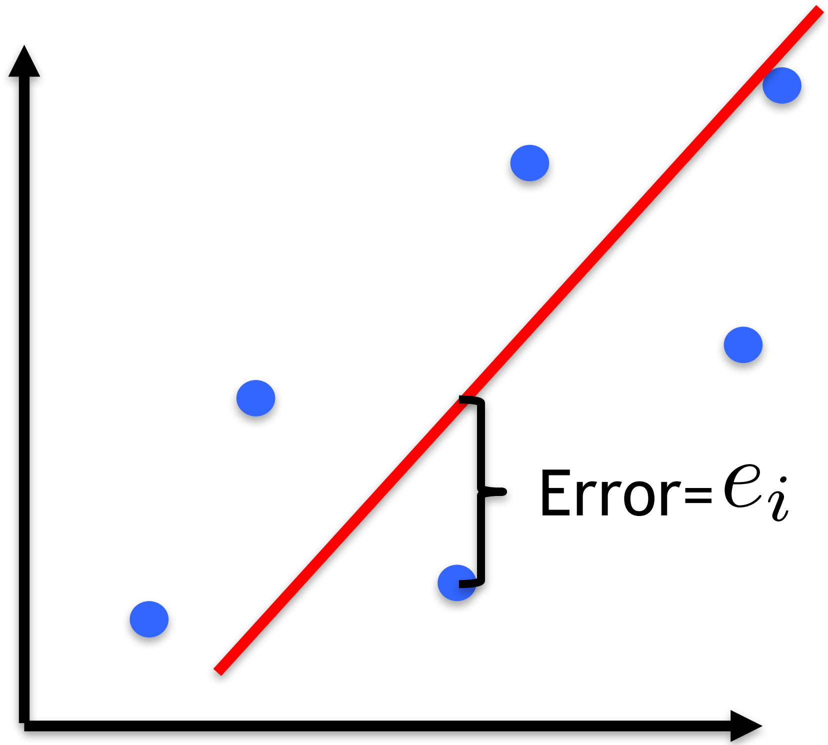
Sum of Squared Error = f(x) = $||\mathbf{A}\theta - \mathbf{y}||^2$

Simple Example: Least Squares



A Simple Example: Least Squares

- Solution is the normal equations



Solution = $\boxed{\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}}$ $\nabla\mathbf{f}\left(\mathbf{x}^*\right) = \mathbf{0}$

Descent Algorithms

- Used when cost function is more complicated
- Idea: Follow search directions that reduce the cost!
- Two Types
  - Gradient Descent
  - Newton's Method

Gradient Descent

- Recall that the gradient of a function is given by

$$\nabla f\left(\mathbf{x}\right) = \begin{pmatrix} \frac{\partial f}{\partial \mathbf{x}_1} & \frac{\partial f}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f}{\partial \mathbf{x}_n} \end{pmatrix}$$

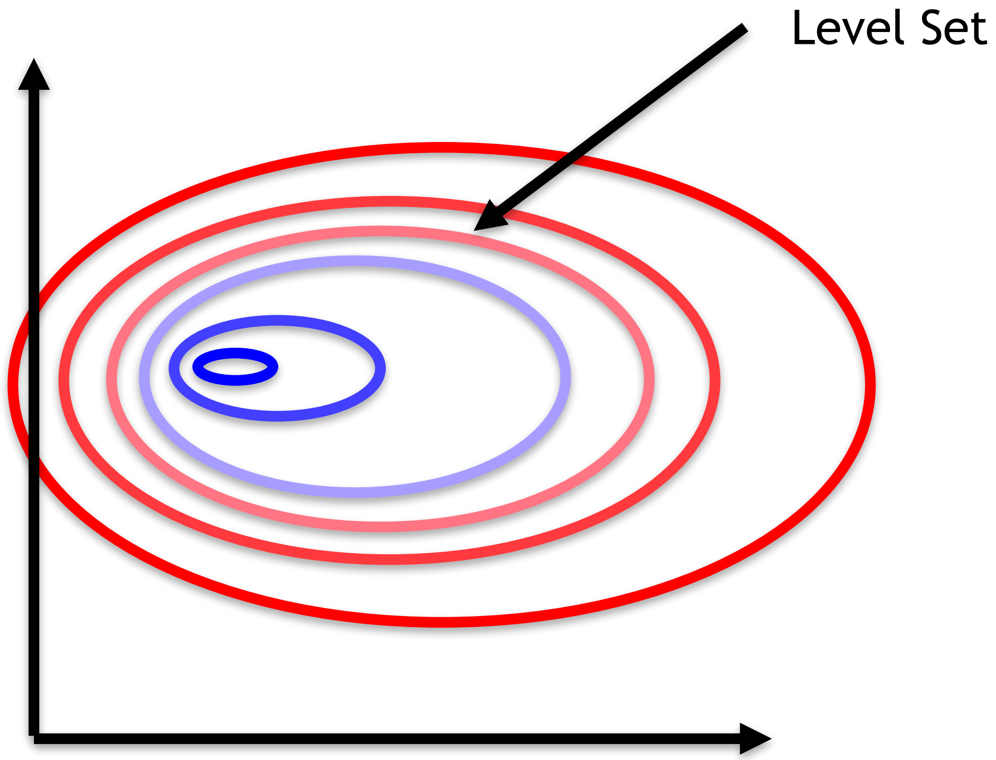
Gradient Descent

- Recall that the gradient of a function is given by

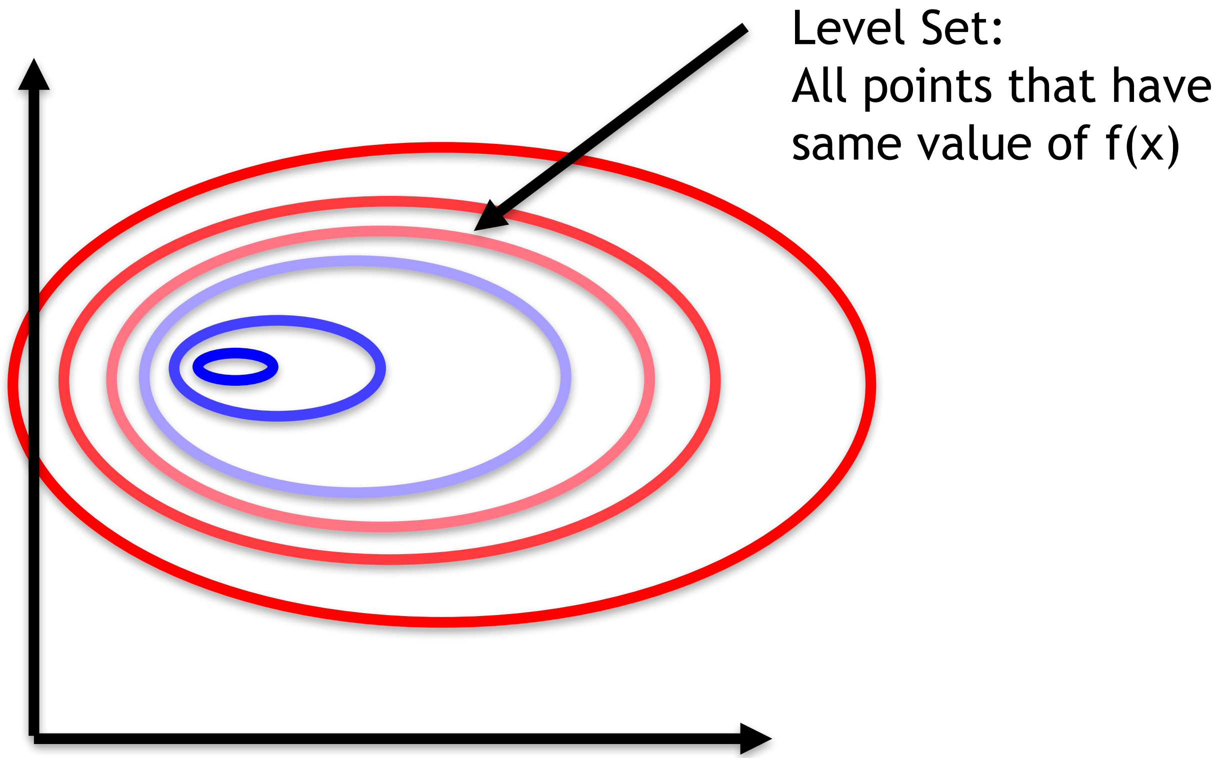
$$\nabla f\left(\mathbf{x}\right) = \begin{pmatrix} \frac{\partial f}{\partial \mathbf{x}_1} & \frac{\partial f}{\partial \mathbf{x}_2} & \cdots & \frac{\partial f}{\partial \mathbf{x}_n} \end{pmatrix}$$

- Points in direction of maximum ascent

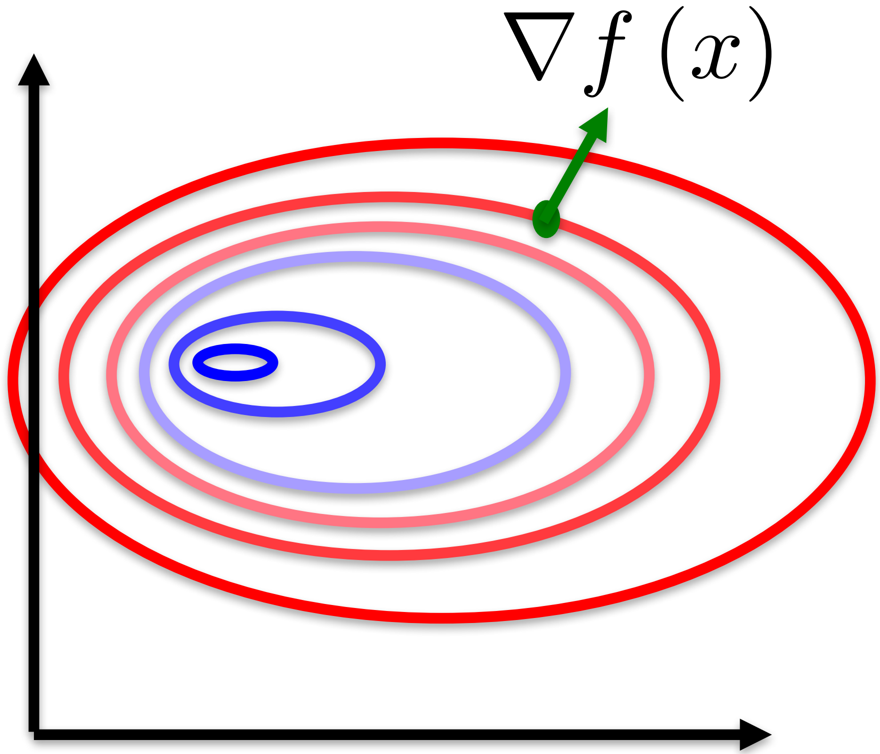
An Aside: Level Sets



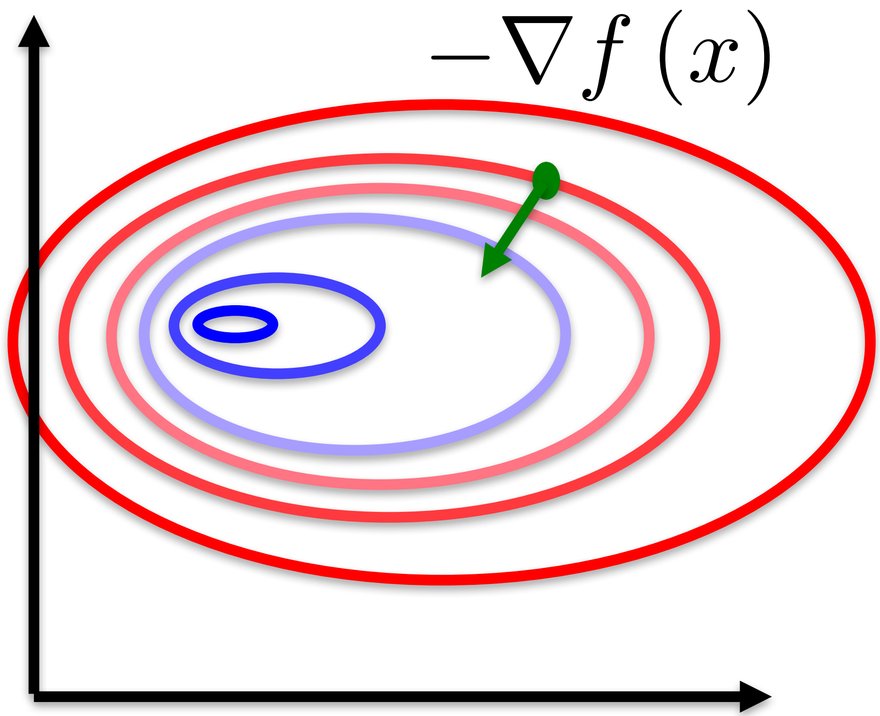
An Aside: Level Sets



Gradient Descent



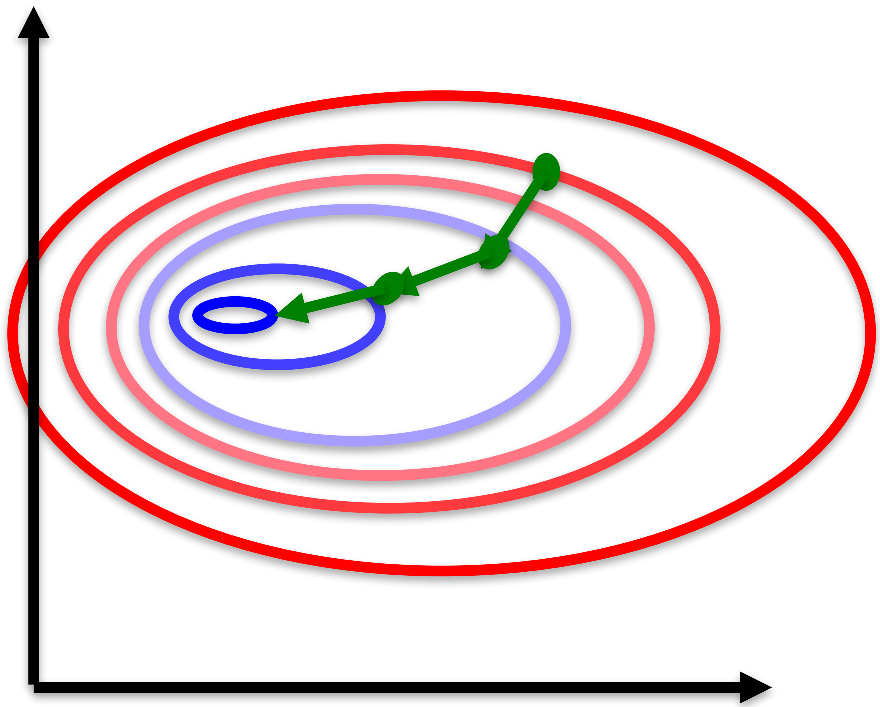
Gradient Descent



Simple Gradient Descent Algorithm

- While not at an optimal point
  - Compute the gradient at current point (x)
  - Move to new point $x = x - \boxed{h} \nabla f\left(x\right)$

Gradient Descent



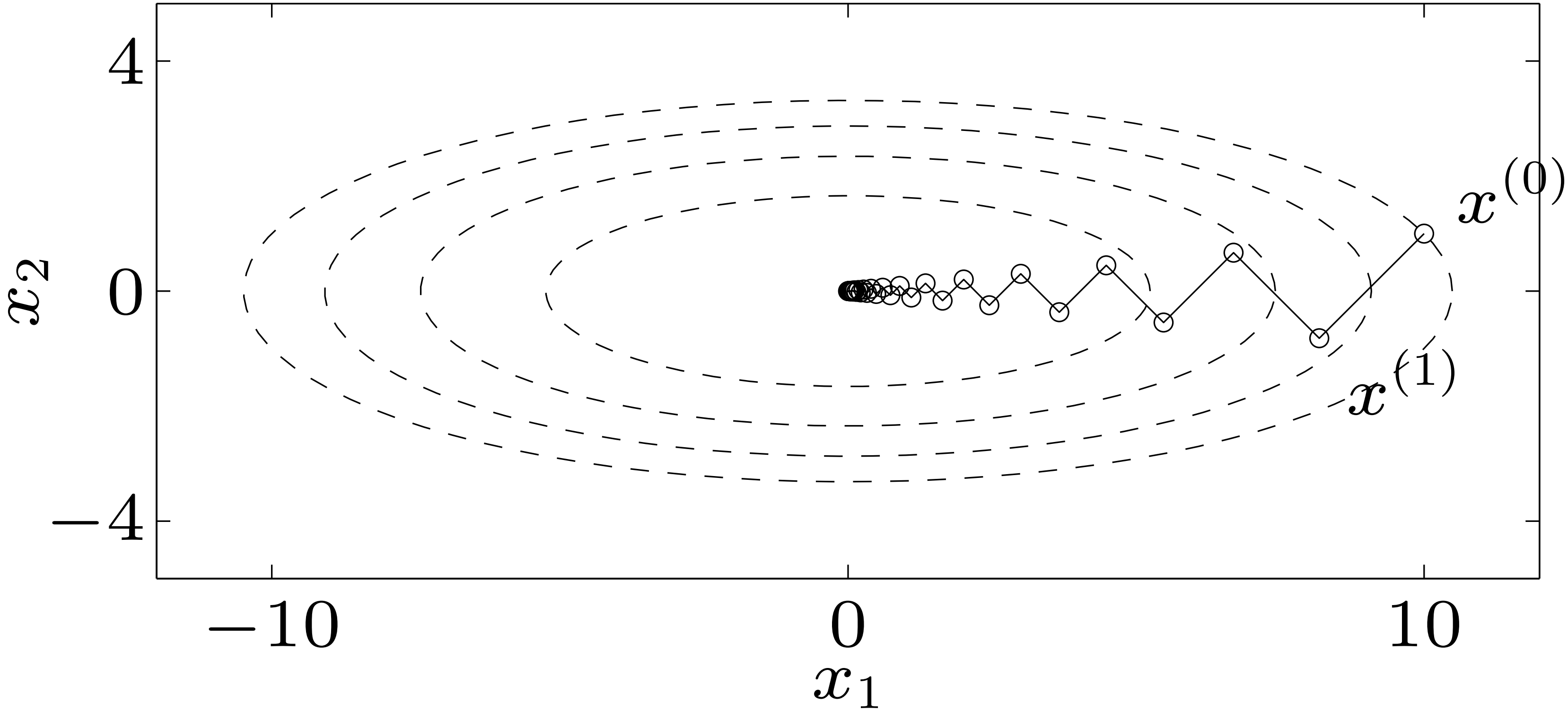
Simple Gradient Descent Algorithm

- While not at an optimal point
  - Compute the gradient at current point (x)
  - Move to new point $x = x - \boxed{h} \nabla f(x)$

Gradient Descent

- Good:
  - Simple to implement
- Bad:
  - Sometimes converges badly

Gradient Descent



Newton’s Method

- Can we choose better search directions ?
- This is the goal of Newton’s Method
- Newton’s Method needs access to the “Hessian” of a function

An Aside: The Hessian

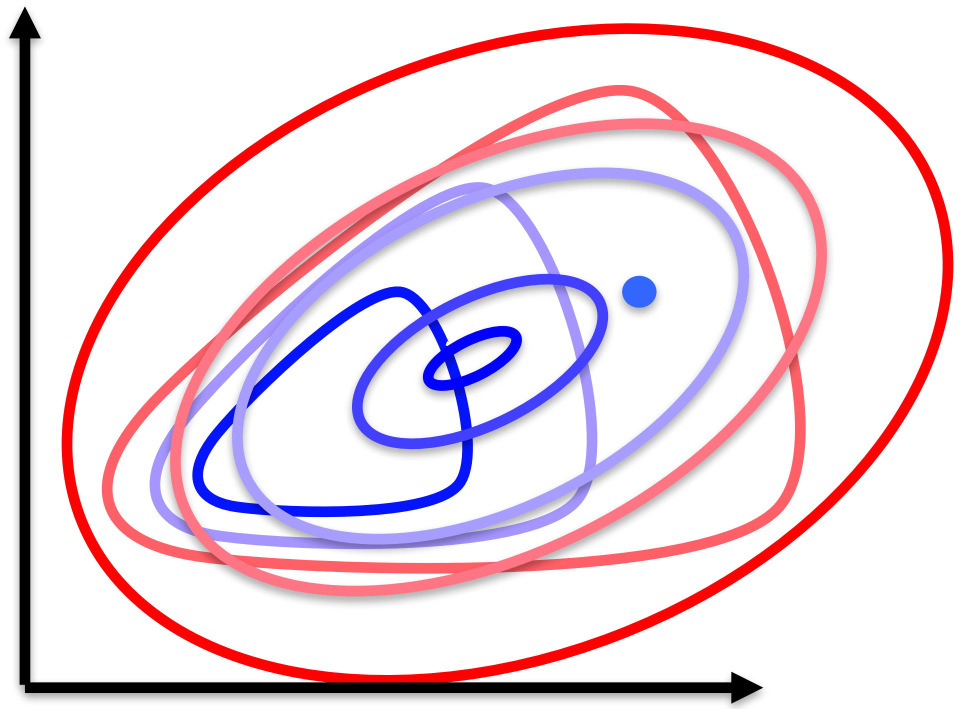
- The Hessian of a function $f(\mathbf{x})$

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Newton’s Method

- In Newton’s Method we approximate our function using a quadratic model
- Use that model to compute the best step length

Newton's Method



Choose best descent direction according to approximation

Newton's Method

- How do we get our approximation ?
- Taylor Expansion (we saw this in lecture 1)

$$f\left(\mathbf{x}^c + \Delta\mathbf{x}\right) \approx f\left(\mathbf{x}^c\right) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

$$\boxed{\nabla f\big|_{\mathbf{x}^c}}$$

$$\boxed{H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}}$$

Newton’s Method

- We minimize the model problem
- Find where the gradient is zero

$$f\left(\mathbf{x}^c\right) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

Newton's Method

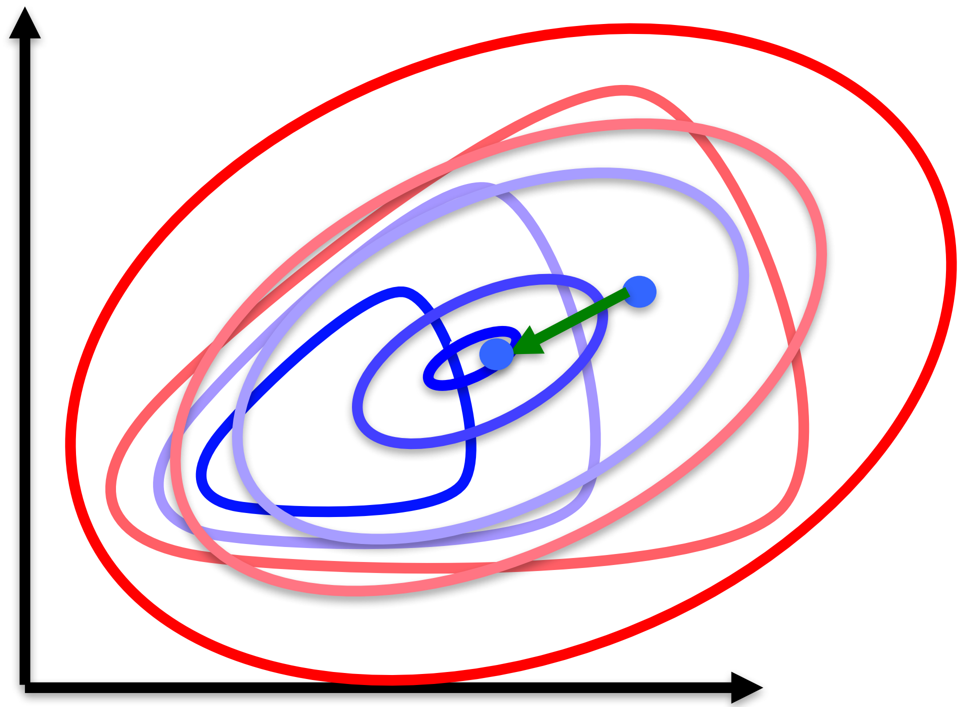
- We minimize the model problem
- Find where the gradient is zero

Model: $f\left(\mathbf{x}^c\right) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$

Gradient: $\mathbf{H}\Delta\mathbf{x} + \mathbf{g} = \mathbf{0}$

Increment: $\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}$

Newton's Method



Newton's Method

- Initialize $\mathbf{x}^c$
- While not at optimal point
  - Compute gradient (**g**) and Hessian (**H**)
  - Compute $\mathbf{x}^c = \mathbf{x}^c + h\Delta\mathbf{x}$
  - Update $\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}$

Aside: Hessian for Black Box Functions

- Centered Finite Differences:

$$\frac{\partial f}{\partial \mathbf{x}_i}(\mathbf{x}) \approx \frac{f(\mathbf{x} + \epsilon \mathbf{e}_i) - f(\mathbf{x} - \epsilon \mathbf{e}_i)}{2\epsilon}$$

- Second order accurate

The Hessian via Finite Differences

- Each entry of the Hessian: $\dfrac{\partial f^2}{\partial^2 \mathbf{x}}$

- Can apply finite differences twice to get a formula for each entry.

