PY410 / 505
Computational Physics 1

Salvatore Rappoccio

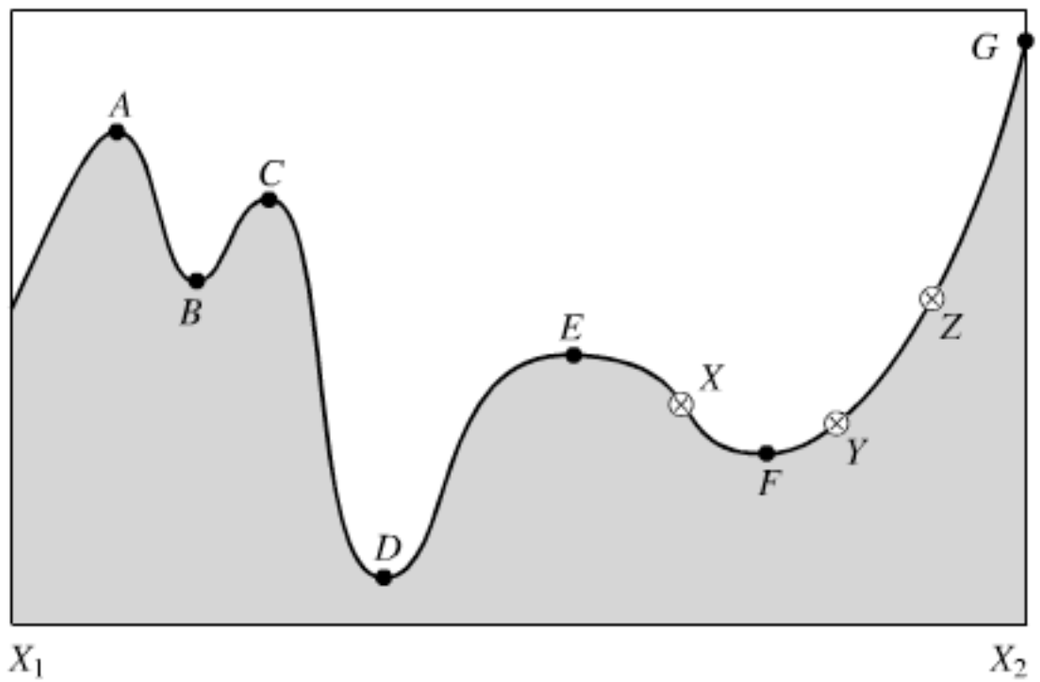
Minimization and maximization

- Many times we're just interested in the equilibria of nonlinear systems :
  - N-body problem in orbits
  - Nonlinear potentials
  - Dynamic behavior far from equilibria
- A good thing to have in your toolbox is to compute zeroes and extrema of functions
  - We've already done the roots in one dimension
  - Now we're ready to start generalizing this to n dimensions
  - Also ready to compute extrema (mins and maxes)