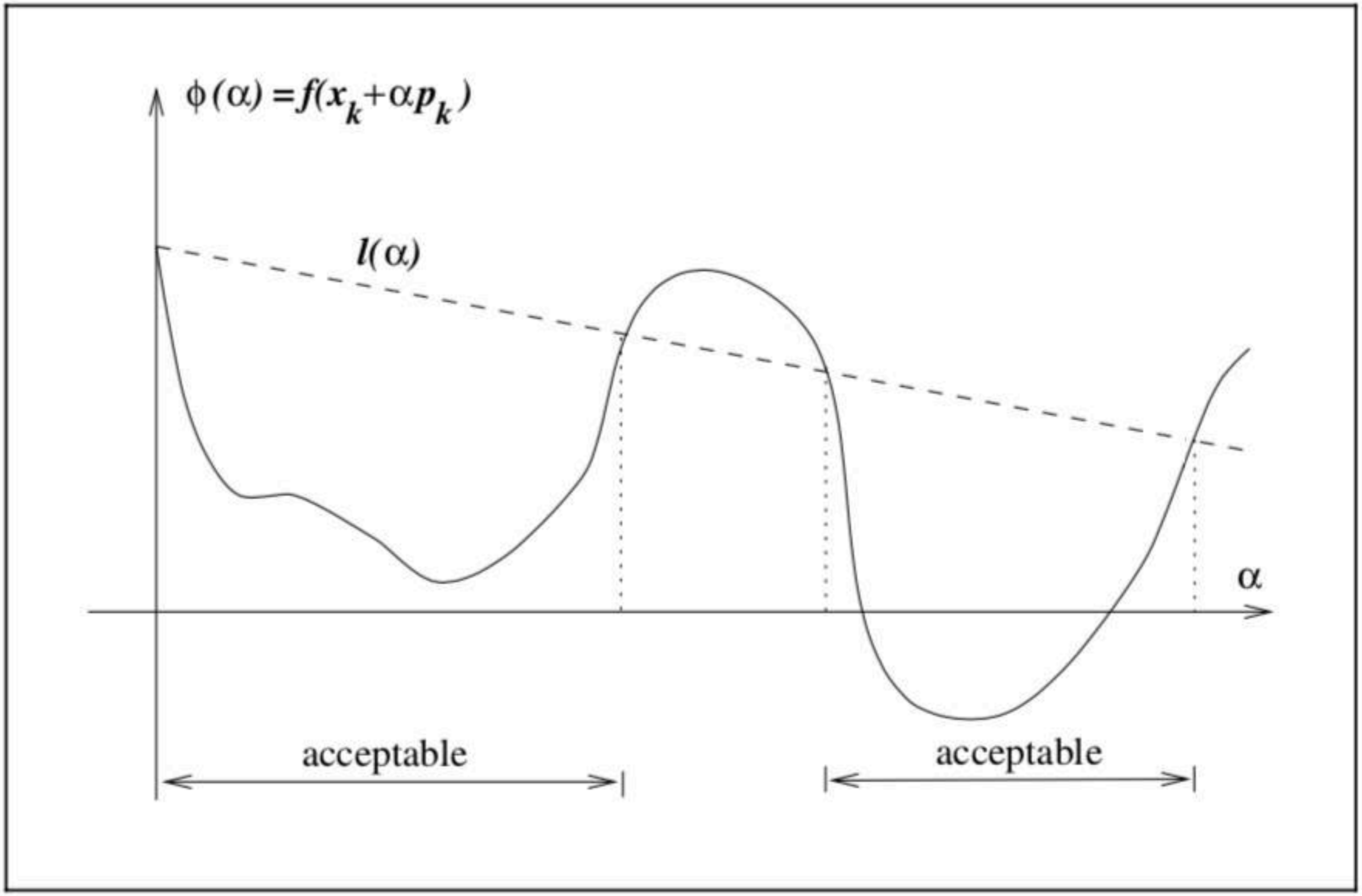
Gradient Descent vs. Newton's Method

- Gradient Descent is simpler
- Newton's Method converges faster, esp. near the solution
- Available Newton's Method Implementations:
  - MATLAB: fminunc
  - LBFGS: http://www.chokkan.org/software/liblbfgs/

Search Direction vs. Step Length

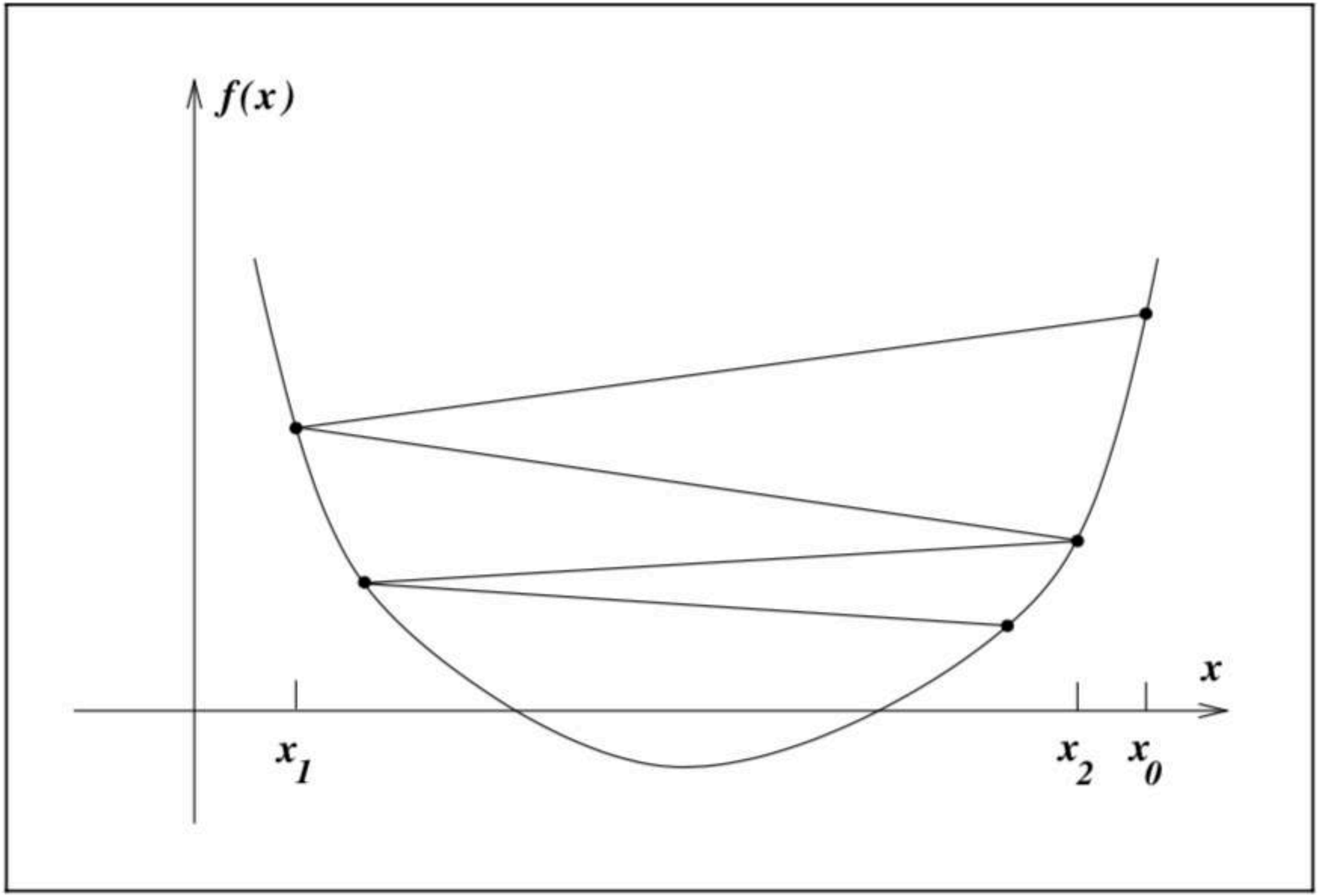
- While not at optimal point
  - Compute gradient (**g**) and Hessian (**H**)
  - Compute $\Delta \mathbf{x} =$ **?**
  - Update $\mathbf{x}^c = \mathbf{x}^c + h\Delta \mathbf{x}$

When Good Optimizations Go Bad



Numerical Optimization – Nocedal and Wright, pg. 33

When Good Optimizations Go Bad



Numerical Optimization – Nocedal and Wright, pg. 32

Choosing step length automatically

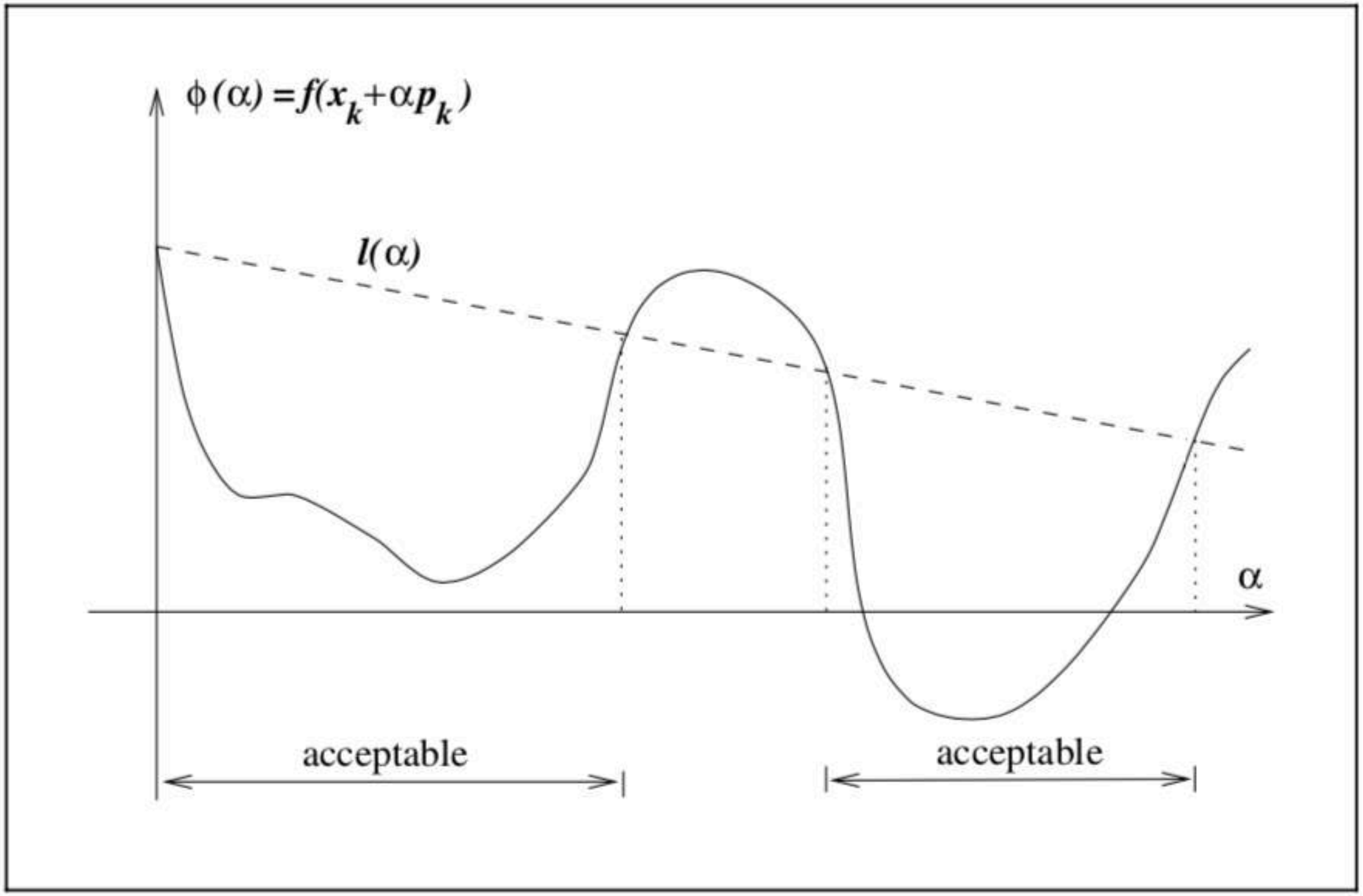
- While not at optimal point
  - Compute gradient (**g**) and Hessian (**H**)
  - Compute $\Delta\mathbf{x} =$ **?**
  - Update $\mathbf{x}^c = \mathbf{x}^c + h\Delta\mathbf{x}$

Characteristics of a Good Optimization Step

- We’ve seen that just guaranteeing a decreasing cost function is not enough
- We need sufficient decrease in the cost function
- How do we define sufficient decrease ?

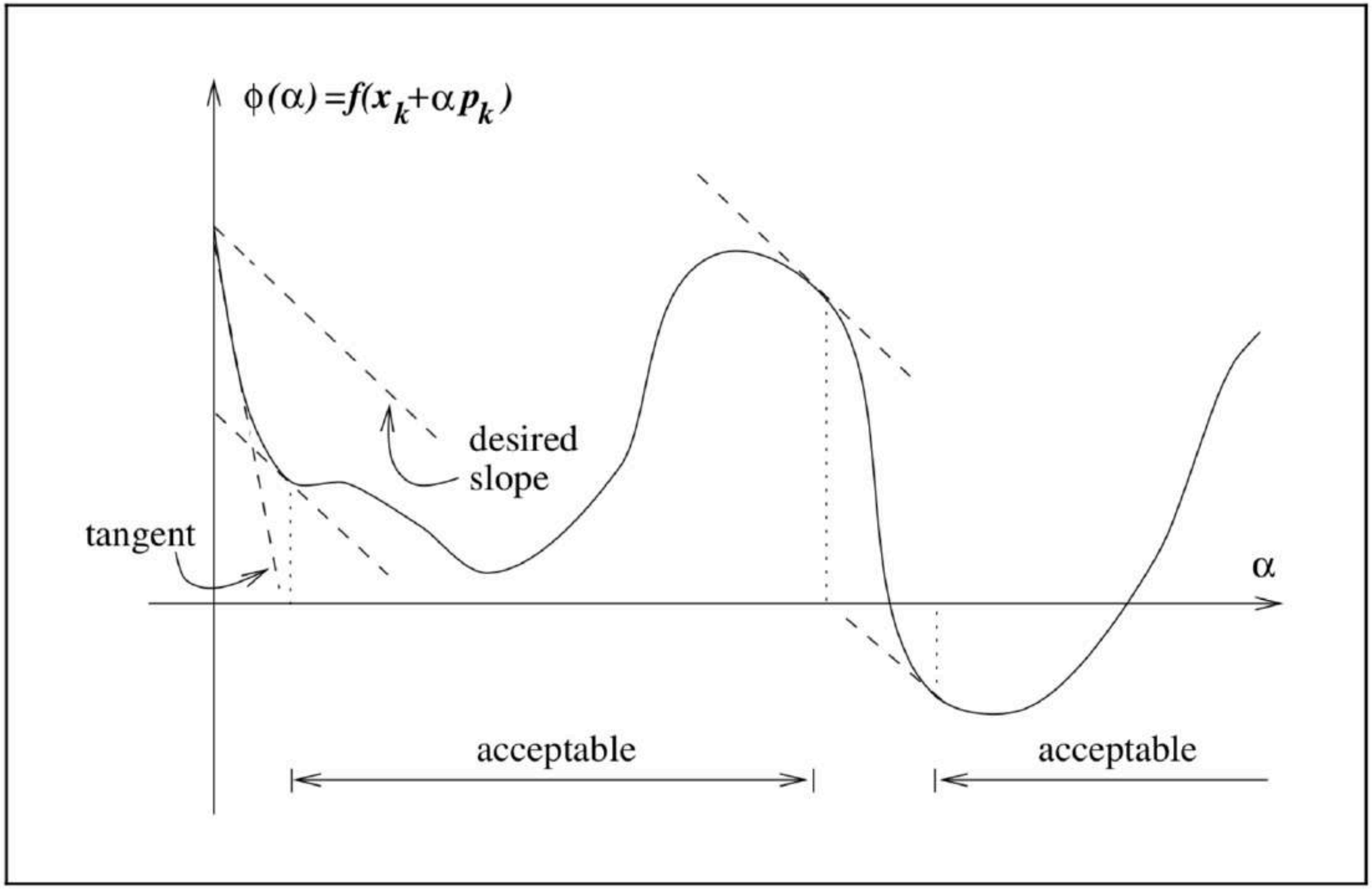
Sufficient Decrease

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k,$$



Curvature Condition

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k.$$



Wolfe Conditions

$$f(x_k + \alpha p_k) \le f(x_k) + c_1 \alpha \nabla f_k^T p_k,$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \ge c_2 \nabla f_k^T p_k,$$

- Typical values
  - C1=1e-8 to 1e-4
  - C2 = 0.1 to 0.9

Backtracking Line Search

Algorithm 3.1 (Backtracking Line Search).

Choose $\bar{\alpha} > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$; Set $\alpha \leftarrow \bar{\alpha}$;

repeat until $f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f_k^T p_k$

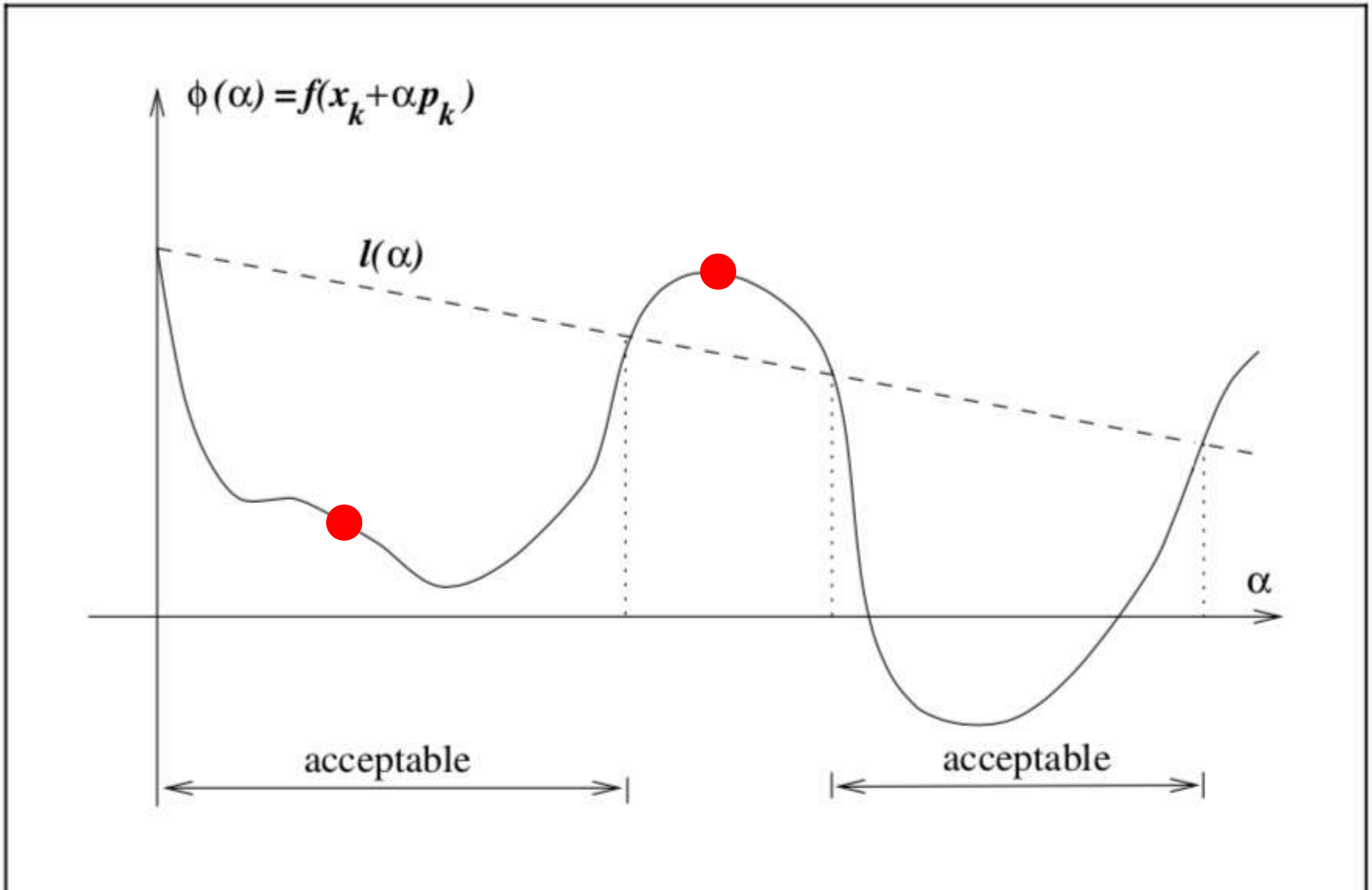
$\alpha \leftarrow \rho\alpha$;

end (repeat)

Terminate with $\alpha_k = \alpha$.

Numerical Optimization – Nocedal and Wright, pg. 37

Backtracking Line Search



Examples of Optimization in Engineering

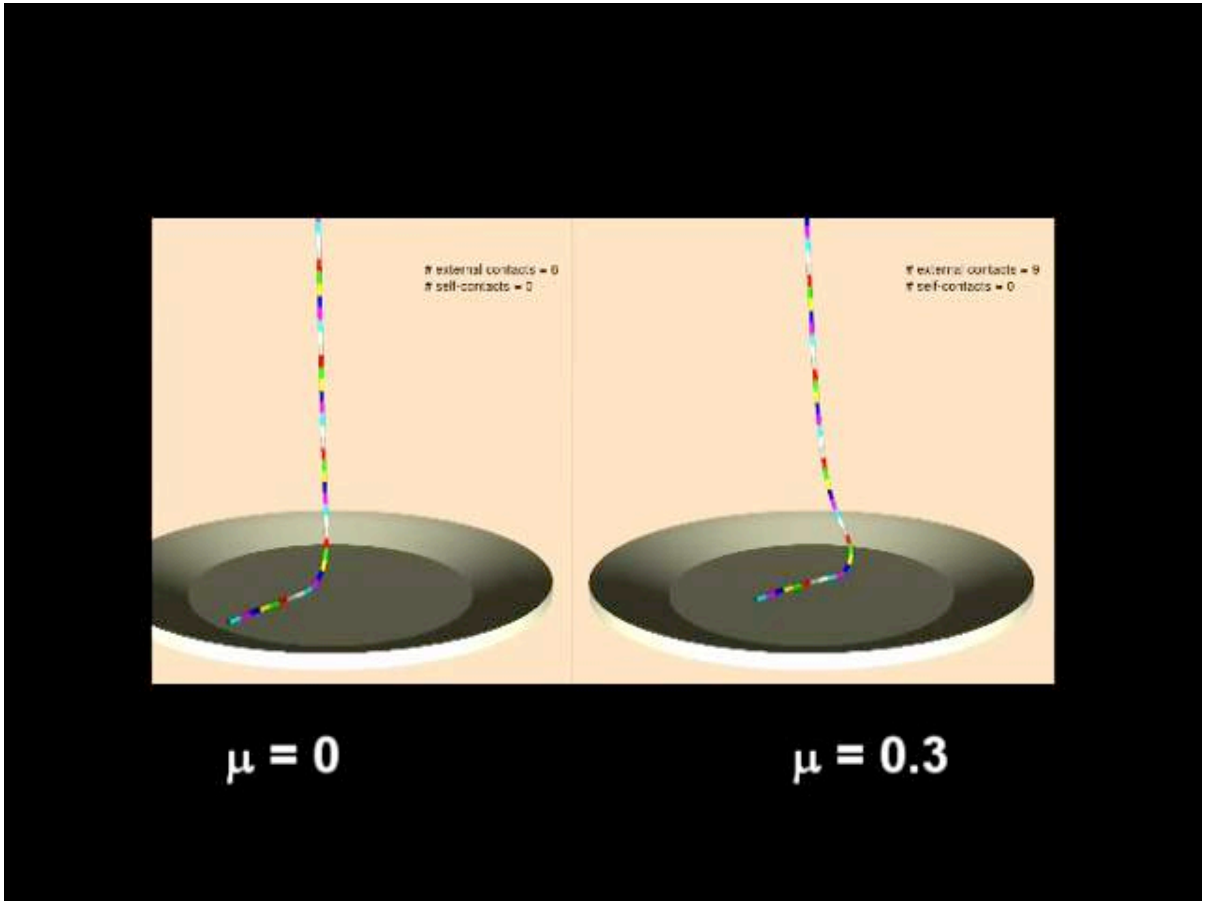
- Static Equilibrium: Find the minimum energy state of a deformable object
- Typically done using a Newton's method

Examples from Engineering

ENGINEERPEDIA

Examples in Graphics

- Newton's method is used to compute frictional force between hairs



Types of Optimization

- Continuous vs. Discrete
- Constrained vs. Unconstrained

Constrained Optimization

- Optimization involves finding an "optimal value"

- i.e. Maximizing a profit, minimizing an area etc…

$$\min f(x)$$
$$s.t\ \boxed{\mathbf{c}_i(\mathbf{x})} = 0$$

Equality Constraints

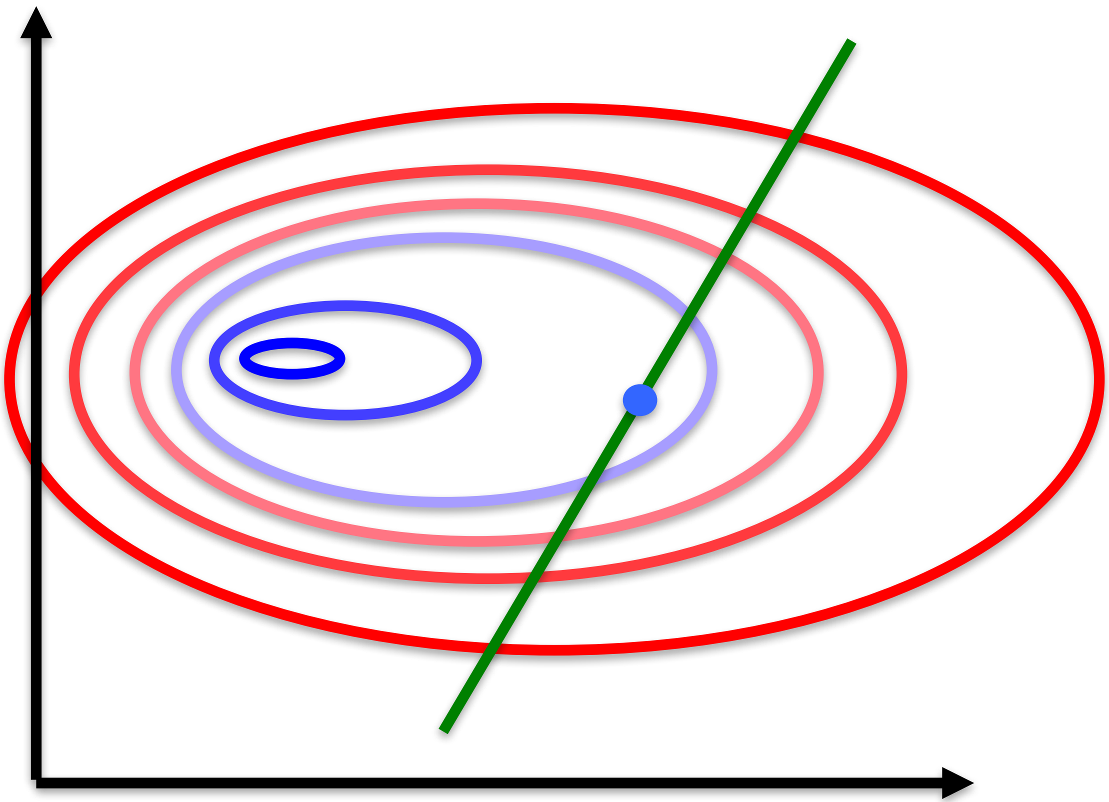
Constrained Optimization

- Optimization involves finding an "optimal value"
- i.e. Maximizing a profit, minimizing an area etc…

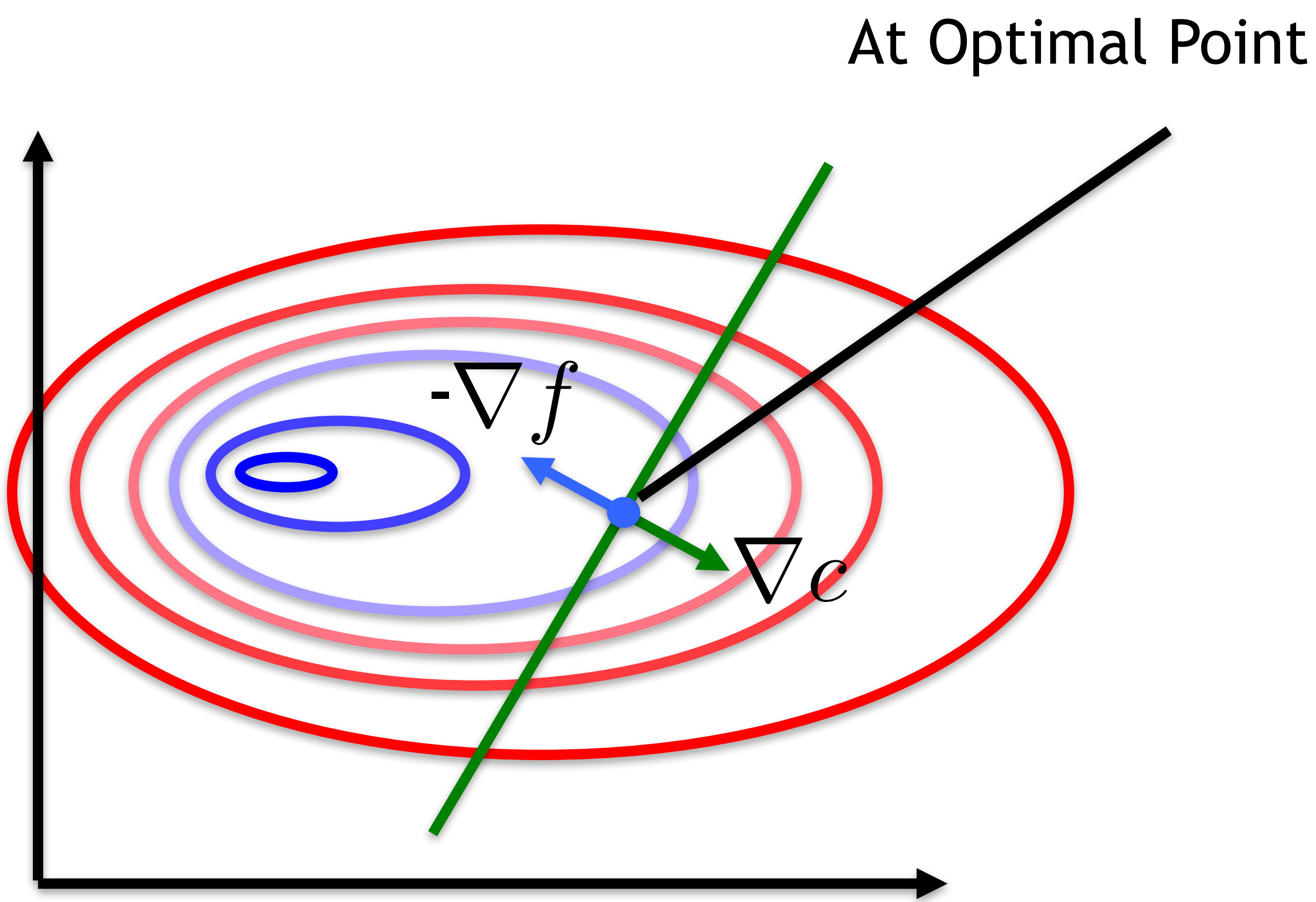
$$\min f\left(x\right)$$
$$s.t\ \boxed{Ax} = \mathbf{b}$$

Equality Constraints

Constrained Optimization



Constrained Optimization



Constrained Optimization

- Equation from Geometry

$$-\nabla f = \boxed{\lambda}\nabla c$$

Lagrange Multipliers!

Constrained Optimization

- Equation from Geometry

$$-\nabla f = \lambda \nabla c$$

$$\nabla f + \lambda \nabla c = 0$$

$$\nabla \left(f + \lambda c \right) = 0$$

$$\boxed{\min \left(f + \lambda c \right)}$$

Minimize old cost function + constraints · Lagrange Multipliers

Constrained Optimization

- Find Optimal Point!

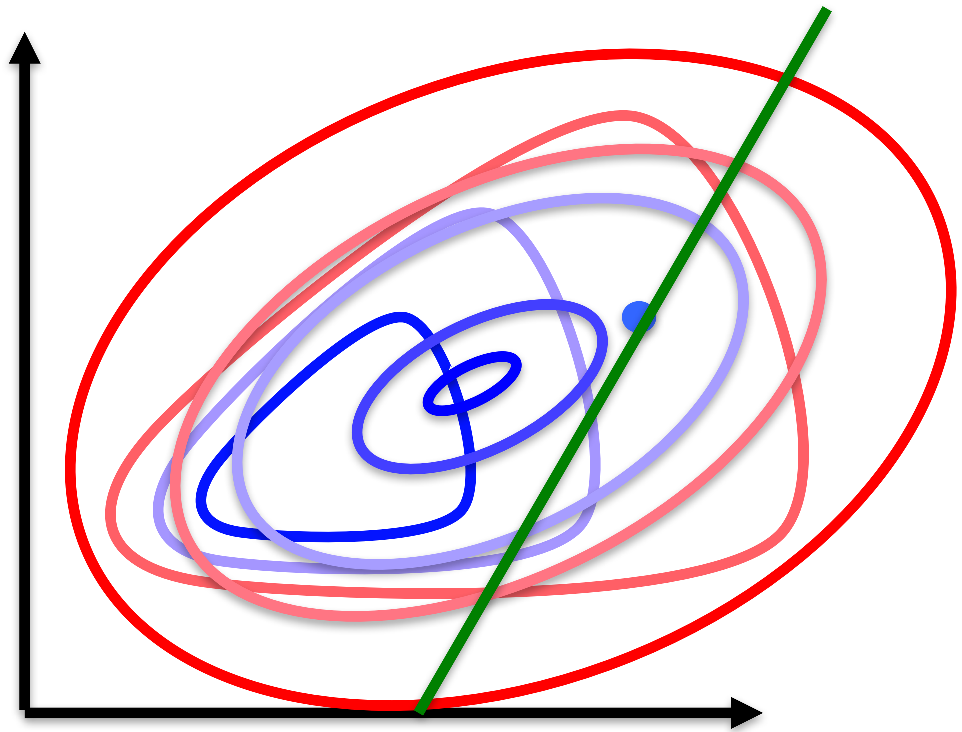
$$\nabla_{\mathbf{x}} f(\mathbf{x}) + \mathbf{A}^T \lambda = 0$$

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Constrained Optimization

- This simple tool is incredibly powerful
- Let's use it to build an equality constrained Newton's Method

Equality Constrained Newton's Method



Newton's Method

- How do we get our approximation ?
- Taylor Expansion!!!!

$$f\left(\mathbf{x}^c + \Delta\mathbf{x}\right) \approx f\left(\mathbf{x}^c\right) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

$$\boxed{\nabla f\big|_{\mathbf{x}^c}}$$

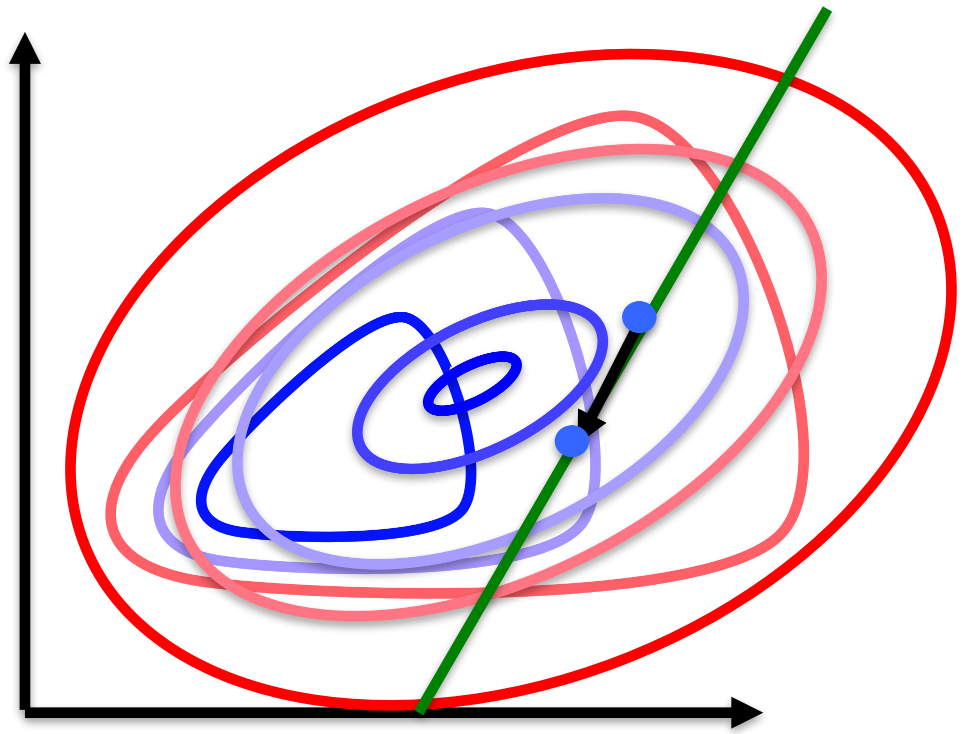
$$\boxed{H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}}$$

Equality Constrained Newton

- Add constraints to model problem

$$\Delta x = \arg\min f\left(\mathbf{x}^c\right) + \Delta\mathbf{x}^T\mathbf{g} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$
$$s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}$$

Equality Constrained Newton's Method



Equality Constrained Newton

- Very useful for general equality constrained problems
- Available in MATLAB as fmincon
- Easy to modify unconstrained Newton Code

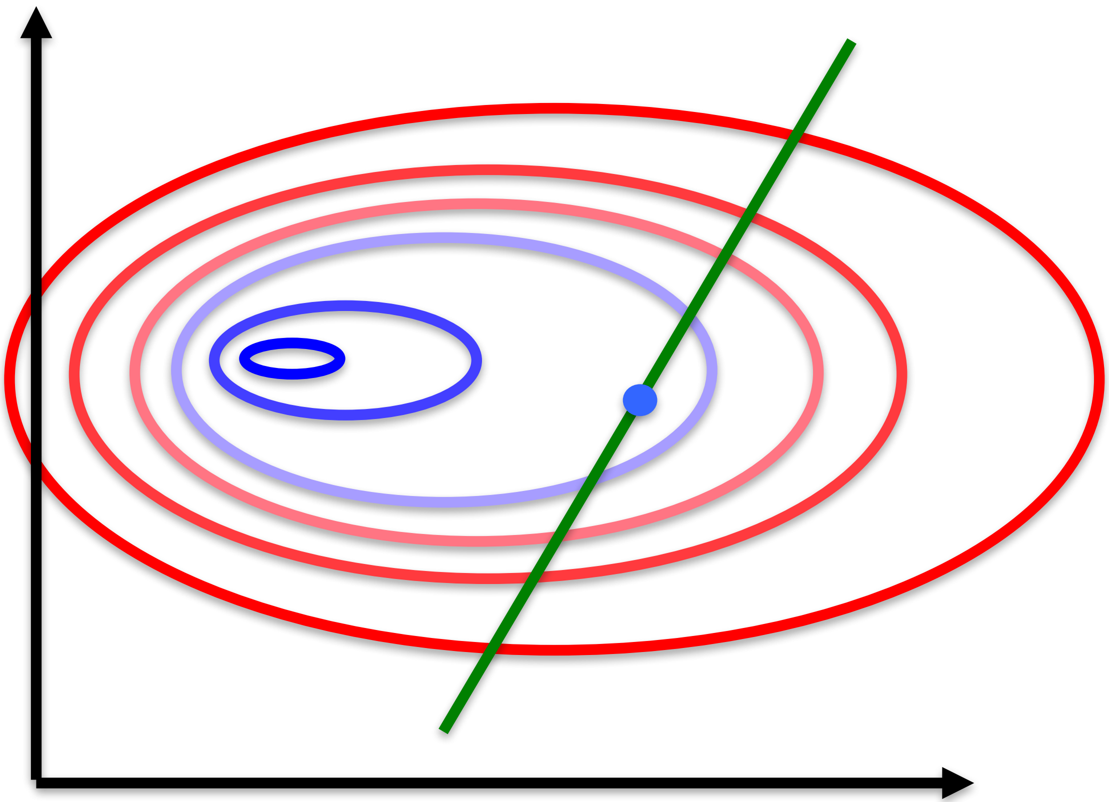
So Far !

- Gradient Descent
- Newton's Method
- Equality Constrained Newton's Method

Next: Inequality Constrained Optimization

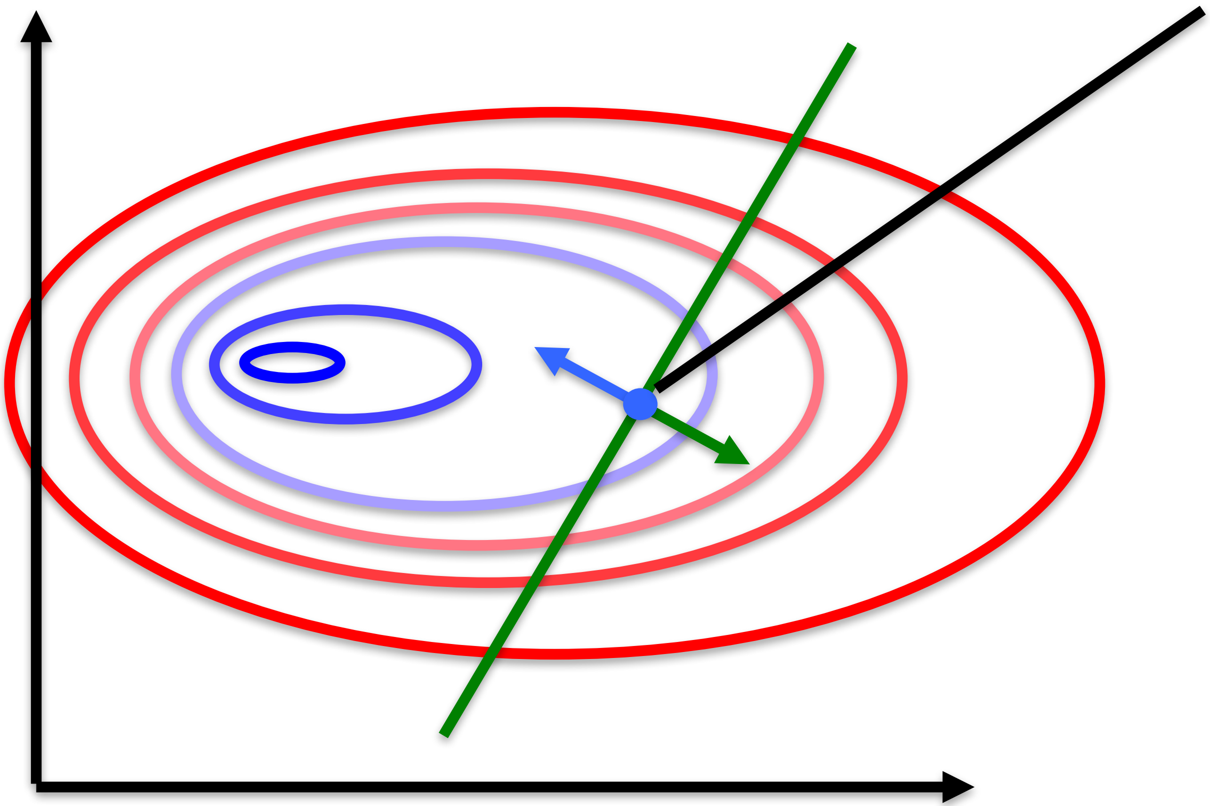
- Specifically we will work up to a particular type of problem called a Quadratic Program

Constrained Optimization



Constrained Optimization

At Optimal Point



Inequality Constrained Optimization

- Optimization involves finding an “optimal value”

- i.e. Maximizing a profit, minimizing an area etc…

$$\min f\left(x\right)$$

$$s.t\ \boxed{c_i\left(\mathbf{x}\right)} \leq 0$$

Inequality Constraints

Inequality Constrained Optimization

- Optimization involves finding an "optimal value"

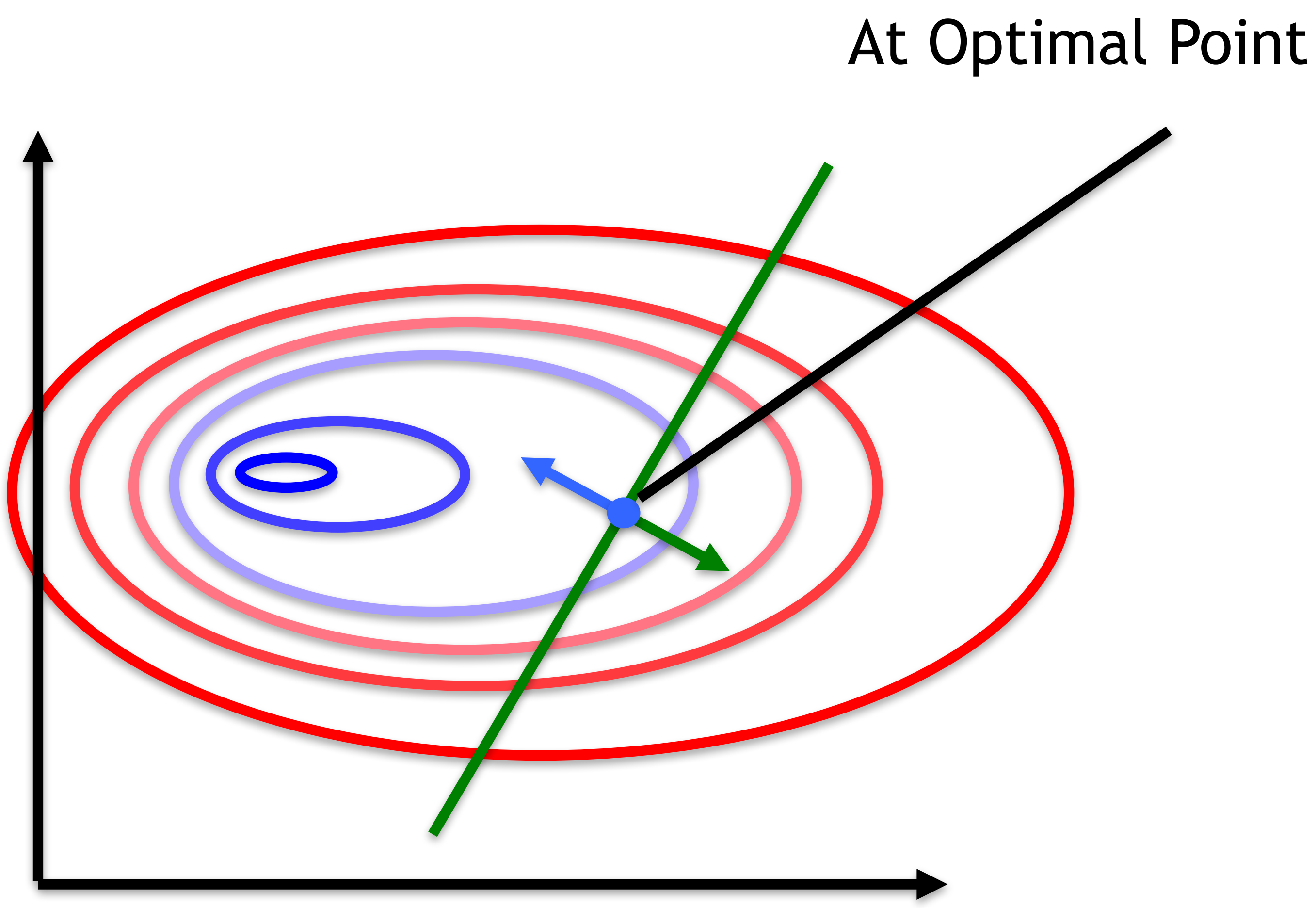
- i.e. Maximizing a profit, minimizing an area etc…

$$\min f\left(x\right)$$

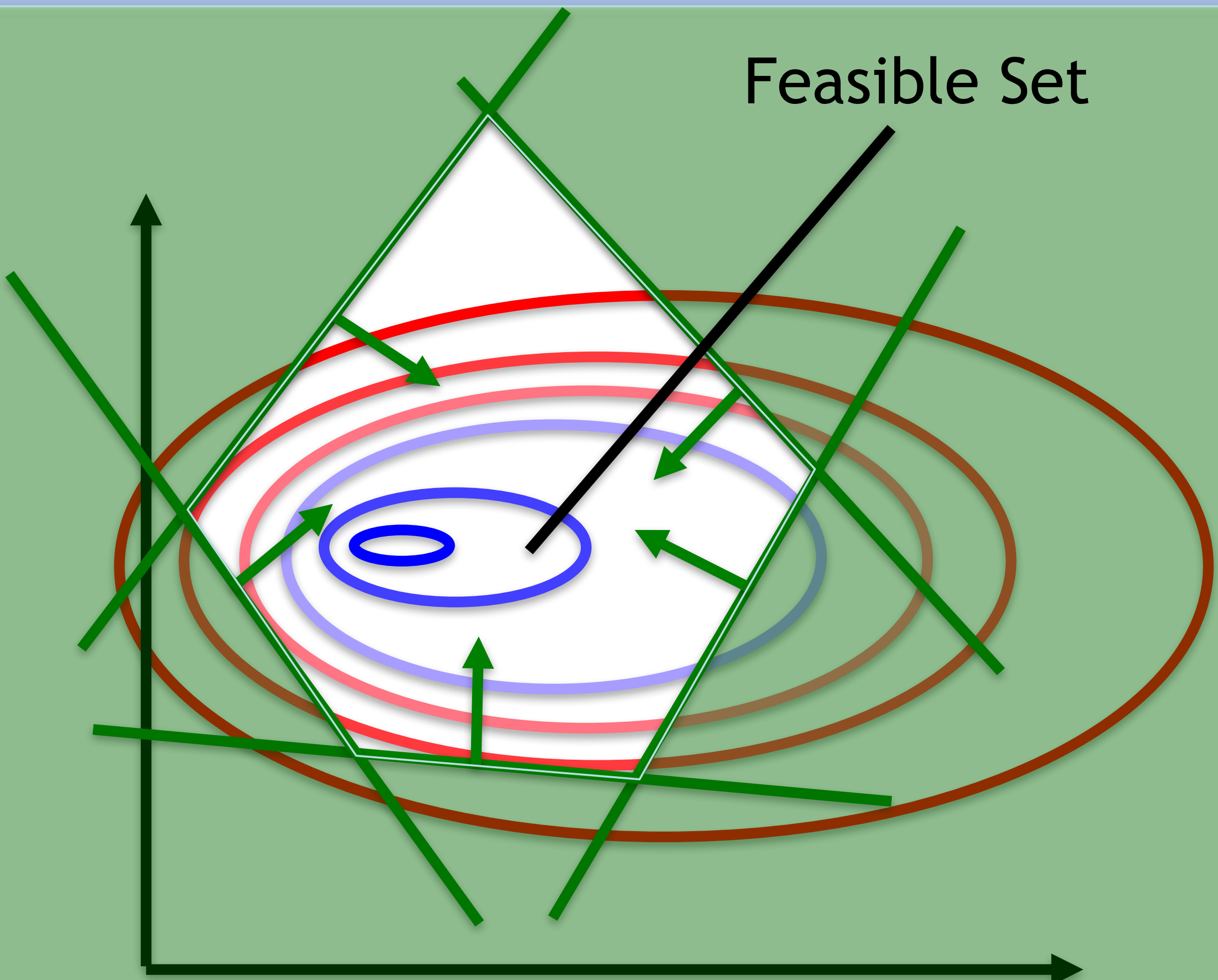
$$s.t\ \boxed{\mathbf{Ax} \leq}\ \mathbf{b}$$