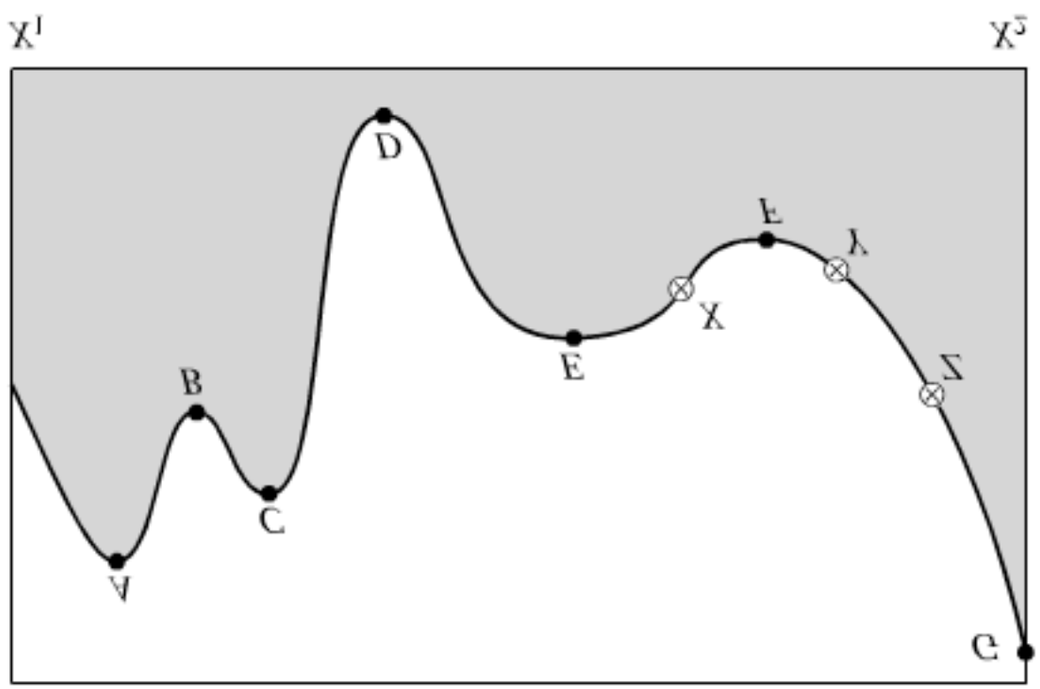
Minimization and maximization

- Minimization versus maximization : what's the difference?

- Minimization :



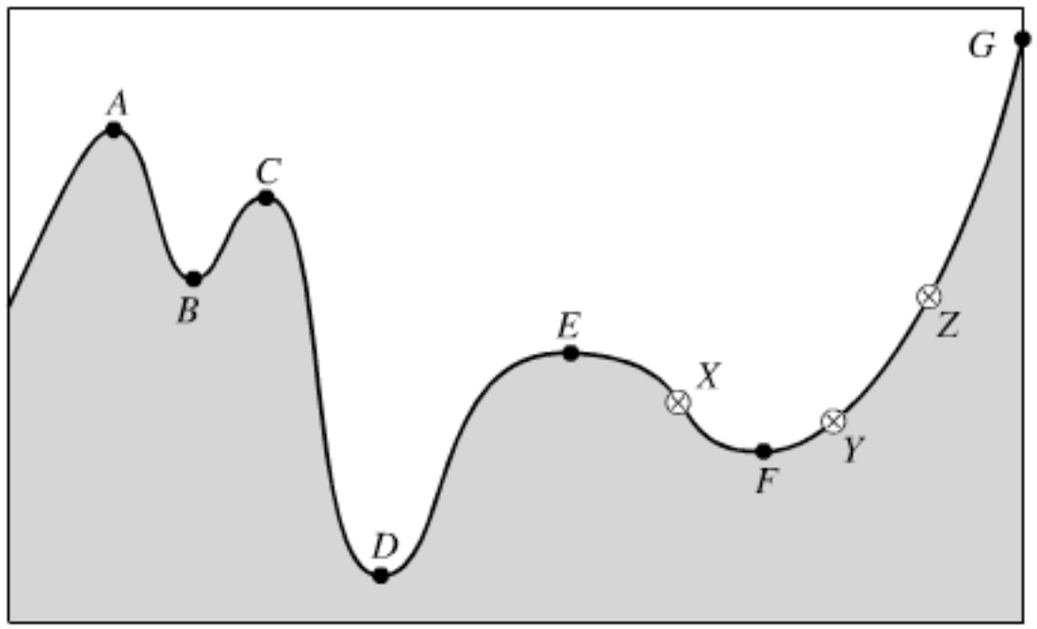
- Maximization :



Minimization and maximization

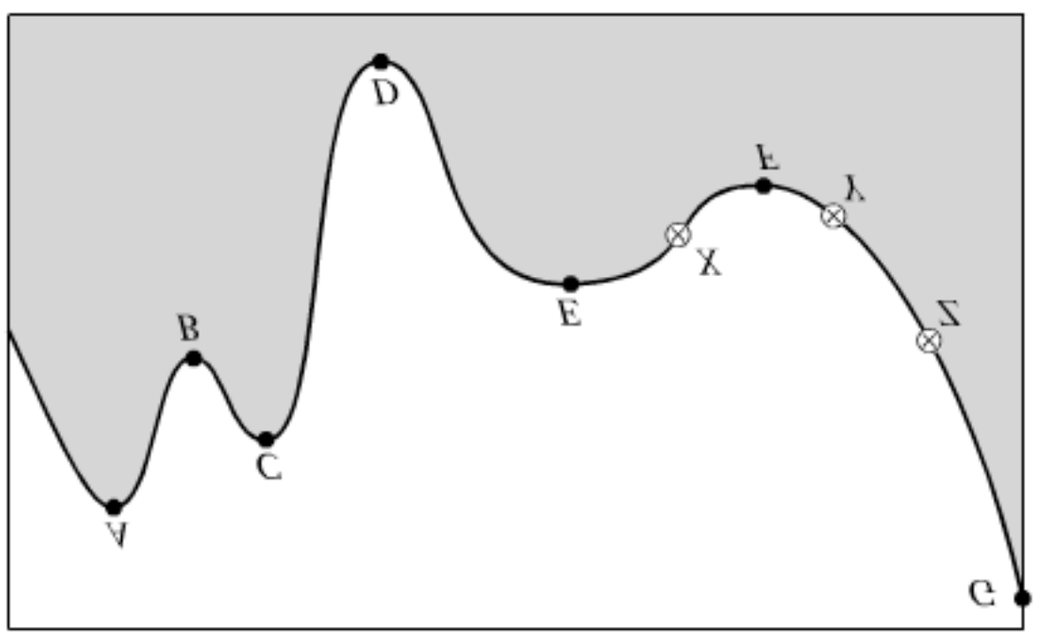
- Minimization versus maximization : what's the difference?