Inequality Constraints

Equality Constrained Optimization



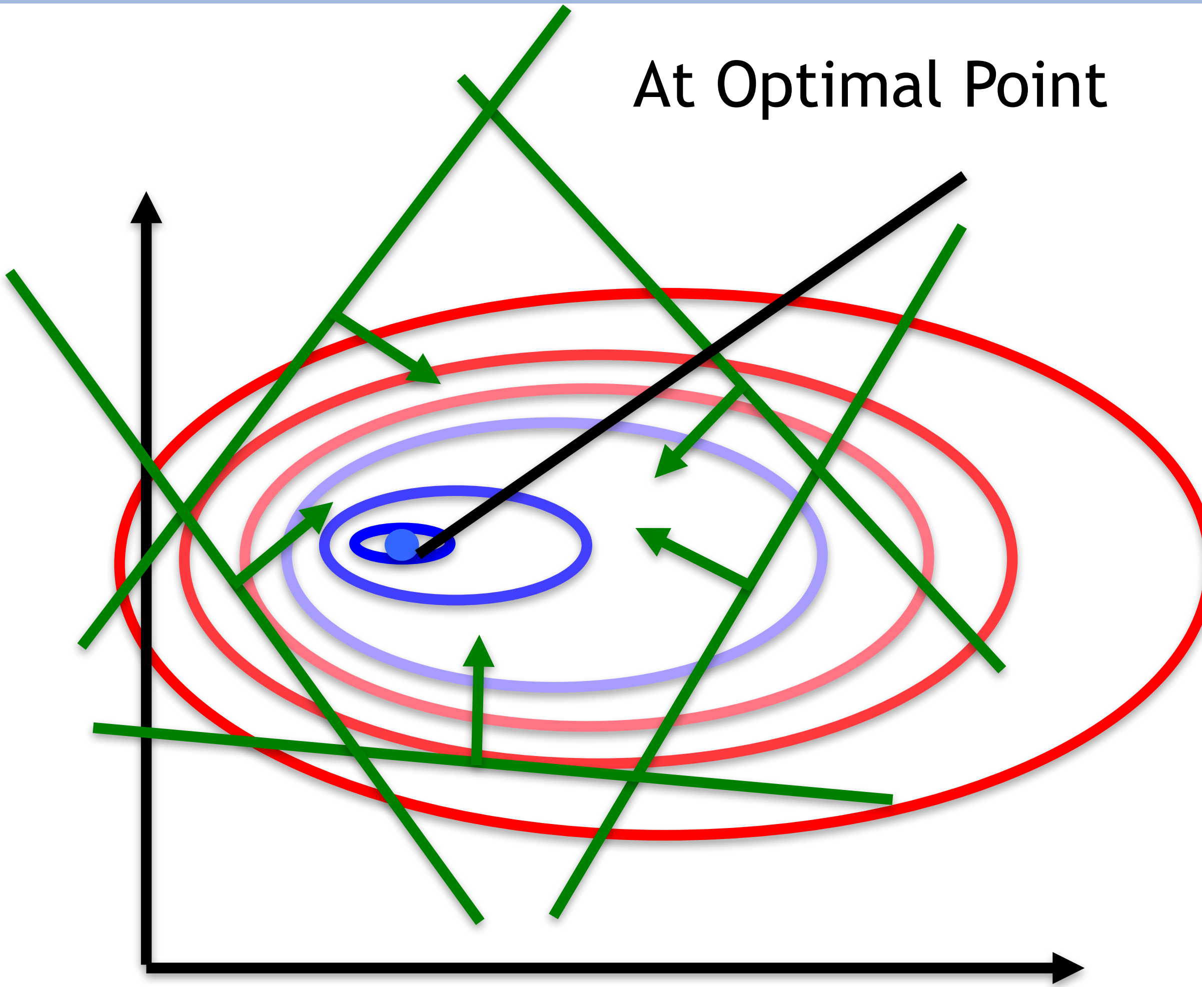
Inequality Constrained Optimization



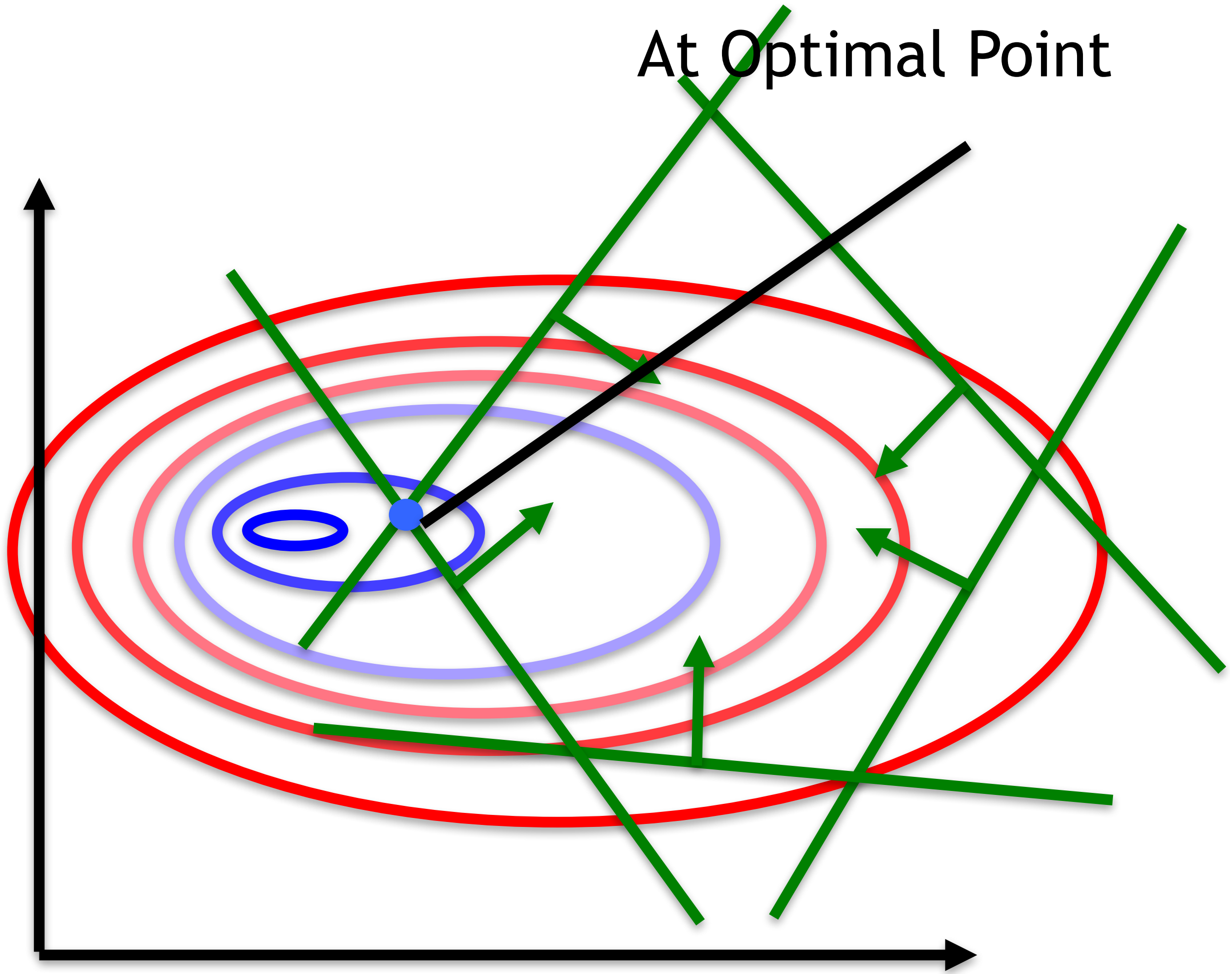
The Active Set

- Hidden inside of each inequality constrained optimization is an equality constrained optimization
- There are two cases for our optimal point…

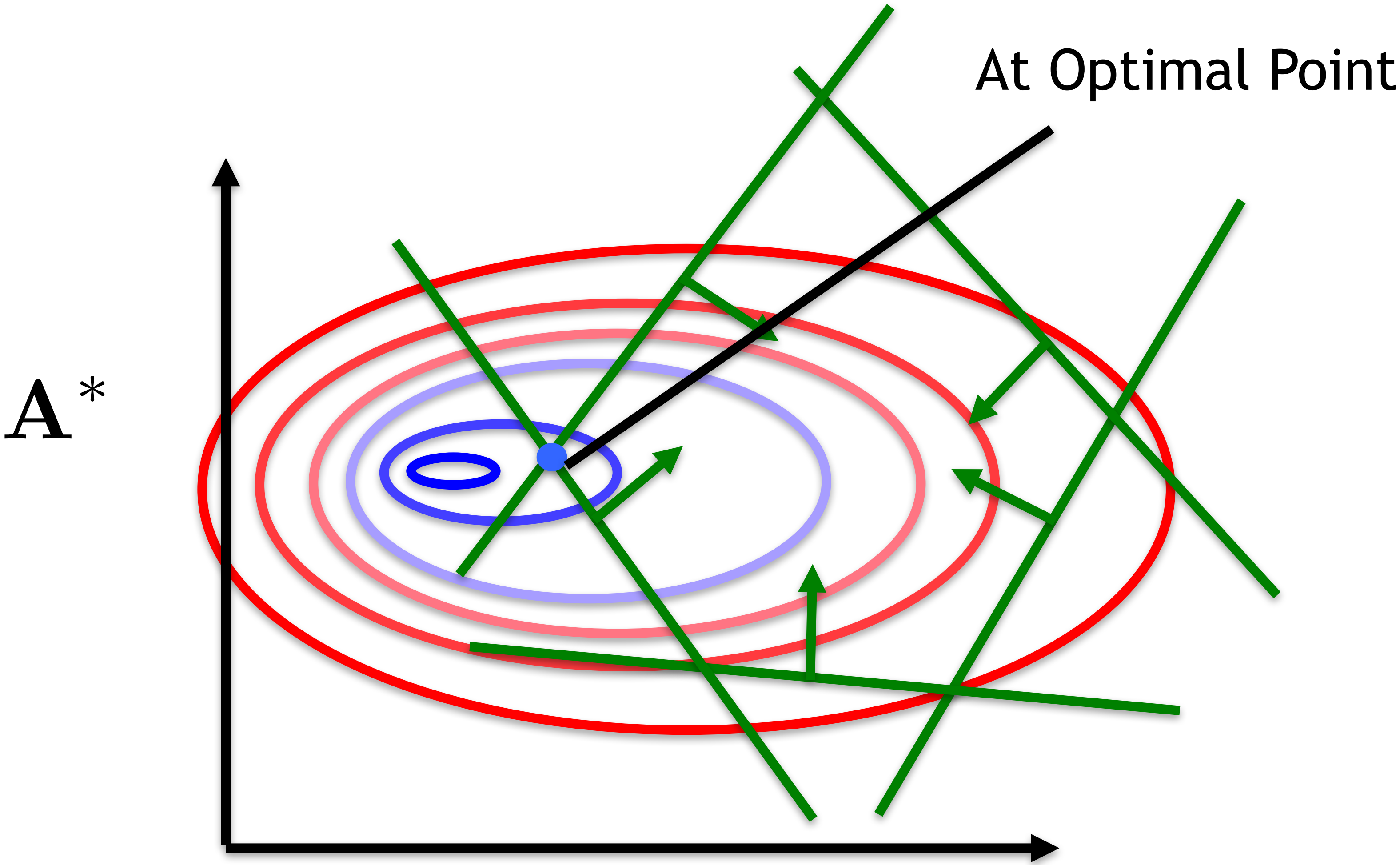
Case 1: Optimal Value Inside Feasible Set



Case 2: Optimal Value On Boundary



Case 2: Optimal Value On Boundary



The Active Set

- On the boundary we satisfy

$$\min f(x)$$

$$s.t\ \boxed{\mathbf{A}^*}\mathbf{x} = \mathbf{b}$$

Active Set

Quadratic Programs

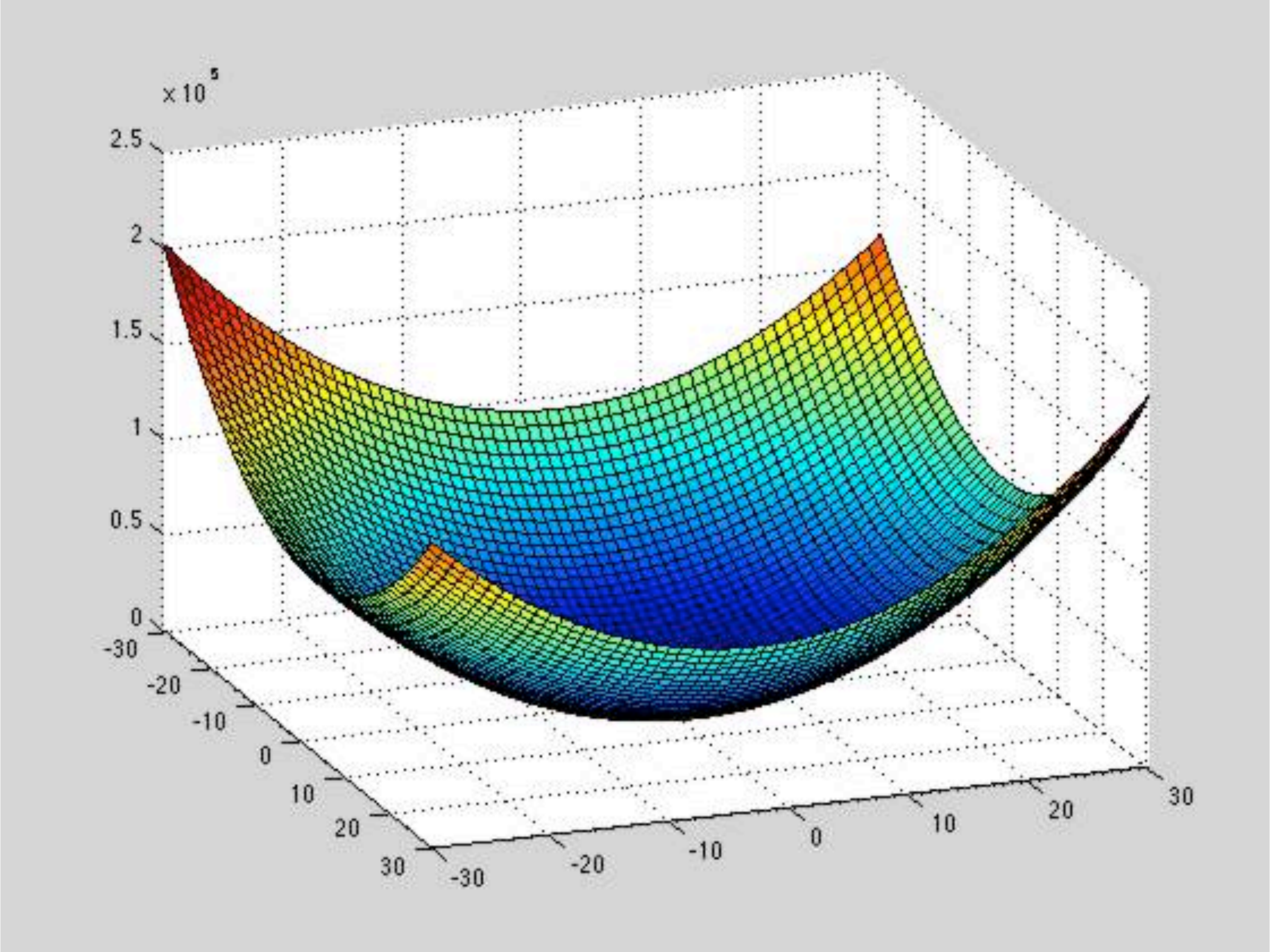
- It's got a quadratic cost function!

$$\min \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{x}^T \mathbf{d}$$

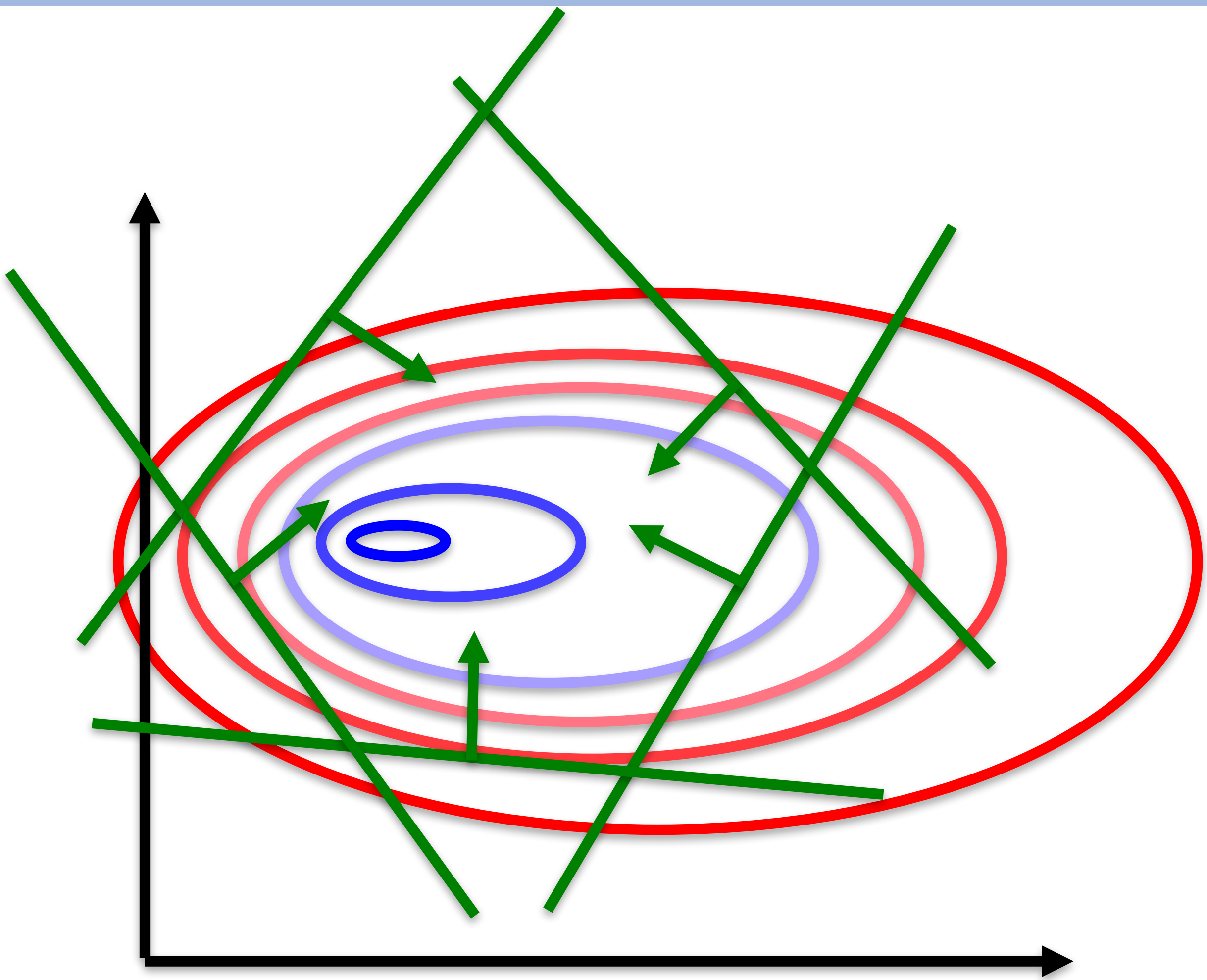
$$s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}$$

$$s.t.\ \mathbf{L}\mathbf{x} \leq \mathbf{m}$$

Quadratic Program



Quadratic Programs



Quadratic Program

- How do we solve this ?
- Active Set: Try different combinations of constraints until the minimum is found
- Interior Point: …

Interior Point Methods

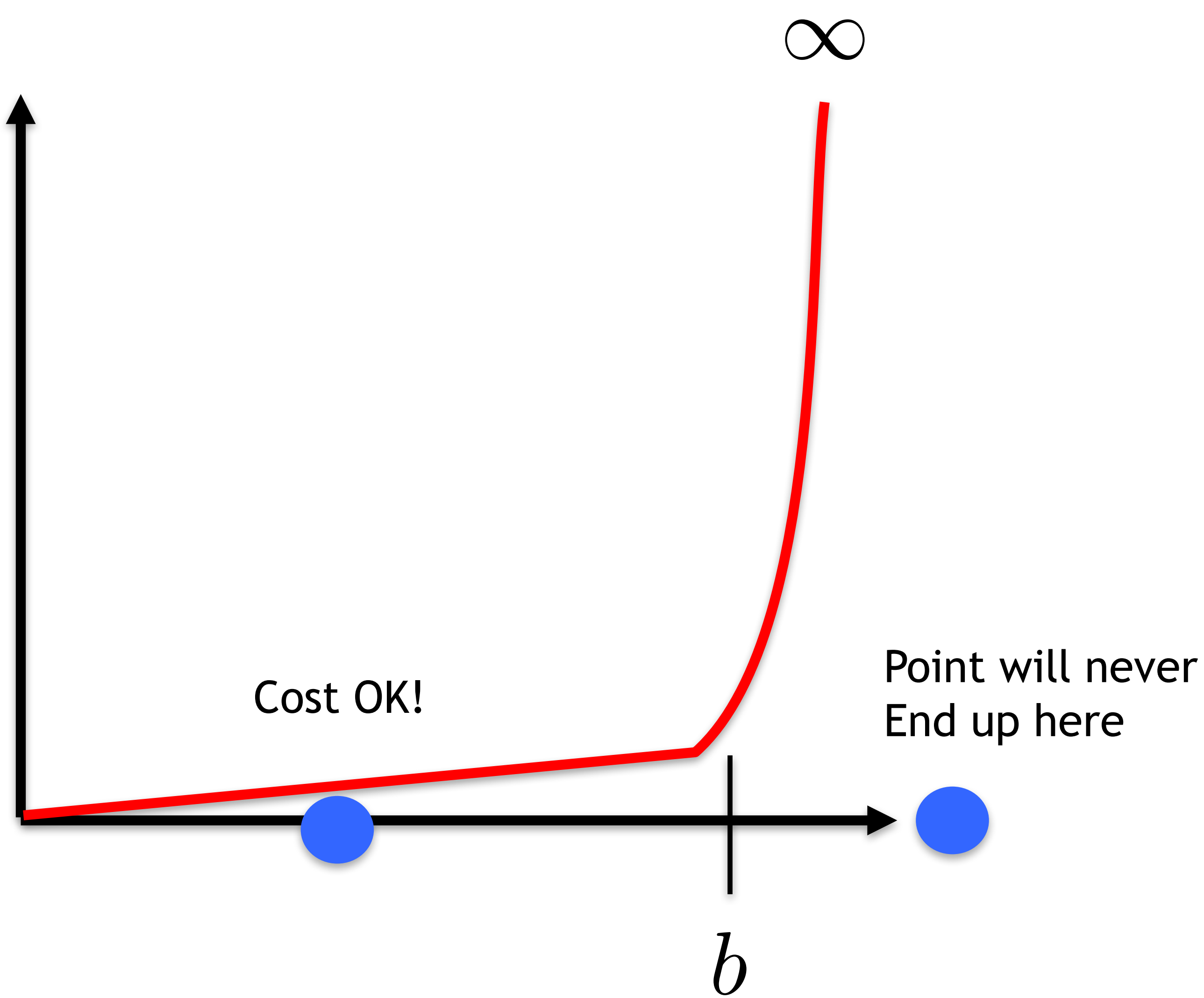
- Replace inequality constraints with functions