- Minimization :



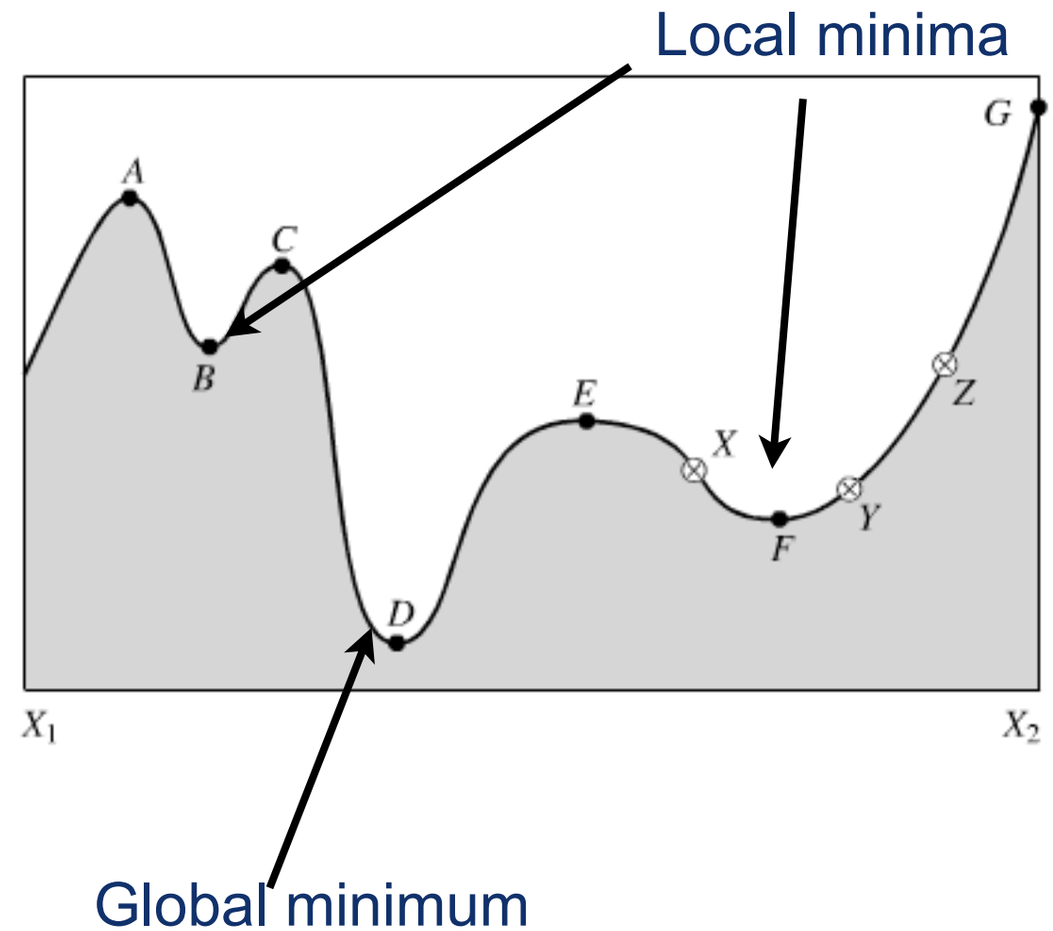
Just minimize -1.0 times the function!