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + (\mathbf{A}\mathbf{x} - \mathbf{b})^T \lambda$$

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + (\mathbf{A}\mathbf{x} - \mathbf{b})^T \lambda + \sum_i \boxed{c_i}(\mathbf{x})$$

Special "Constraint" Function

Interior Point



Interior Point Methods

- Replace inequality constraints with functions

$$L\left(\mathbf{x},\lambda\right)=f\left(\mathbf{x}\right)+\left(\mathbf{Ax}-\mathbf{b}\right)^{T}\lambda$$

$$L\left(\mathbf{x},\lambda\right)=f\left(\mathbf{x}\right)+\left(\mathbf{Ax}-\mathbf{b}\right)+\sum_{i}\boxed{c_i}\left(\mathbf{x}\right)$$

Special “Constraint” Function

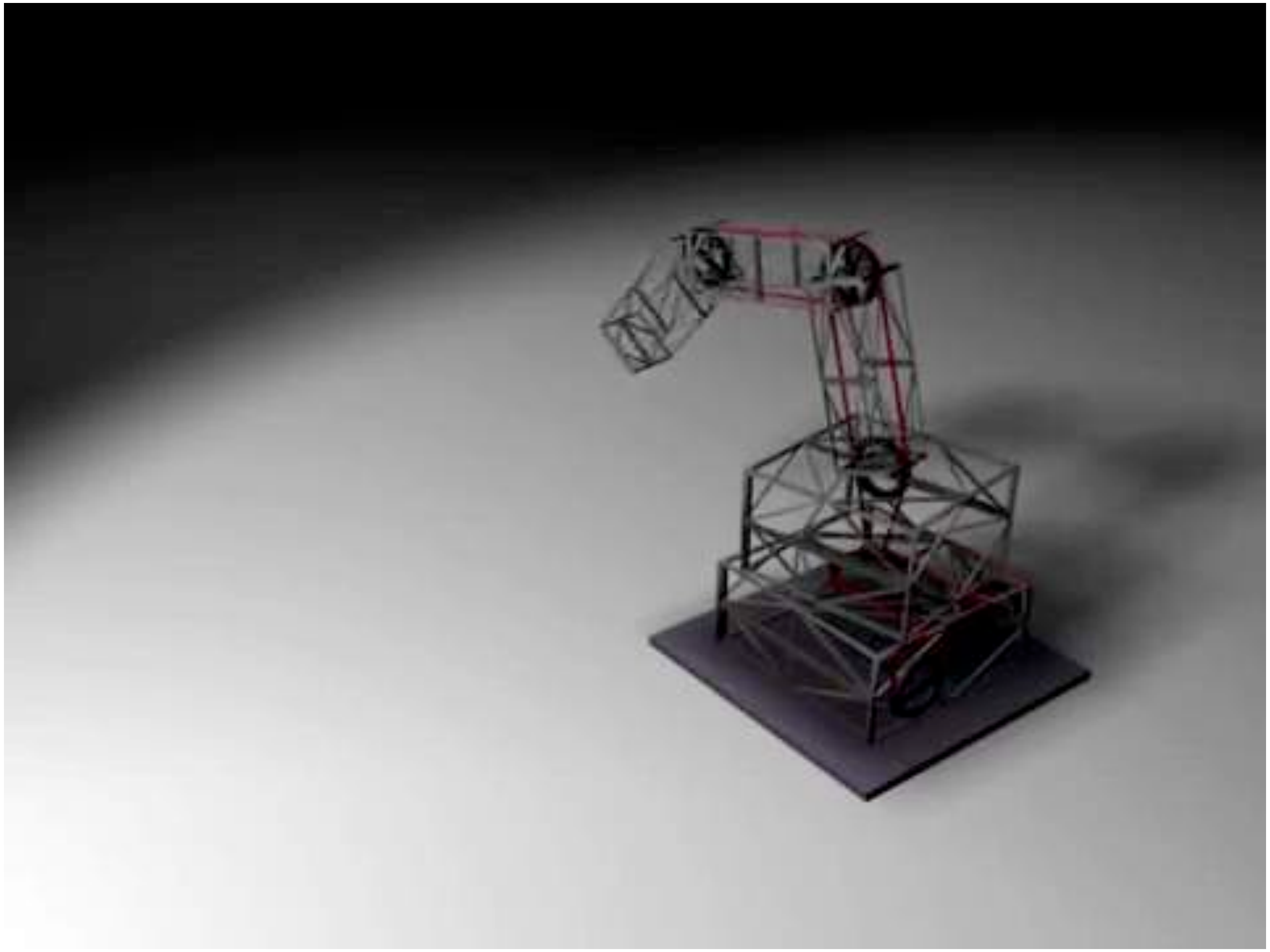
- Now use Equality Constrained Newton!!!

Quadratic Programs and Interior Point

- Quadratic Programs (Active Set)
  - Quadprog++ (http://quadprog.sourceforge.net)
  - MATLAB: quadprog
- Interior Point
  - Ipopt (https://projects.coin-or.org/Ipopt)

Examples of Quadratic Programming

Examples of Quadratic Programming

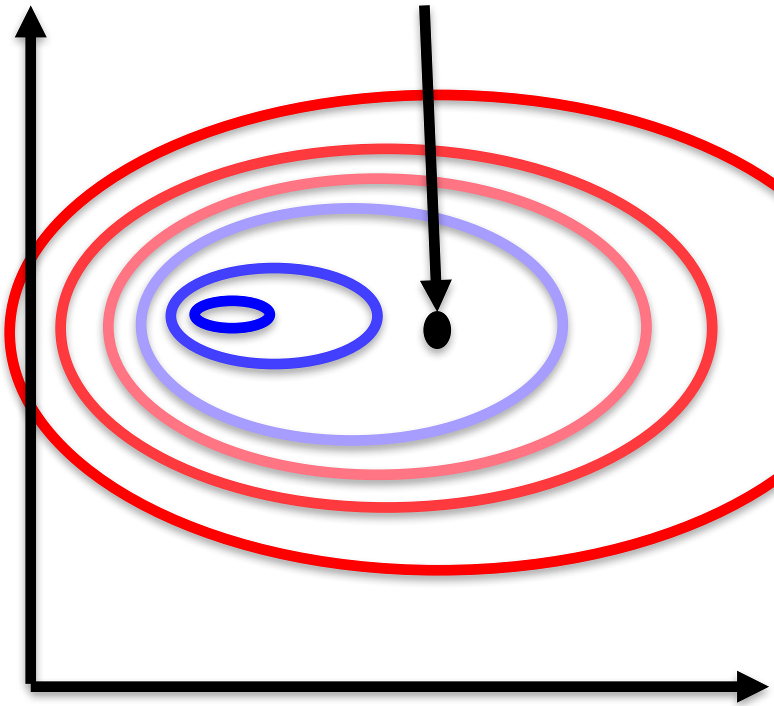


Types of Optimization

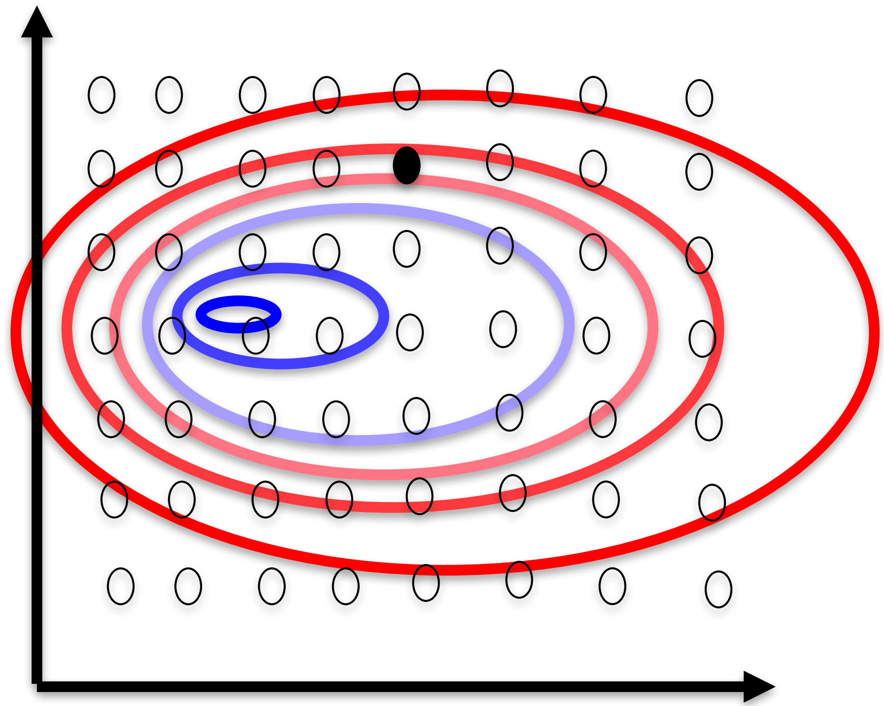
- Continuous vs. Discrete
- Constrained vs. Unconstrained

Continuous | Discrete

This point can move smoothly



Choose from discrete points in parameter space



Branch and Bound Optimizations

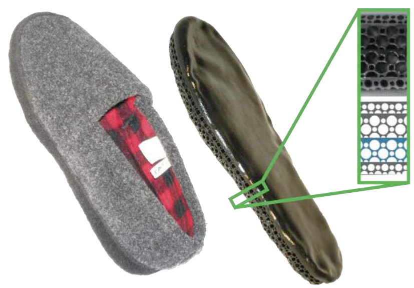
- An optimization technique with 3 phases
  - Branch (divide the solution space into a number of subspaces)
  - Bound (compute some upper and lower bound for the cost of each subspace)
  - Prune (remove subspaces with upper bounds higher than the lower bounds of the least costly subspaces)

Continuum Mechanics and Fabrication

- Example: Cloning Object Behavior

We want to control the printed shoes response to applied force

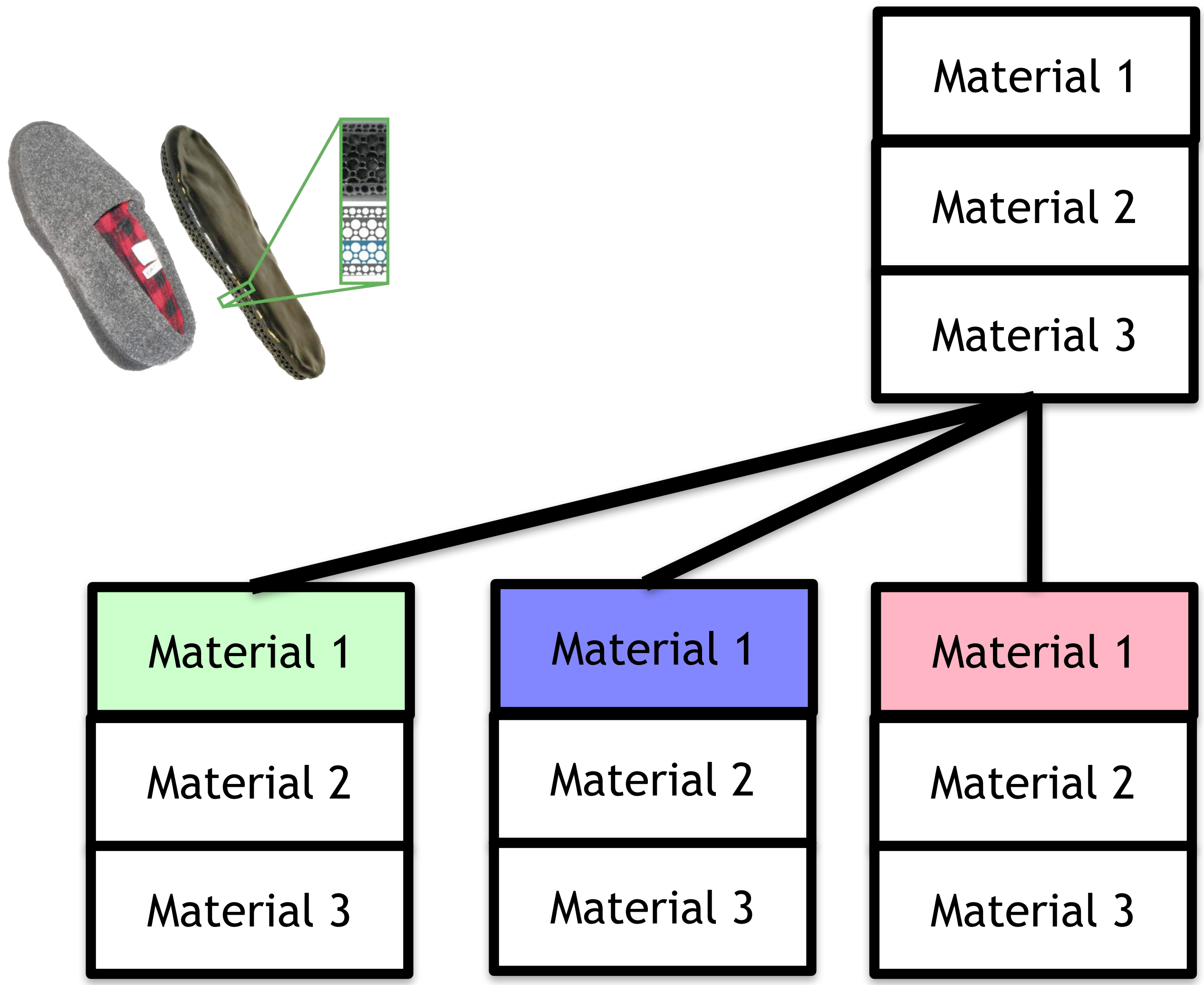
Material Assignment



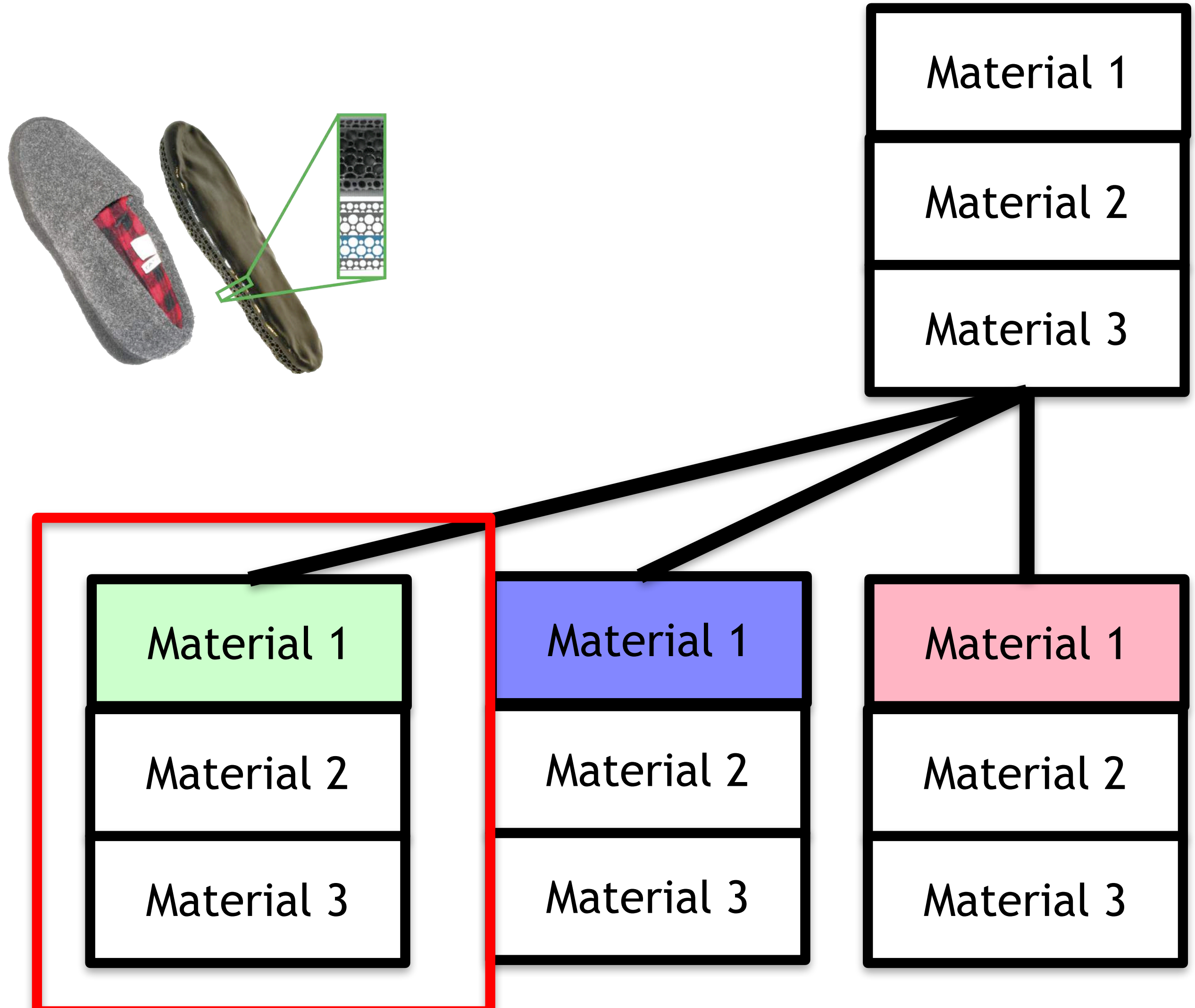
Match Goal Displacement G

Material 1	Material 1	Material 1	Material 1
Material 2	Material 2	Material 2	Material 2
Material 3	Material 3	Material 3	Material 3

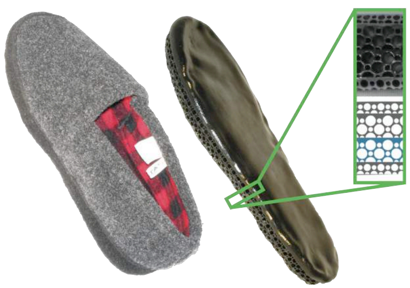
Material Assignment



Bounding Material Assignment



Bounding Material Assignment



Material 1
Material 2
Material 3

Material 1
Material 2
Material 3

SIMULATION

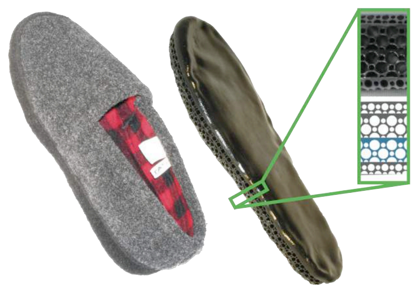
Material 1
Material 2
Material 3

SIMULATION

Material 1
Material 2
Material 3

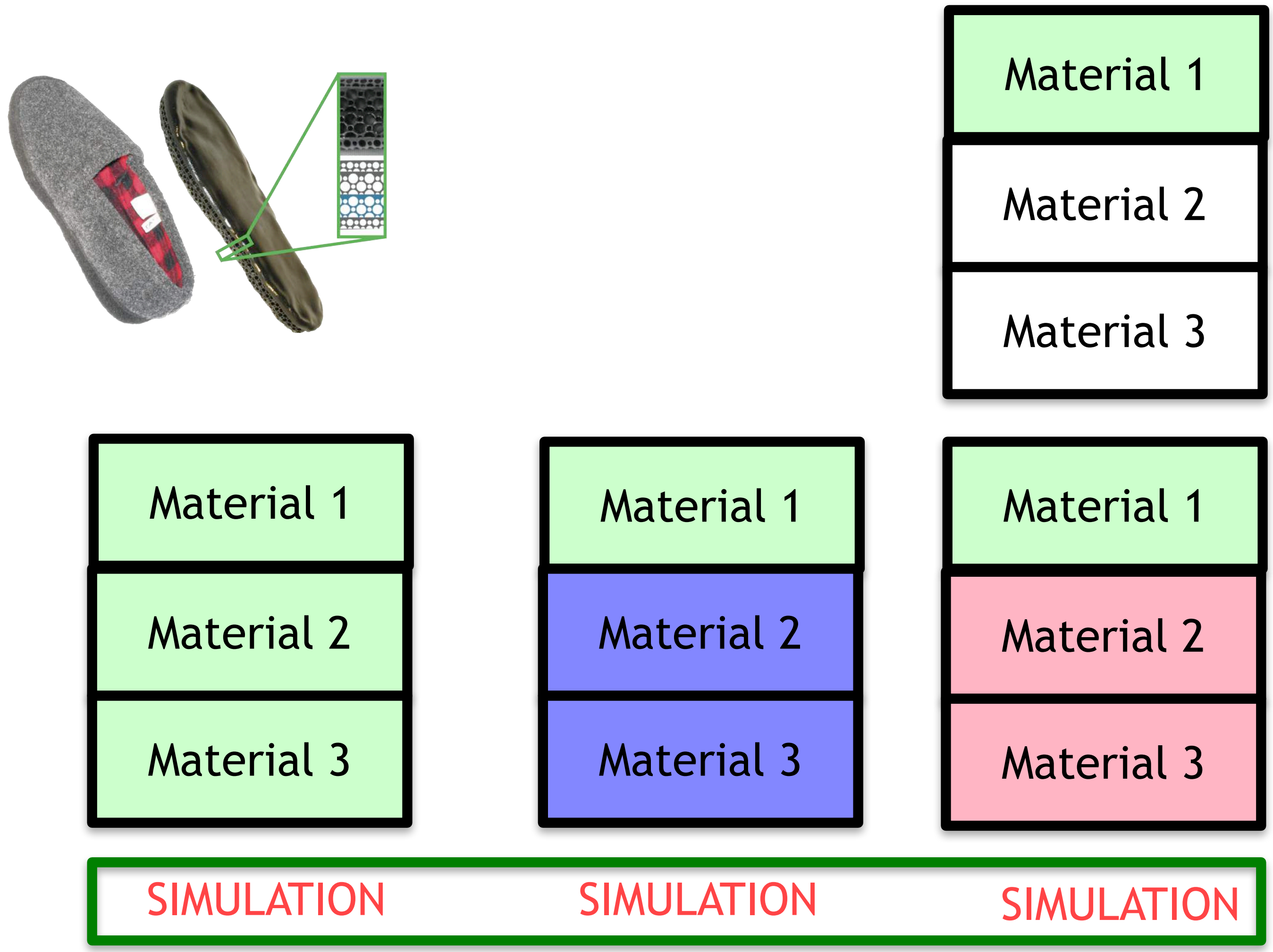
SIMULATION

Bounding Material Assignment

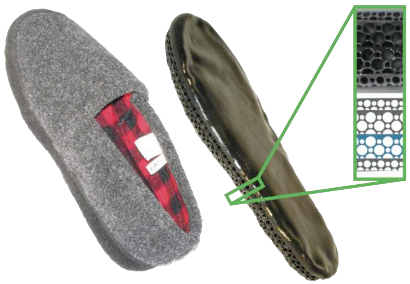


		Material 1
		Material 2
		Material 3
Material 1	Material 1	Material 1
Material 2	Material 2	Material 2
Material 3	Material 3	Material 3
SIMULATION	SIMULATION	SIMULATION