- Maximization :



Minimization and maximization

- A few wrinkles : global versus local extrema require care!
  - Local extrema : easy
  - Global extrema : hard



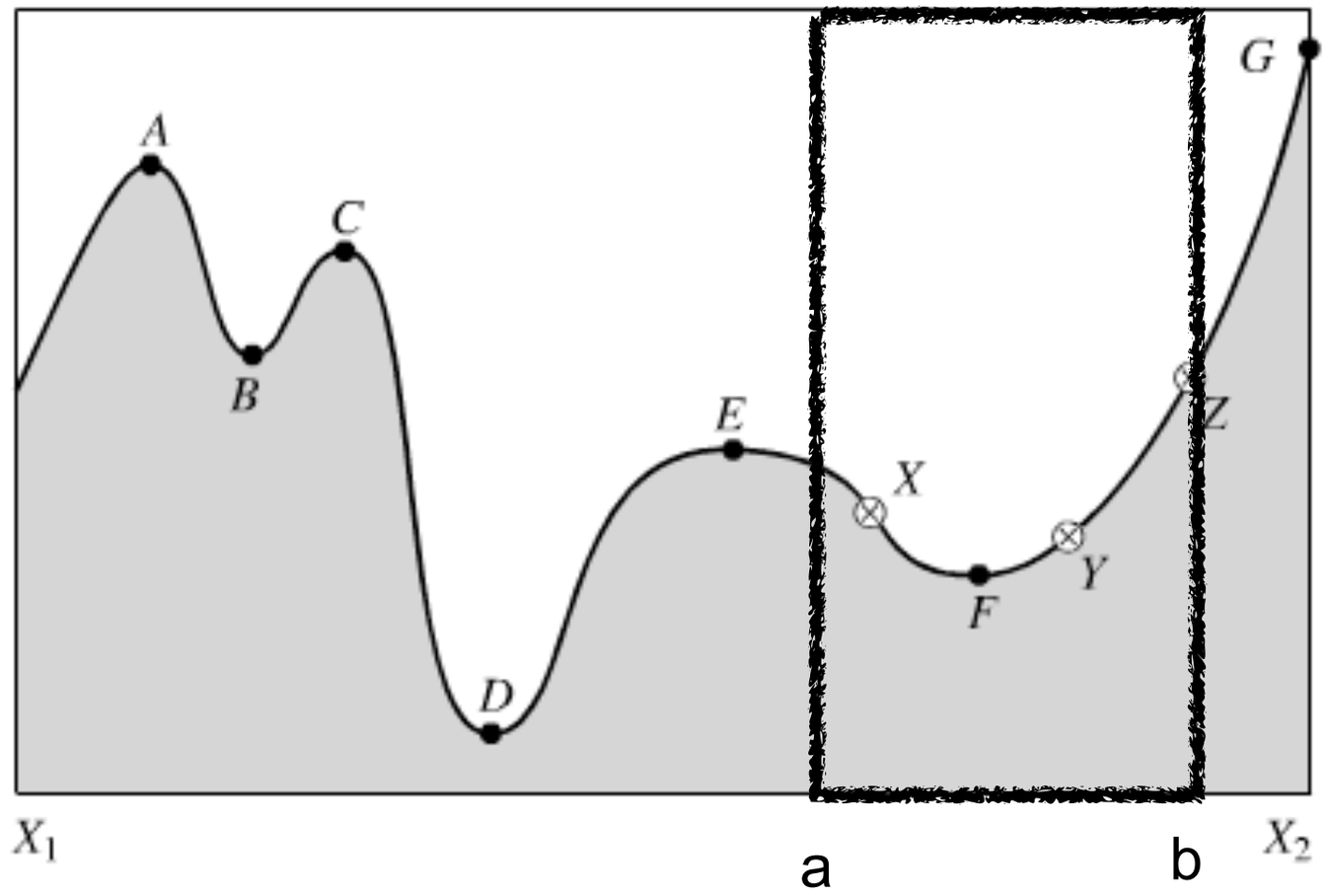
Minimization and maximization

- This is closely related to the problem of finding roots, but here we want roots of the derivative!
- Similar philosophies apply
- Also : there are two cases :
  - If you have/need the derivative
  - If you don't

Minimization and maximization

- First example : Golden Section search

- Very similar to bisection method for finding roots!
  - Bracket extremum in interval [a,b]
  - Iteratively reduce the window until the bracketing interval is sufficiently small

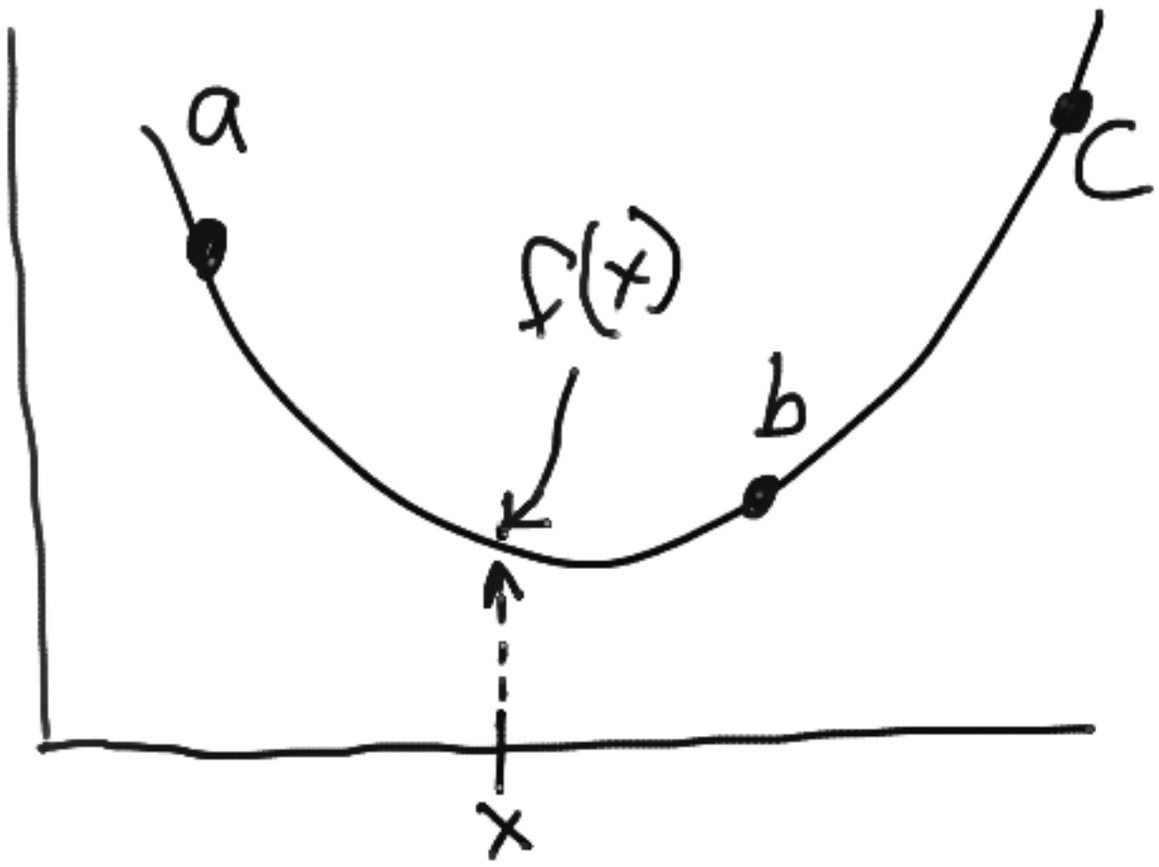


Minimization and maximization

- What does it mean to “bracket” an extremum?