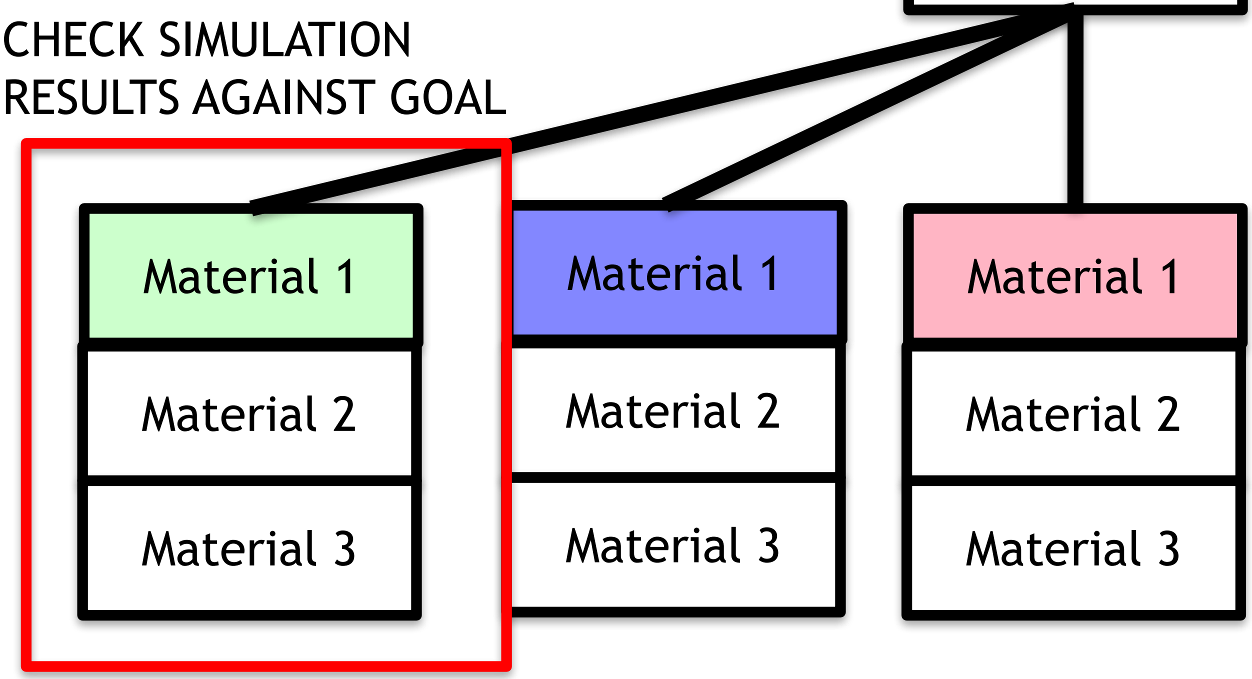
Bounding Material Assignment



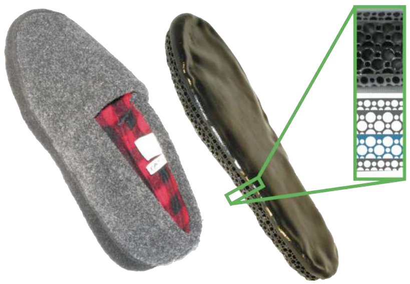
Bounding Material Assignment



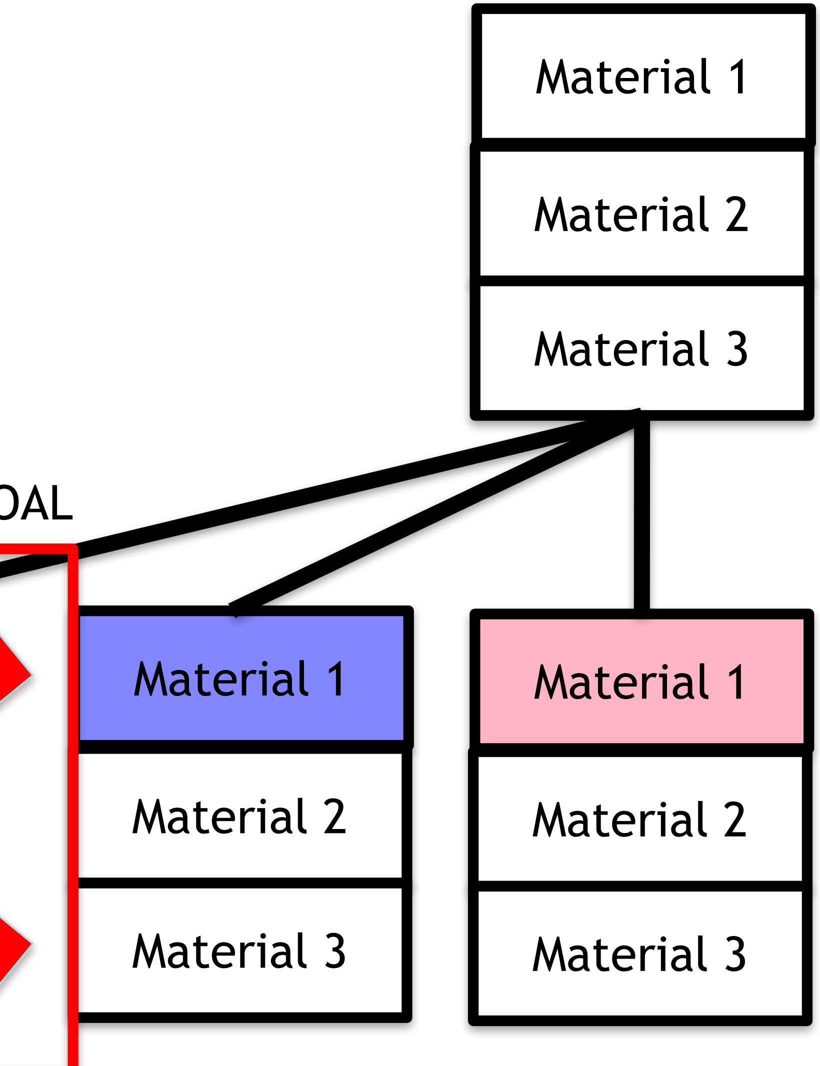
CHECK SIMULATION
RESULTS AGAINST GOAL



Bounding Material Assignment



CHECK SIMULATION
RESULTS AGAINST GOAL



Simulated Annealing

- Has four ingredients
  - Cost function
  - Configuration (made of discrete elements)
  - Neighbor Generator
  - Annealing Schedule

Simulated Annealing

- Basic Idea taken from cooling of materials in metallurgy
- At high “heat” atoms undergo rigorous motion
- As they are cooled they move less

Simulated Annealing

- Cost function: $f(\boxed{\mathbf{q}})$

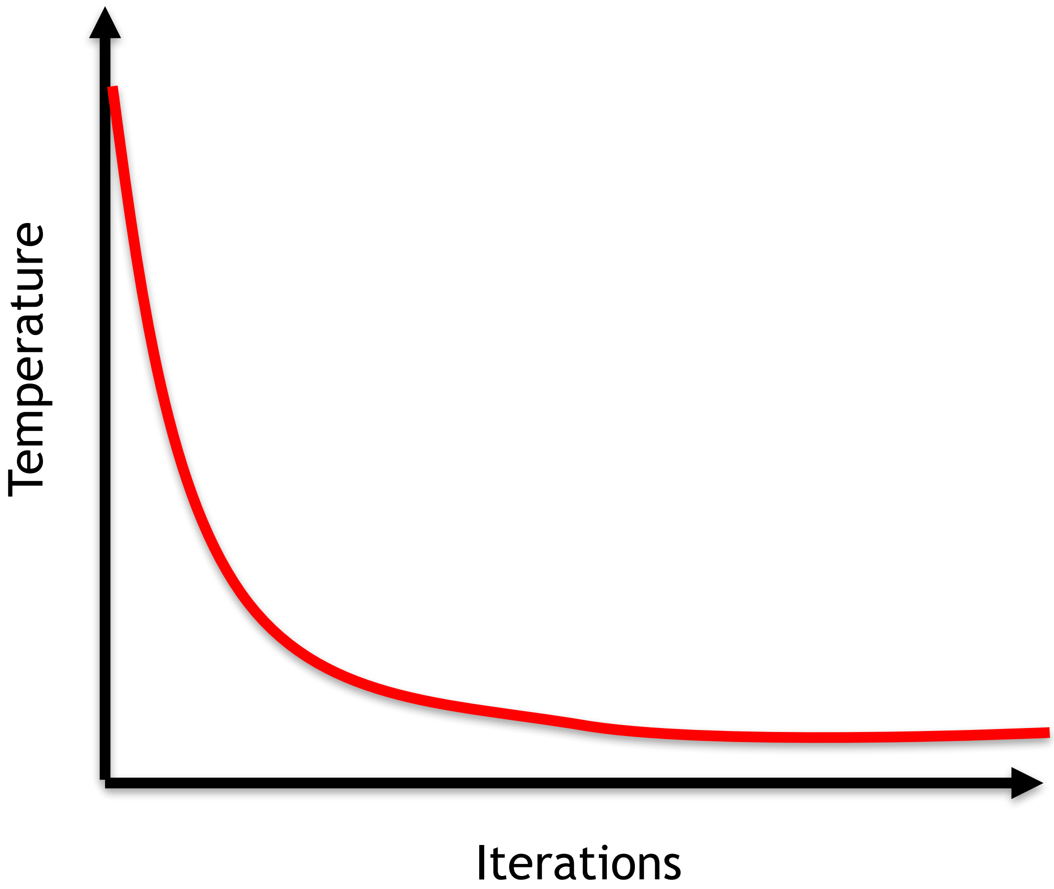
Configuration

Simulated Annealing

- Cost function: $f\left(\mathbf{q}\right)$
- Configuration: $\mathbf{q}$ e.g. Material Assignments
- Neighbor Generator: Rearrange Configuration
  - e.g. Change some materials to ones with nearby stiffness
- Annealing Schedule

Simulated Annealing

- Annealing Schedule



Simulated Annealing

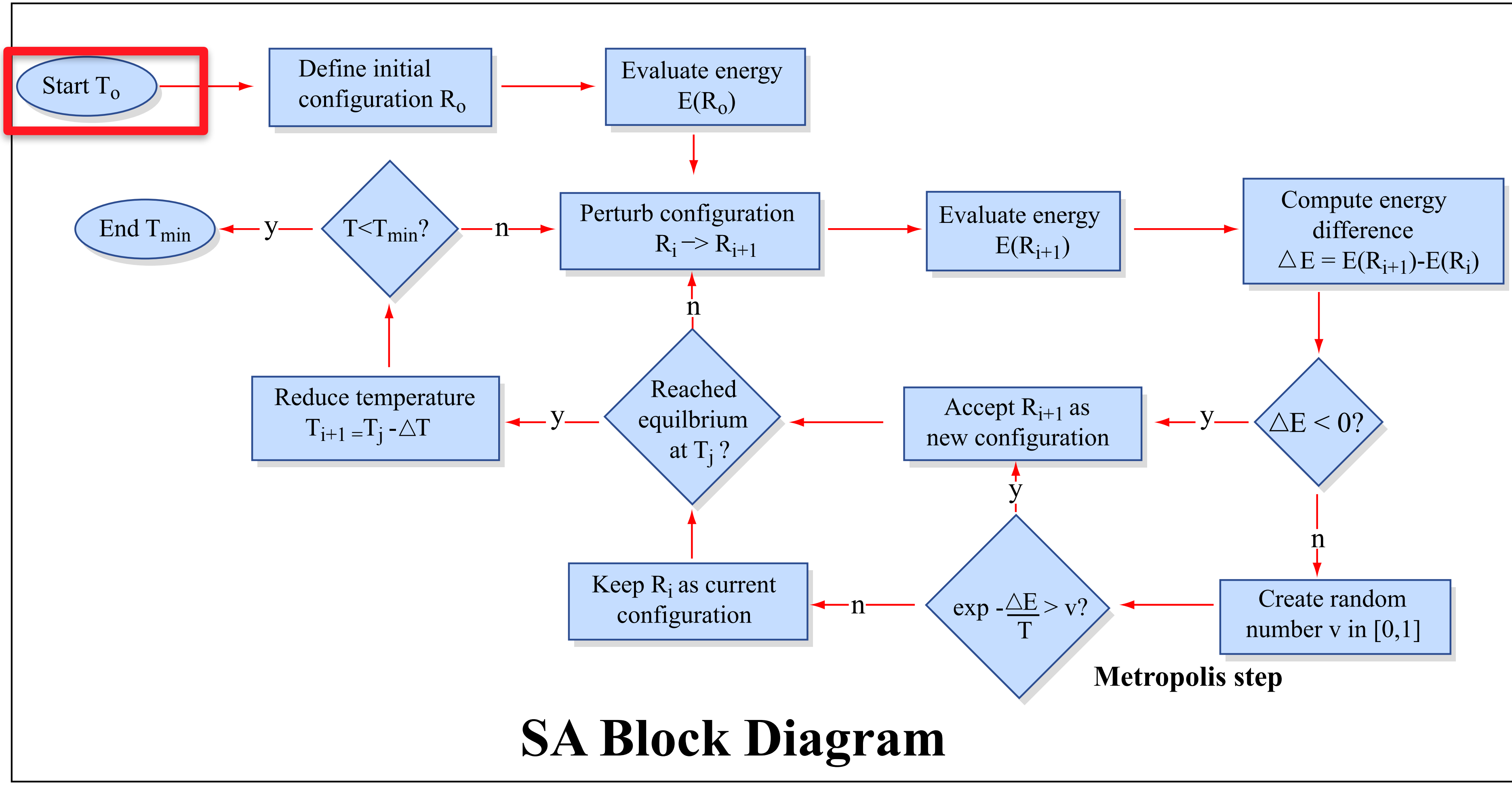
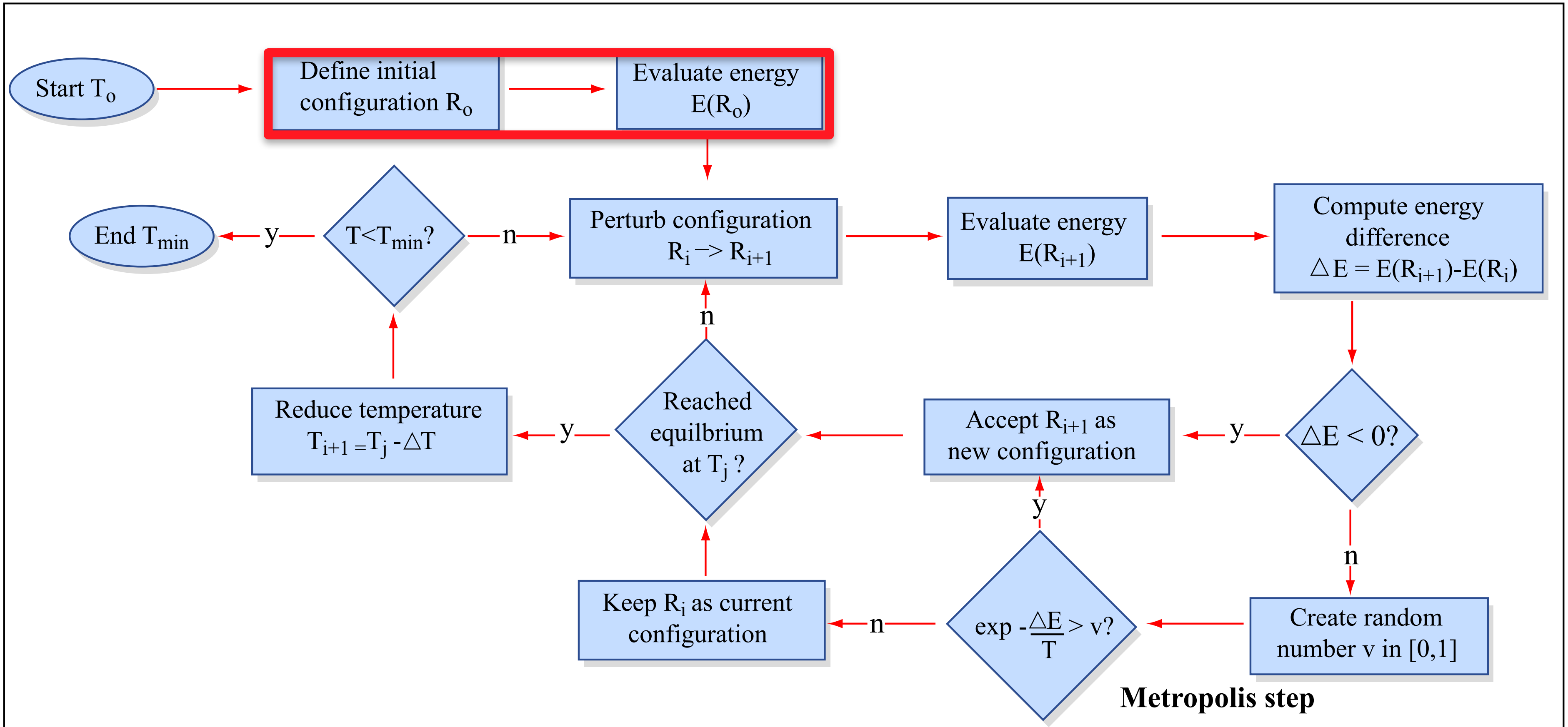


Image by MIT OpenCourseWare.

Simulated Annealing



SA Block Diagram

Image by MIT OpenCourseWare.

Simulated Annealing

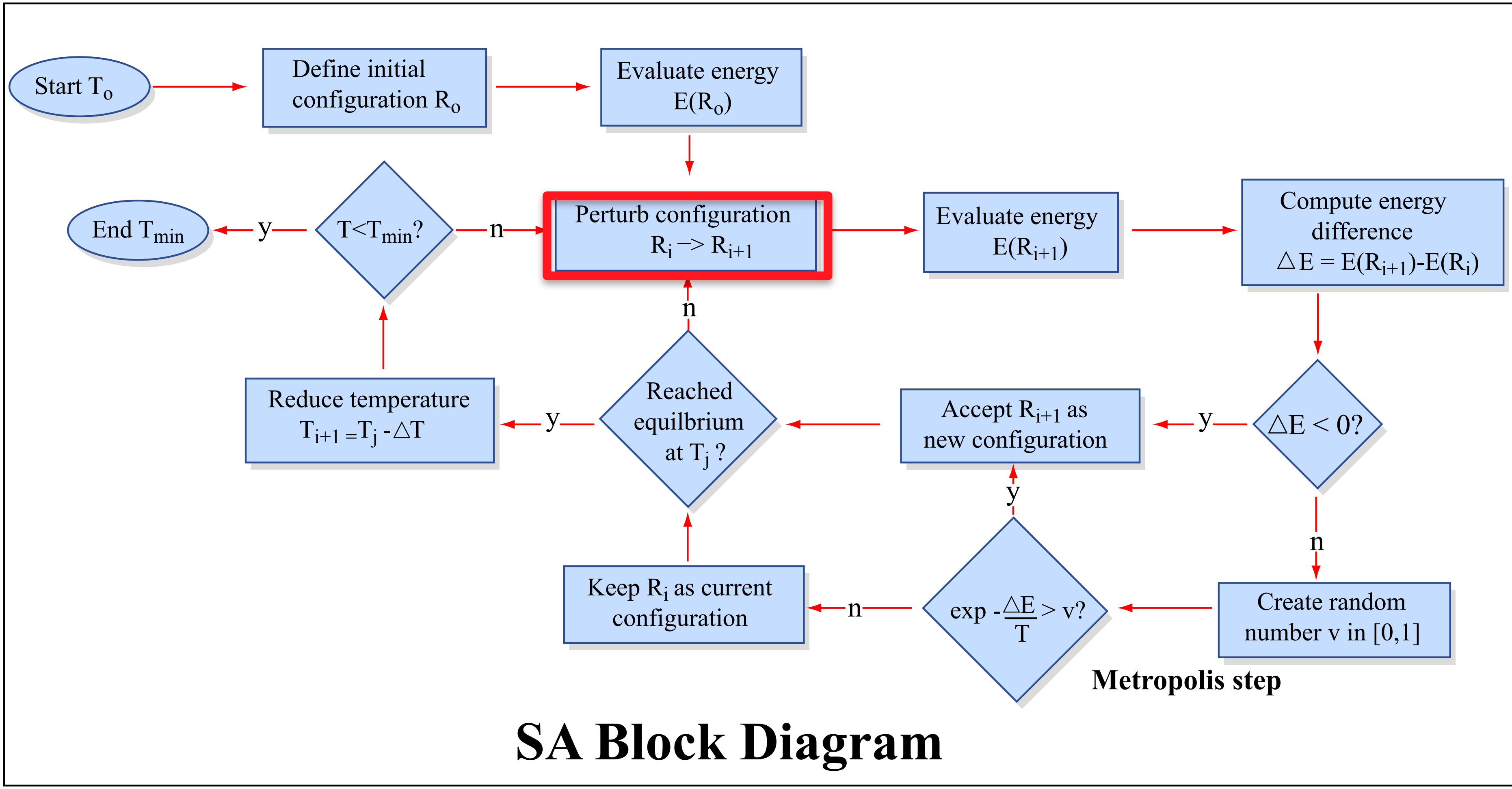
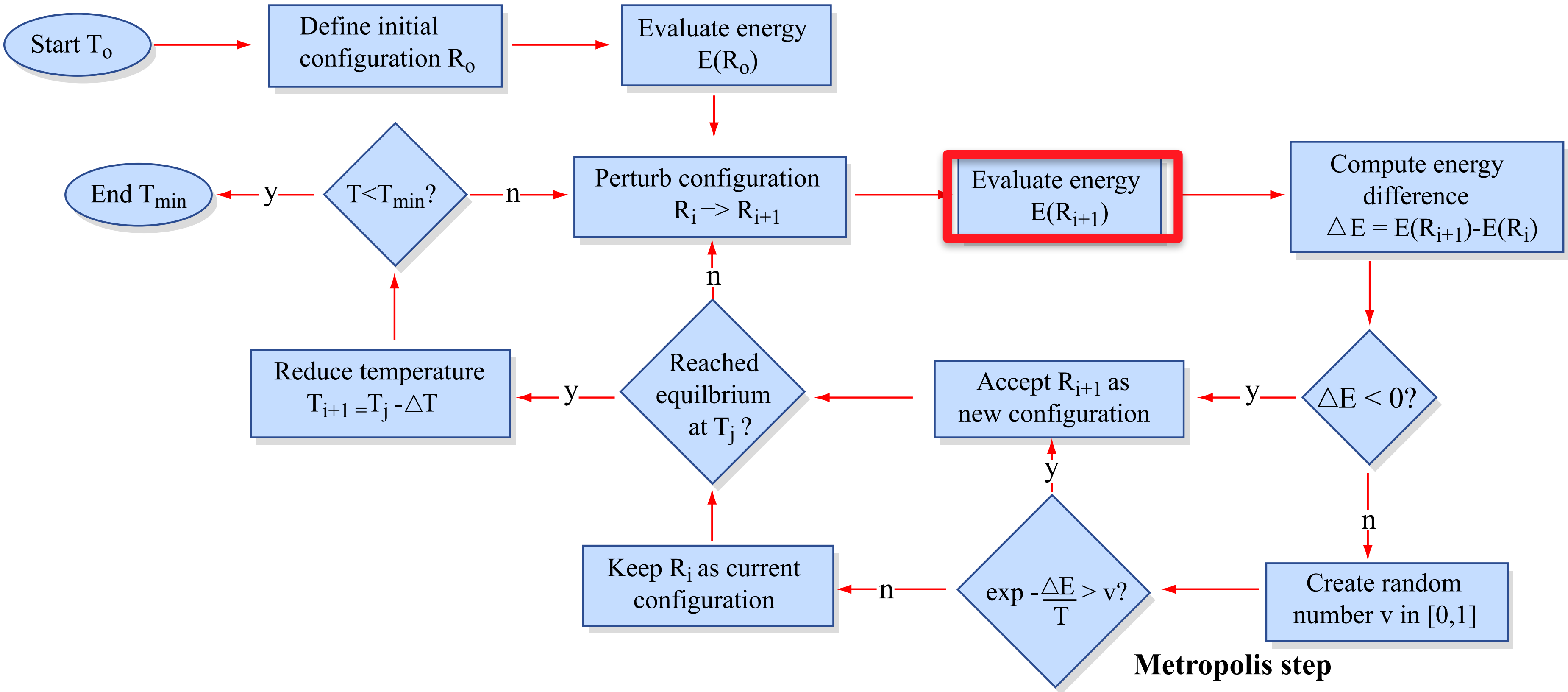


Image by MIT OpenCourseWare.

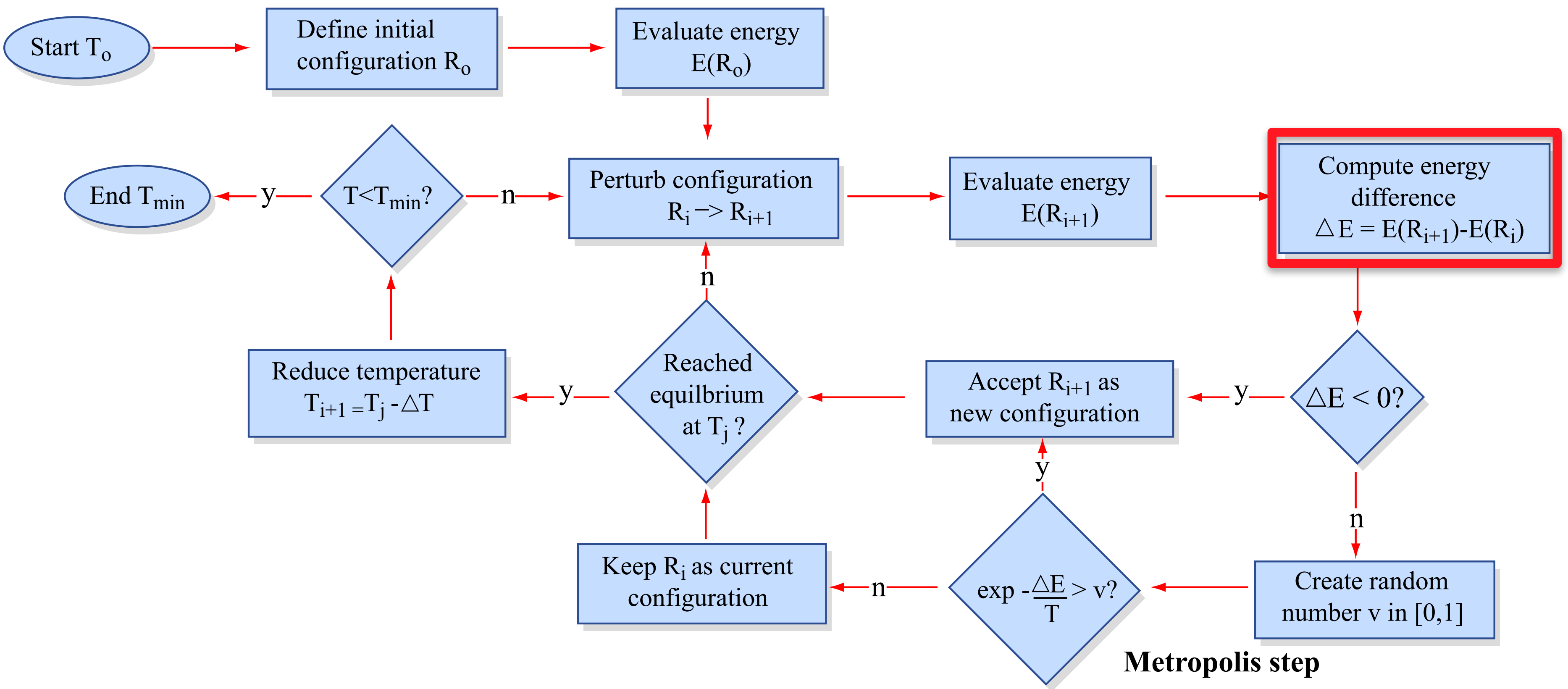
Simulated Annealing



SA Block Diagram

Image by MIT OpenCourseWare.