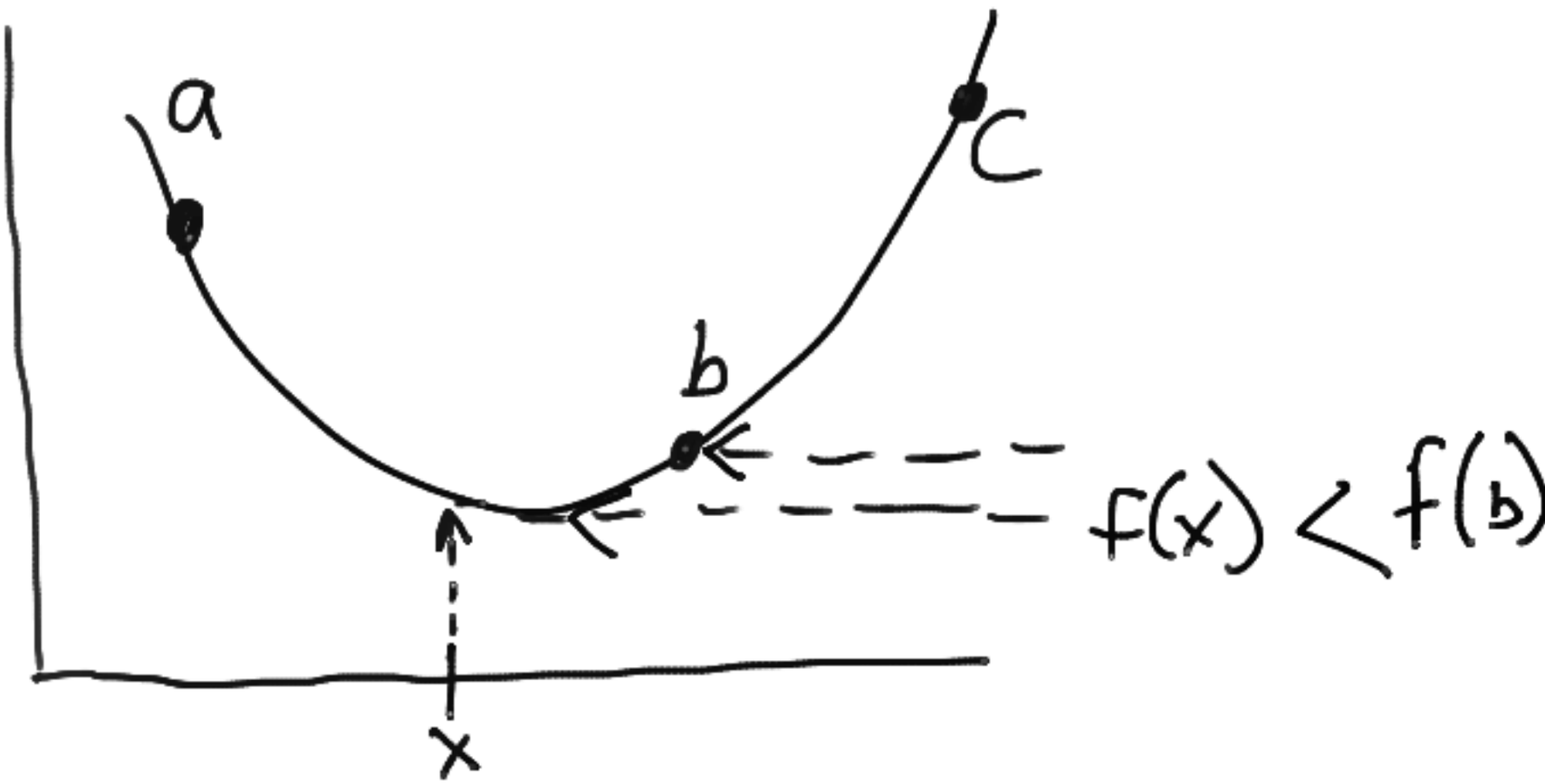
- Pick a,b,c :

- Choose x



Minimization and maximization

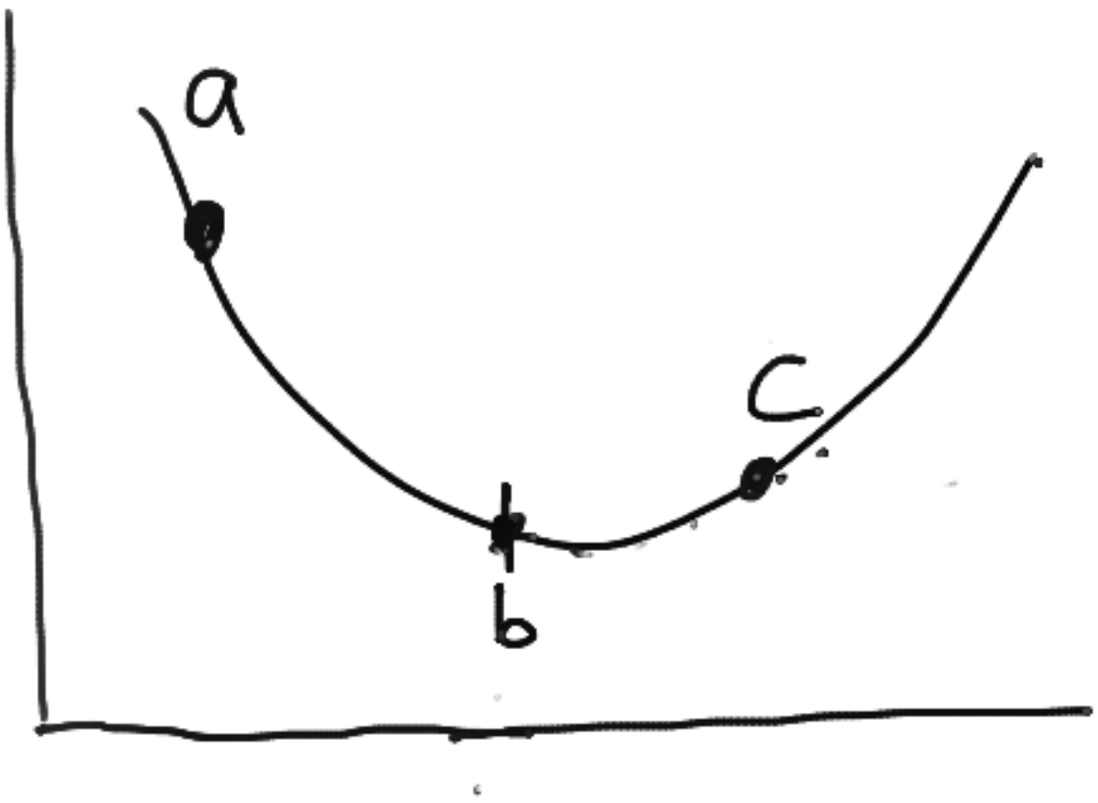
- What does it mean to “bracket” an extremum?