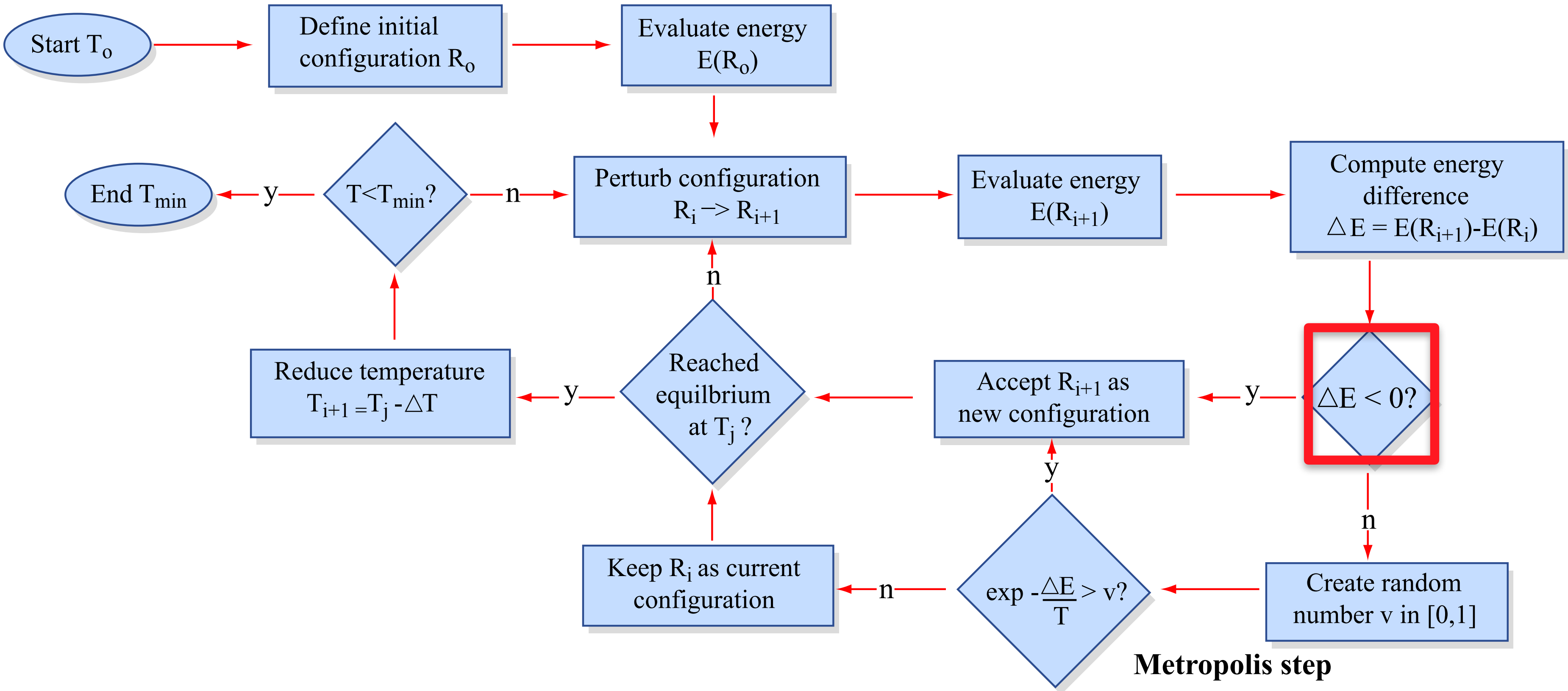
Simulated Annealing



SA Block Diagram

Image by MIT OpenCourseWare.

Simulated Annealing



SA Block Diagram

Image by MIT OpenCourseWare.

Simulated Annealing

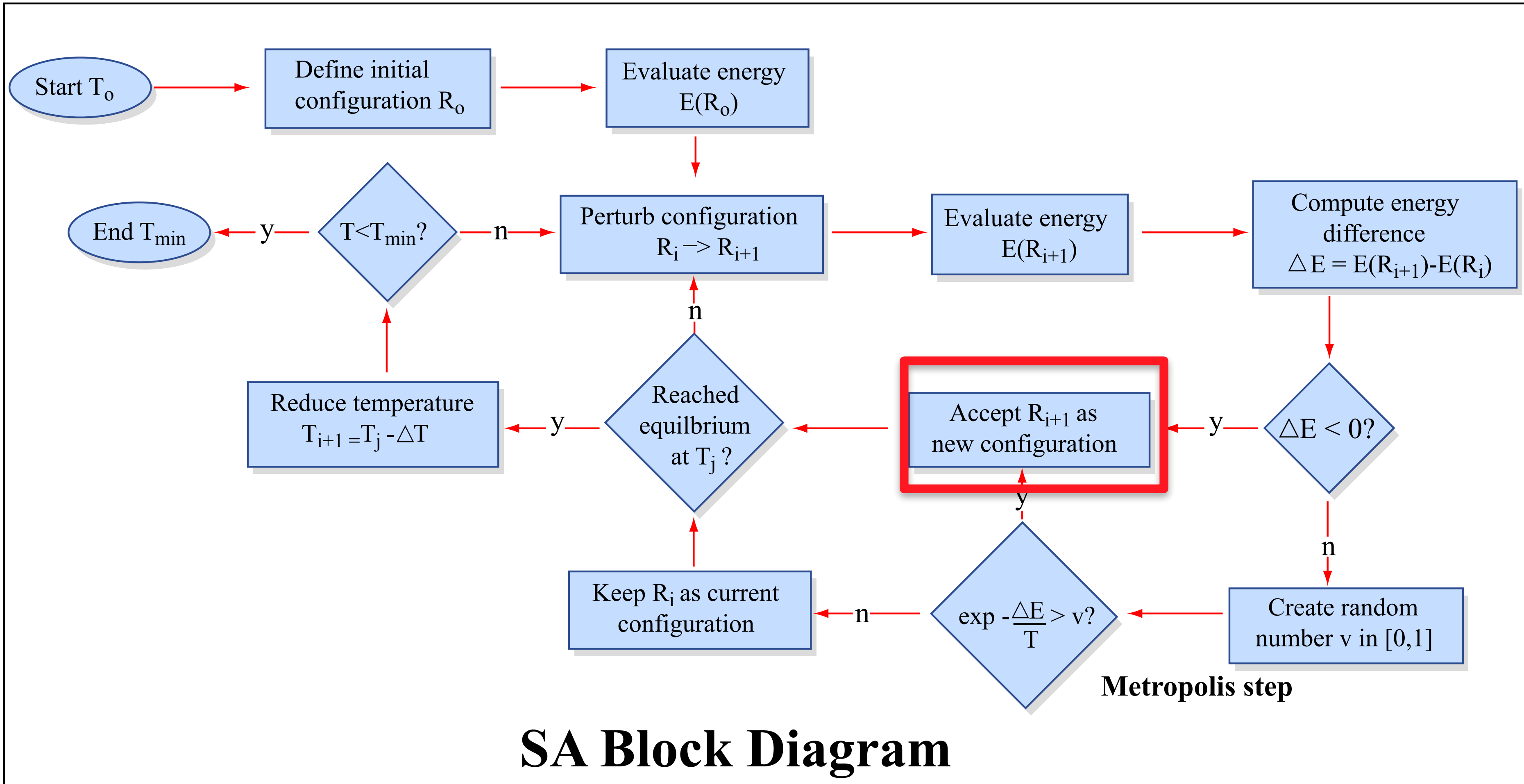


Image by MIT OpenCourseWare.

Simulated Annealing

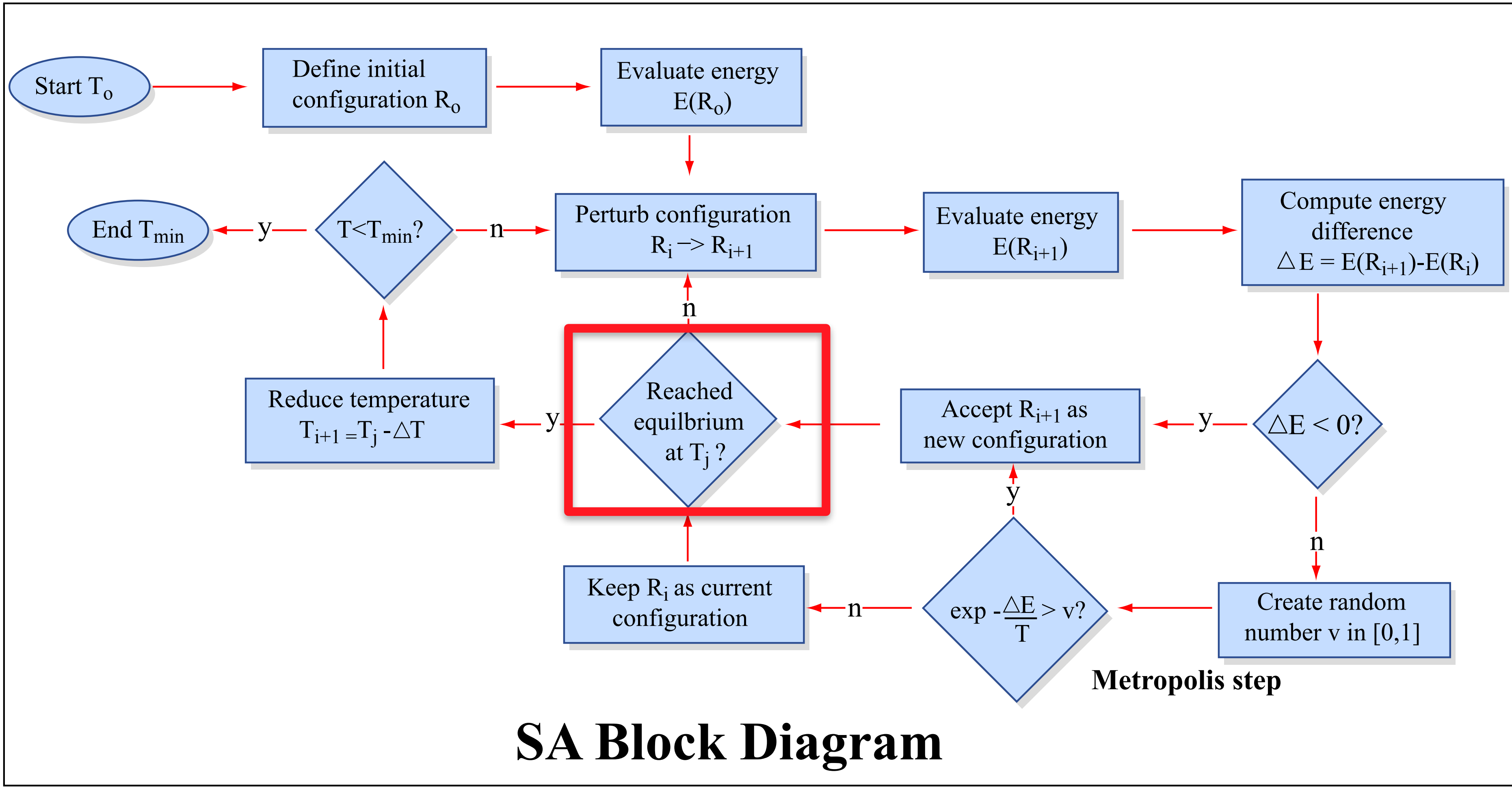


Image by MIT OpenCourseWare.

Simulated Annealing

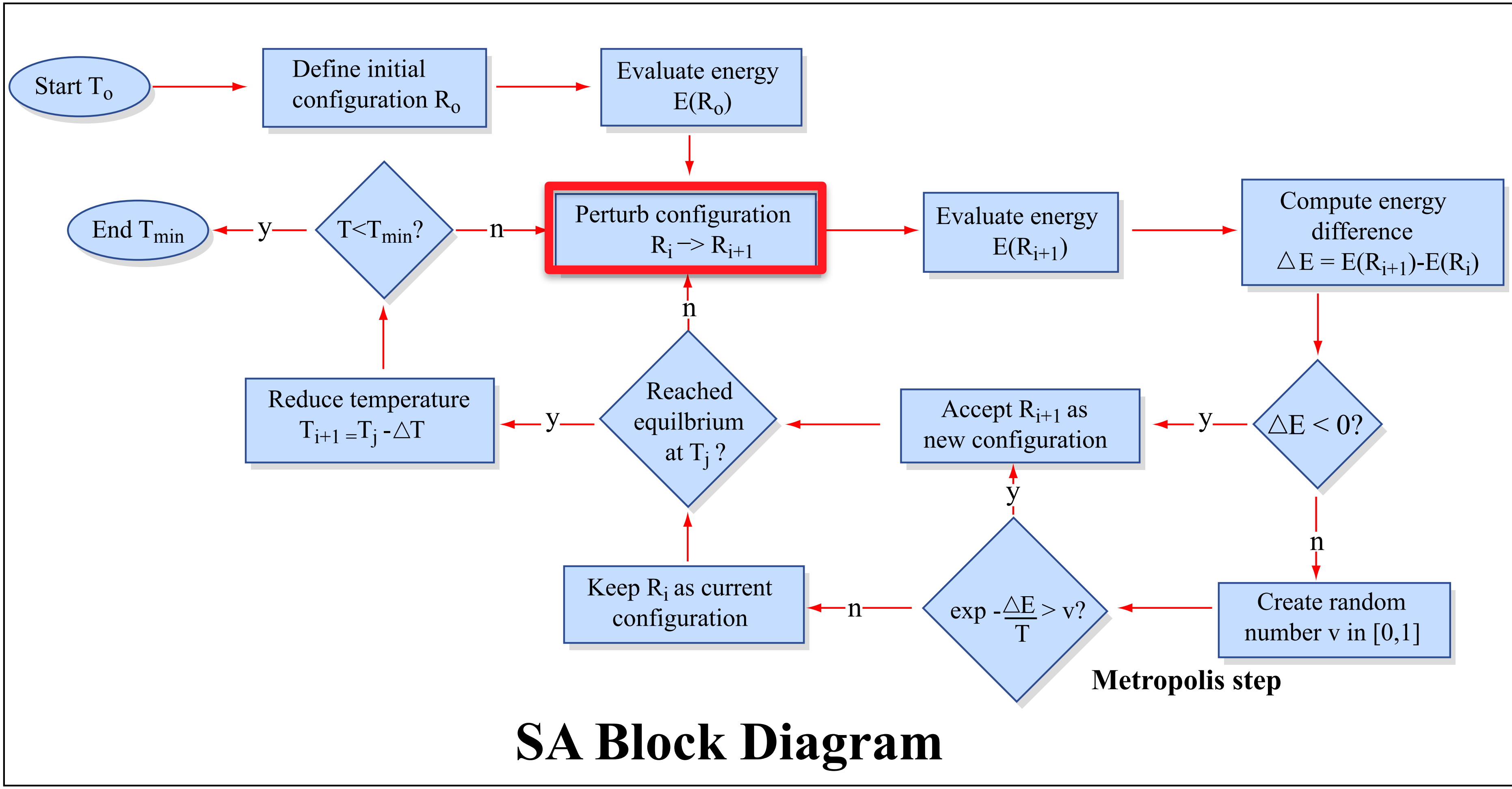


Image by MIT OpenCourseWare.

Simulated Annealing

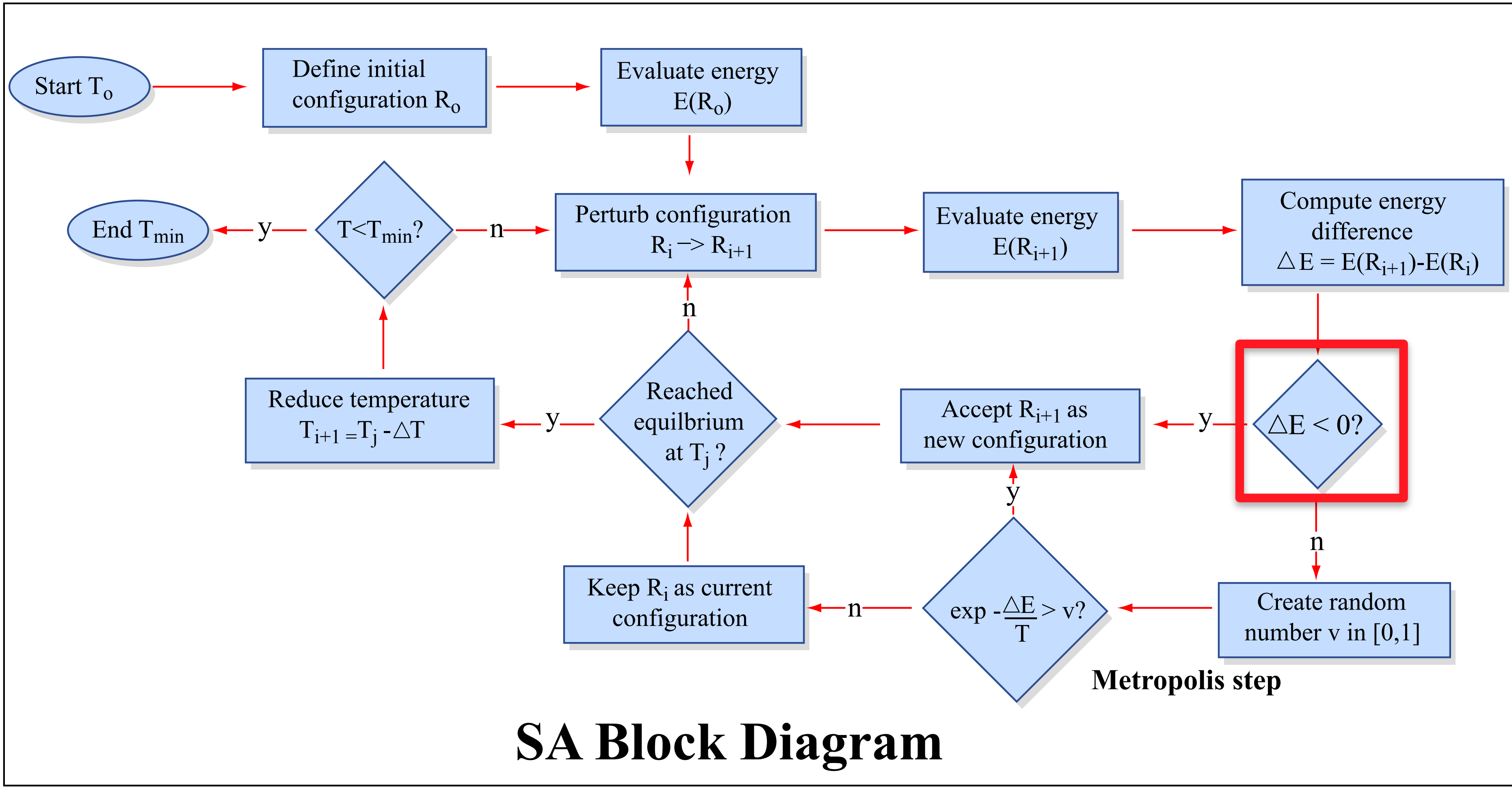


Image by MIT OpenCourseWare.

Simulated Annealing

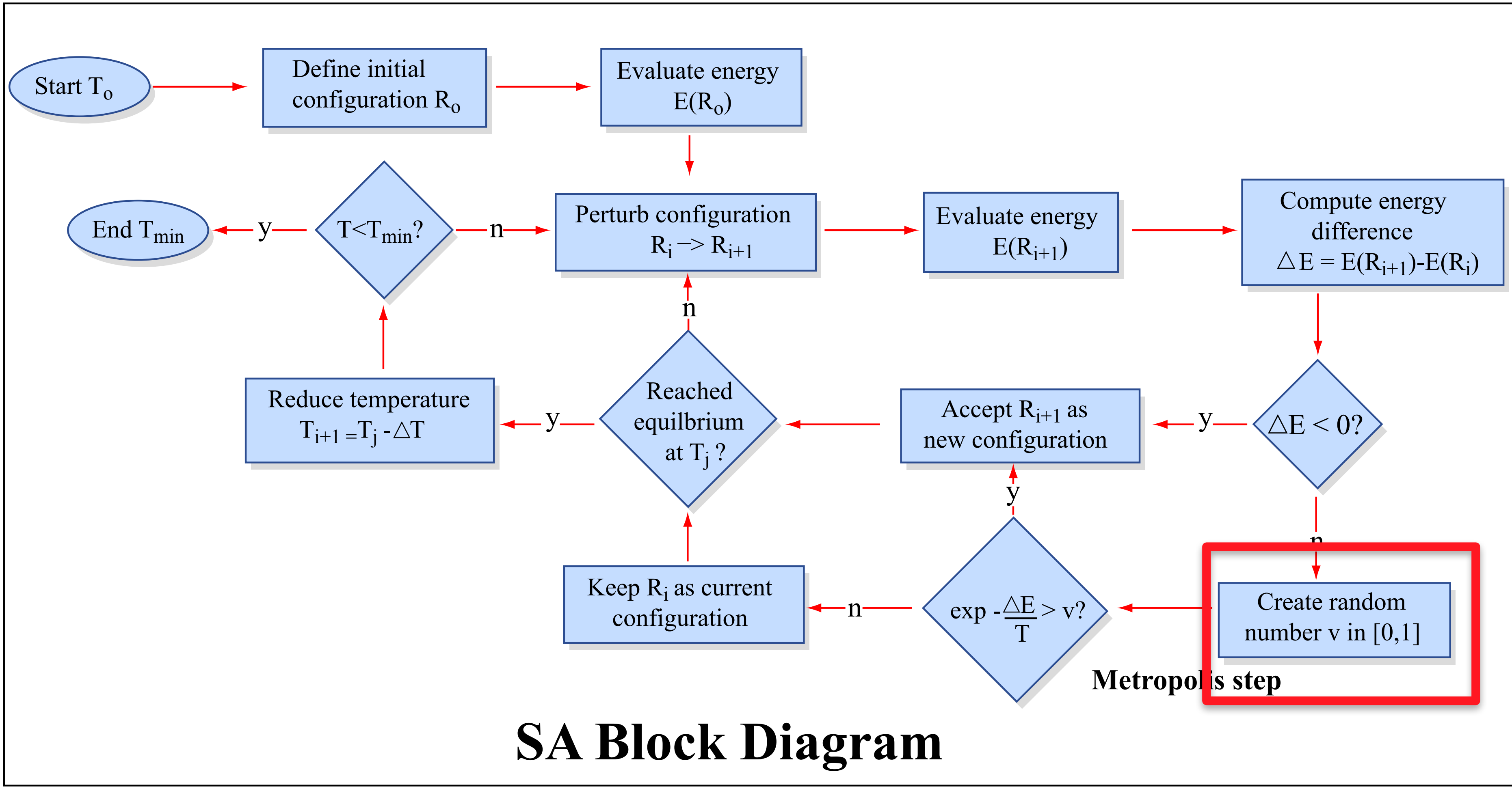


Image by MIT OpenCourseWare.

Simulated Annealing

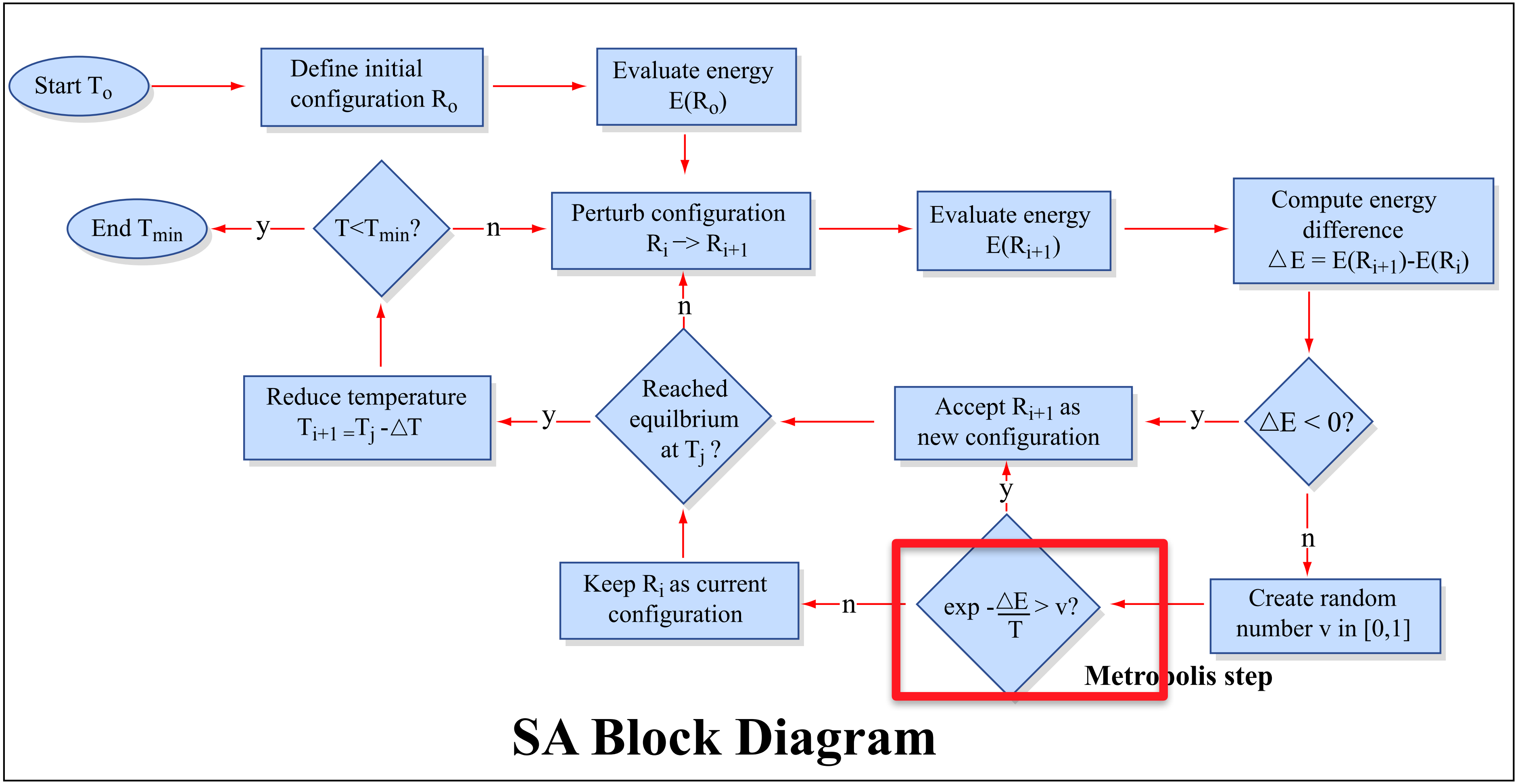


Image by MIT OpenCourseWare.

Simulated Annealing

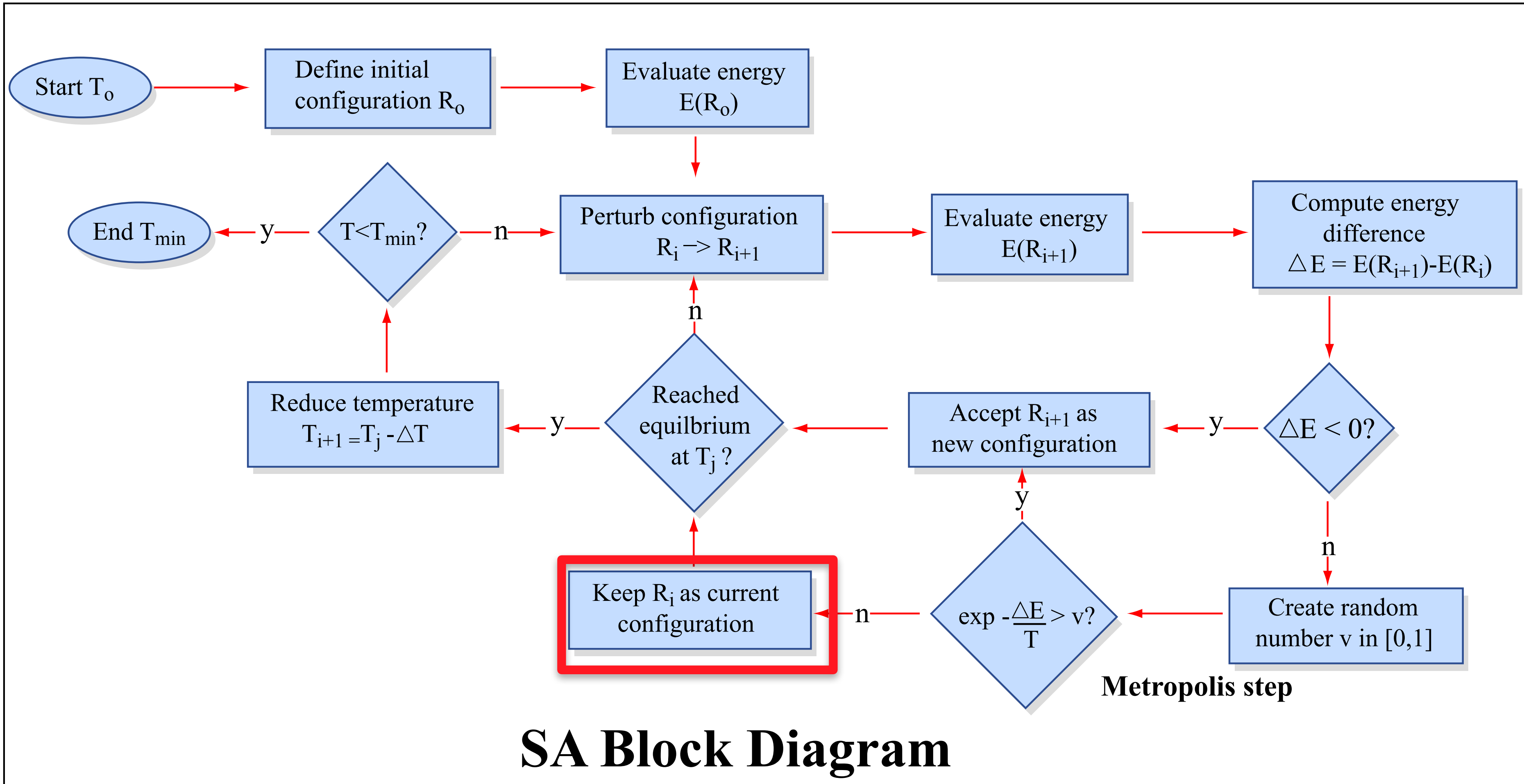


Image by MIT OpenCourseWare.

Simulated Annealing

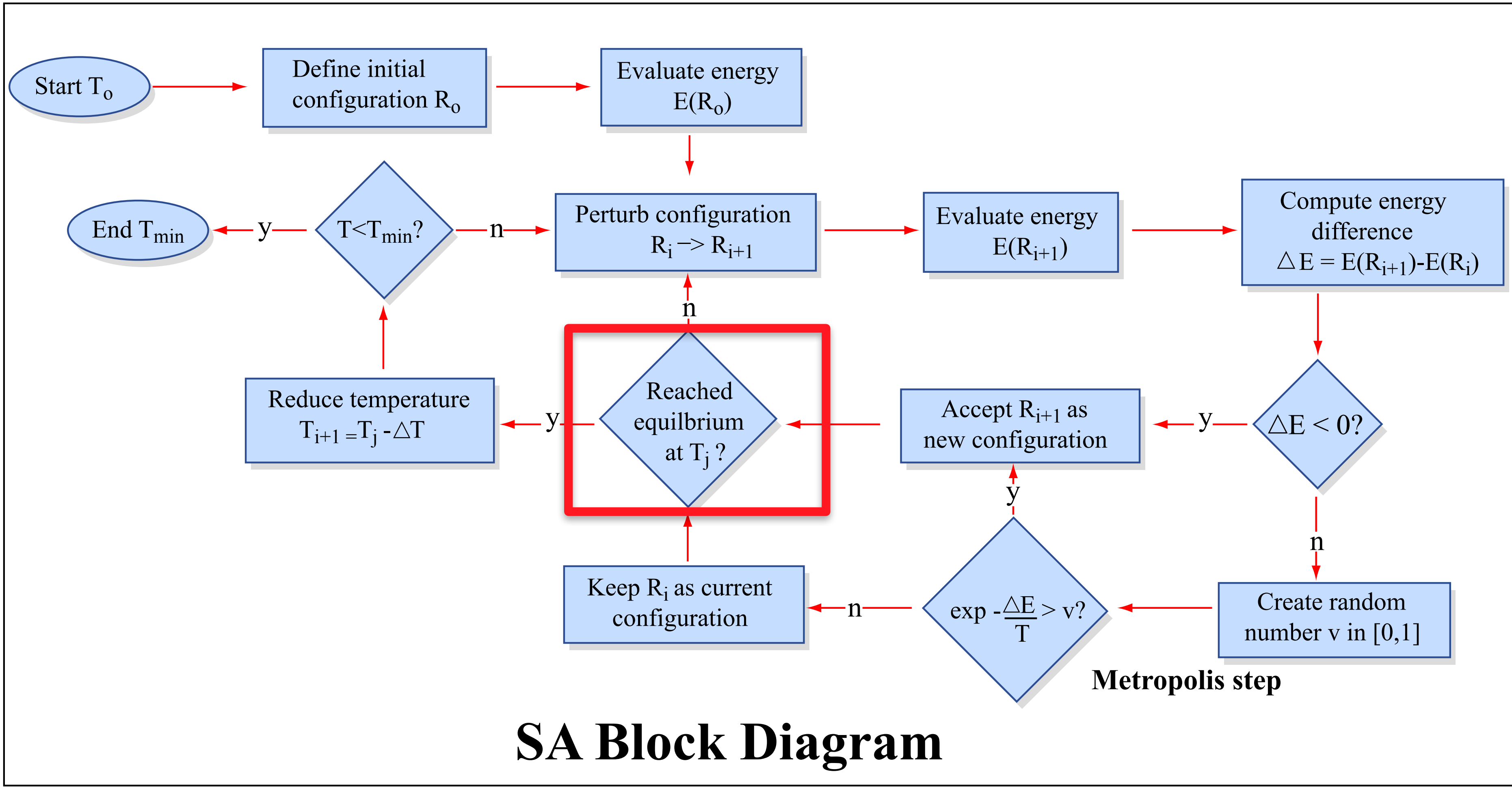


Image by MIT OpenCourseWare.

Simulated Annealing

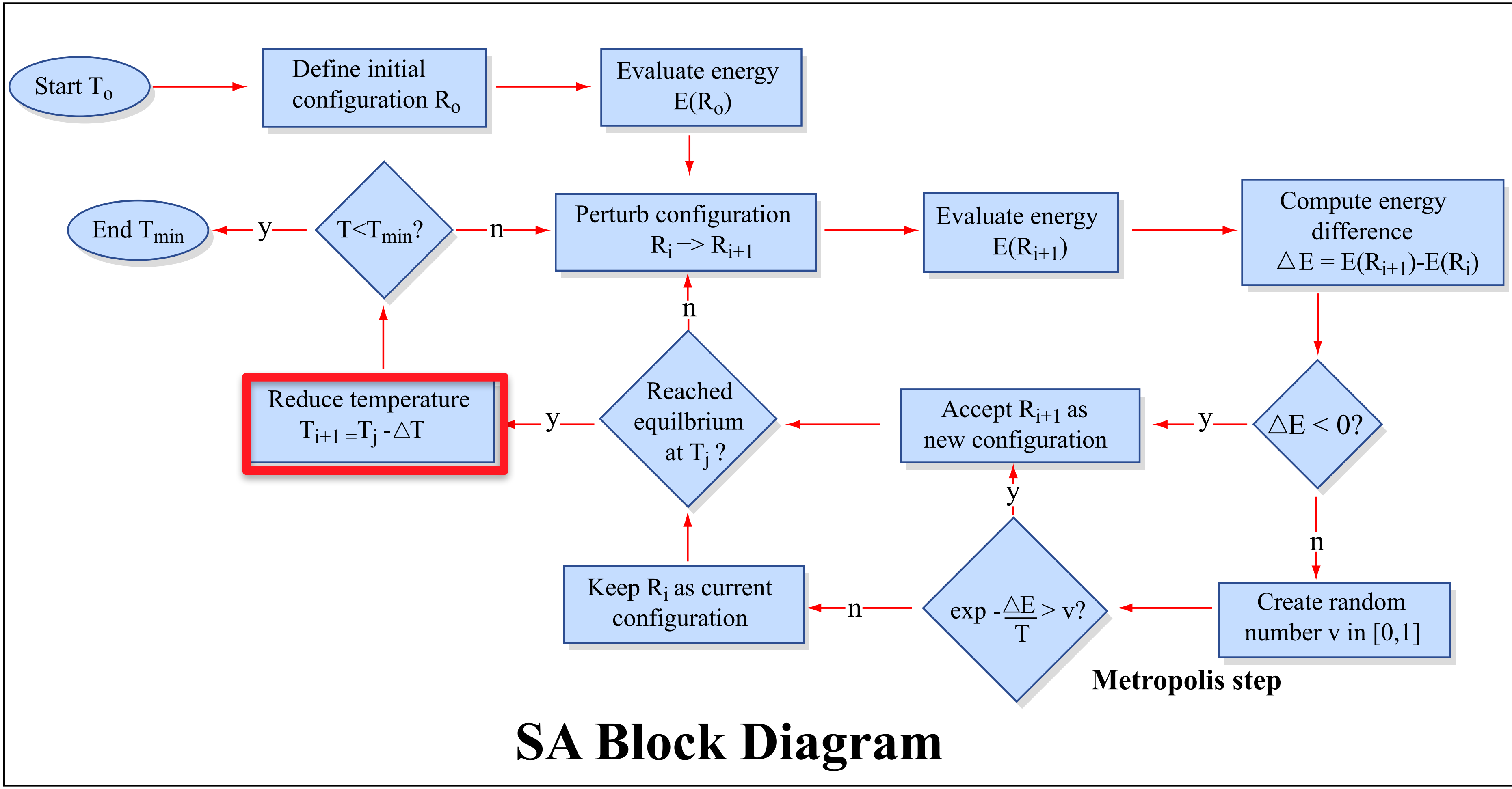
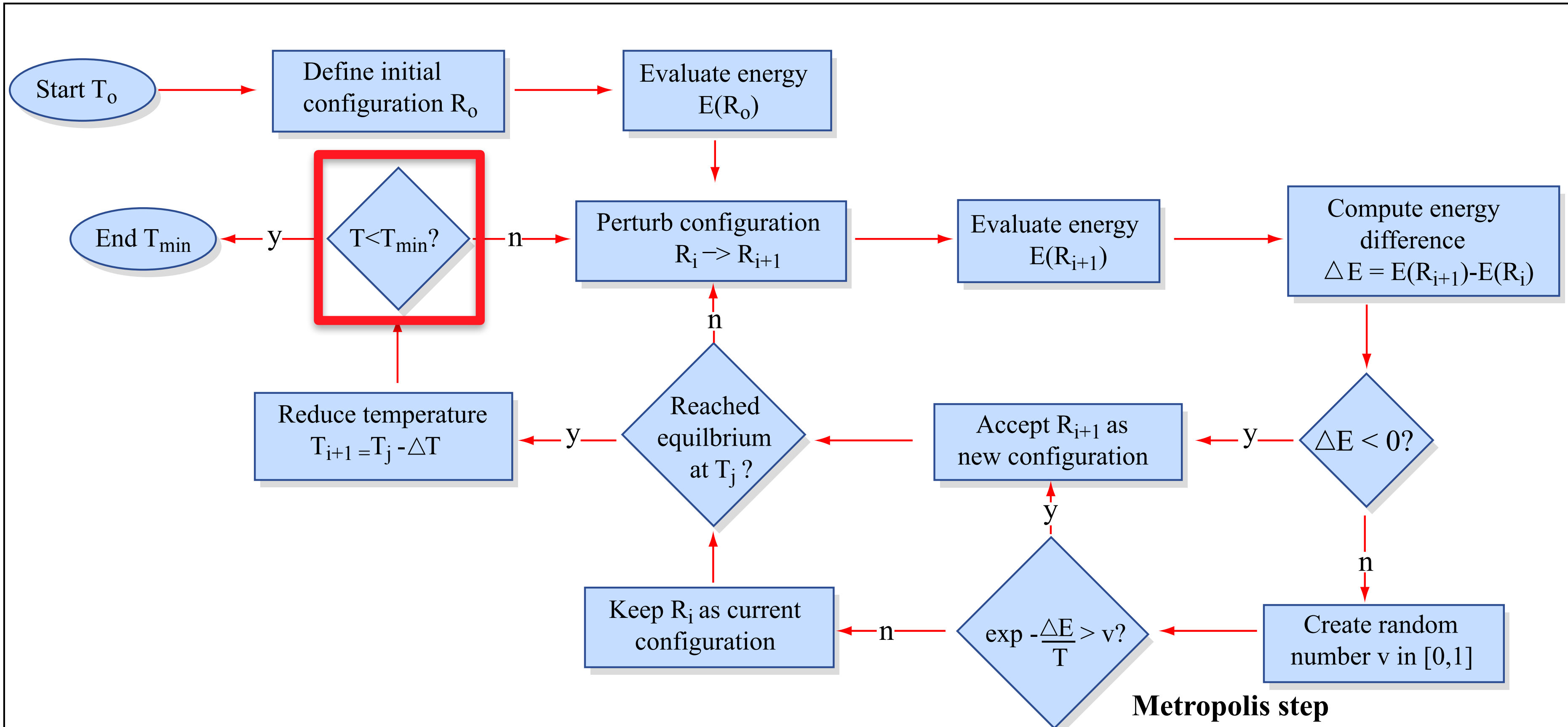


Image by MIT OpenCourseWare.

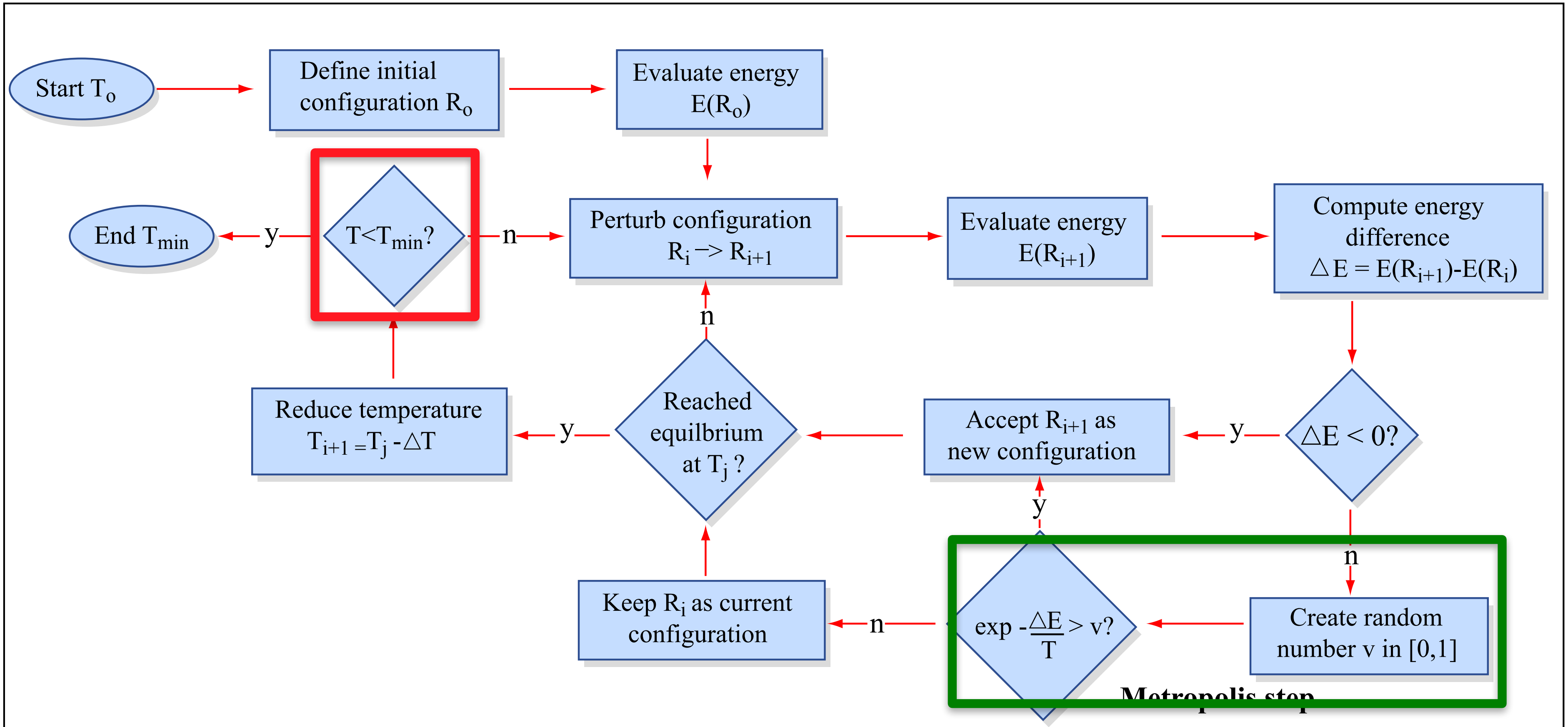
Simulated Annealing



SA Block Diagram

Image by MIT OpenCourseWare.

Simulated Annealing



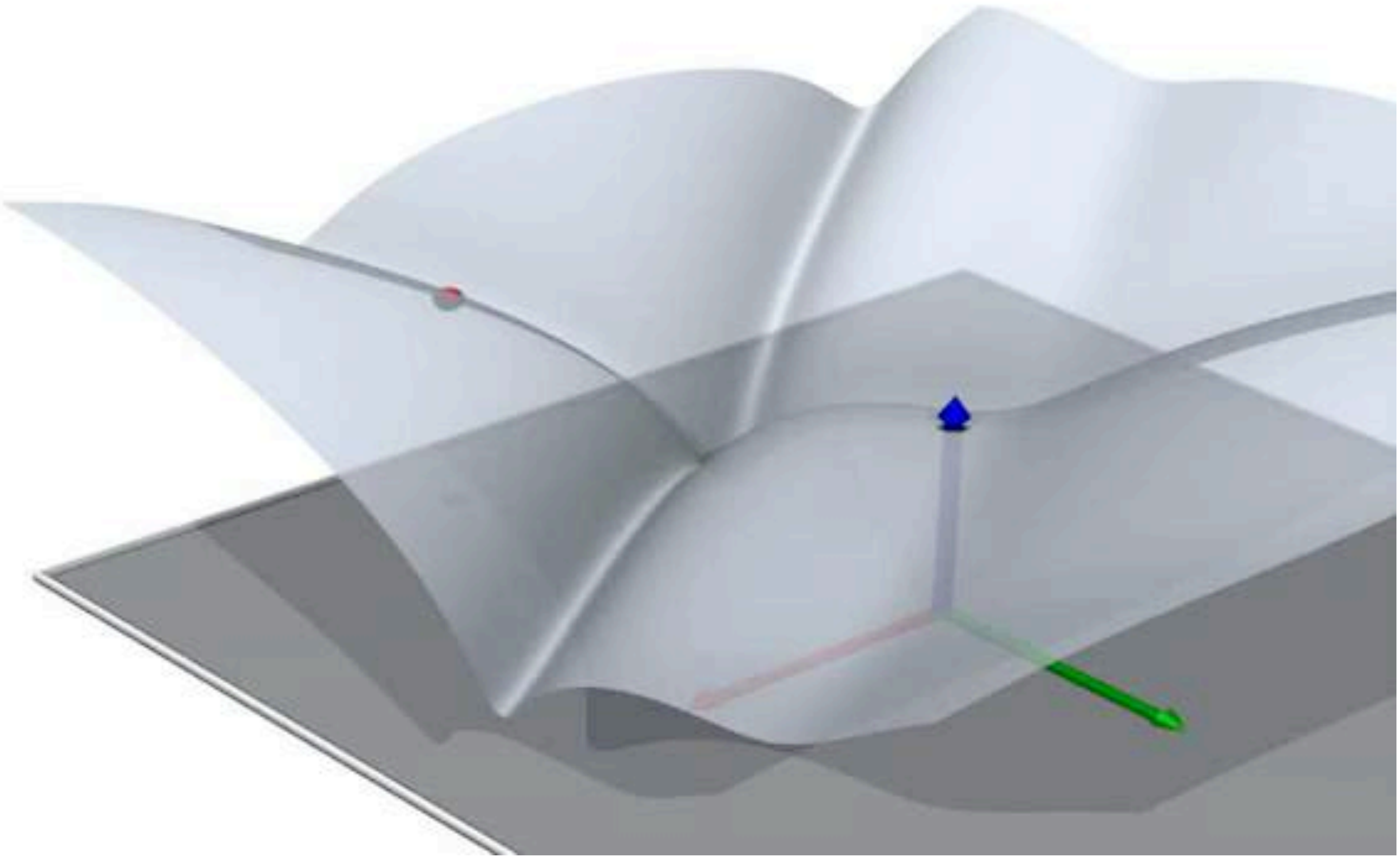
SA Block Diagram

Image by MIT OpenCourseWare.

Simulated Annealing

- Global optimization
- Combinatorial optimization
- Difficult to define good annealing schedule and neighbor generation scheme

Simulated Annealing



Examples from Graphics

The End

- This is the last topic lecture for the course
- The remainder of the lectures will be on current research in computational fabrication