- If f(x) < f(b) :

Minimization and maximization

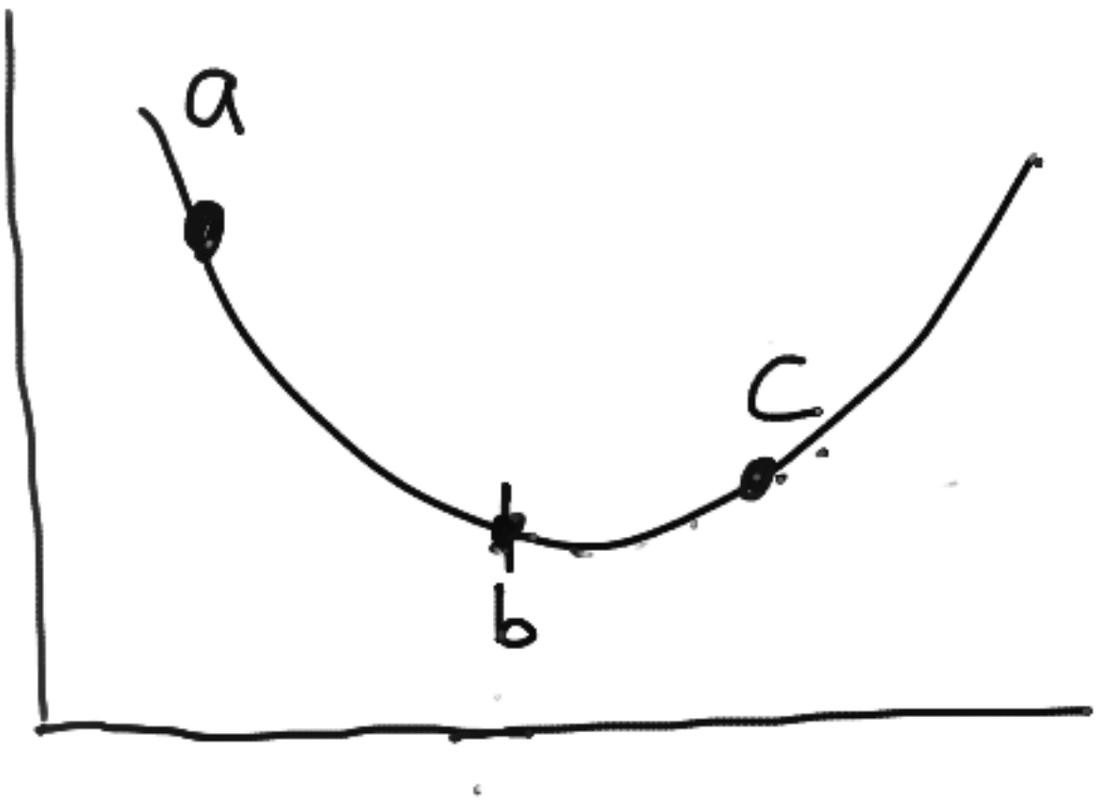
- What does it mean to “bracket” an extremum?



- If f(x) < f(b) :
  - Set
    x---> b,
    b---> c

Minimization and maximization

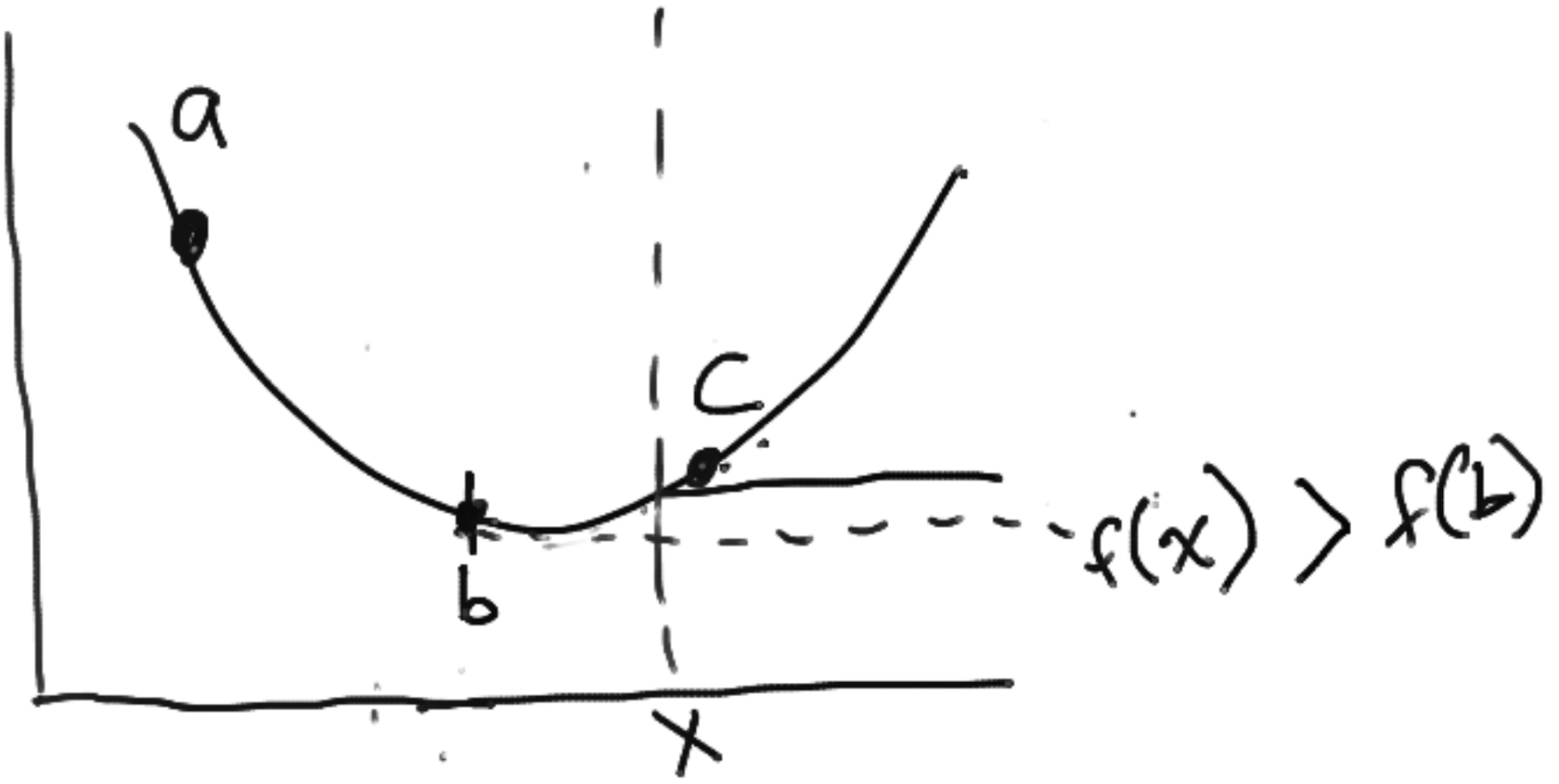
- What does it mean to “bracket” an extremum?



- Repeat :

Minimization and maximization

- What does it mean to “bracket” an extremum?



- If f(x) > f(b) :
  - Set
    x---> c
  - REPEAT!

Minimization and maximization

- Putting it together :

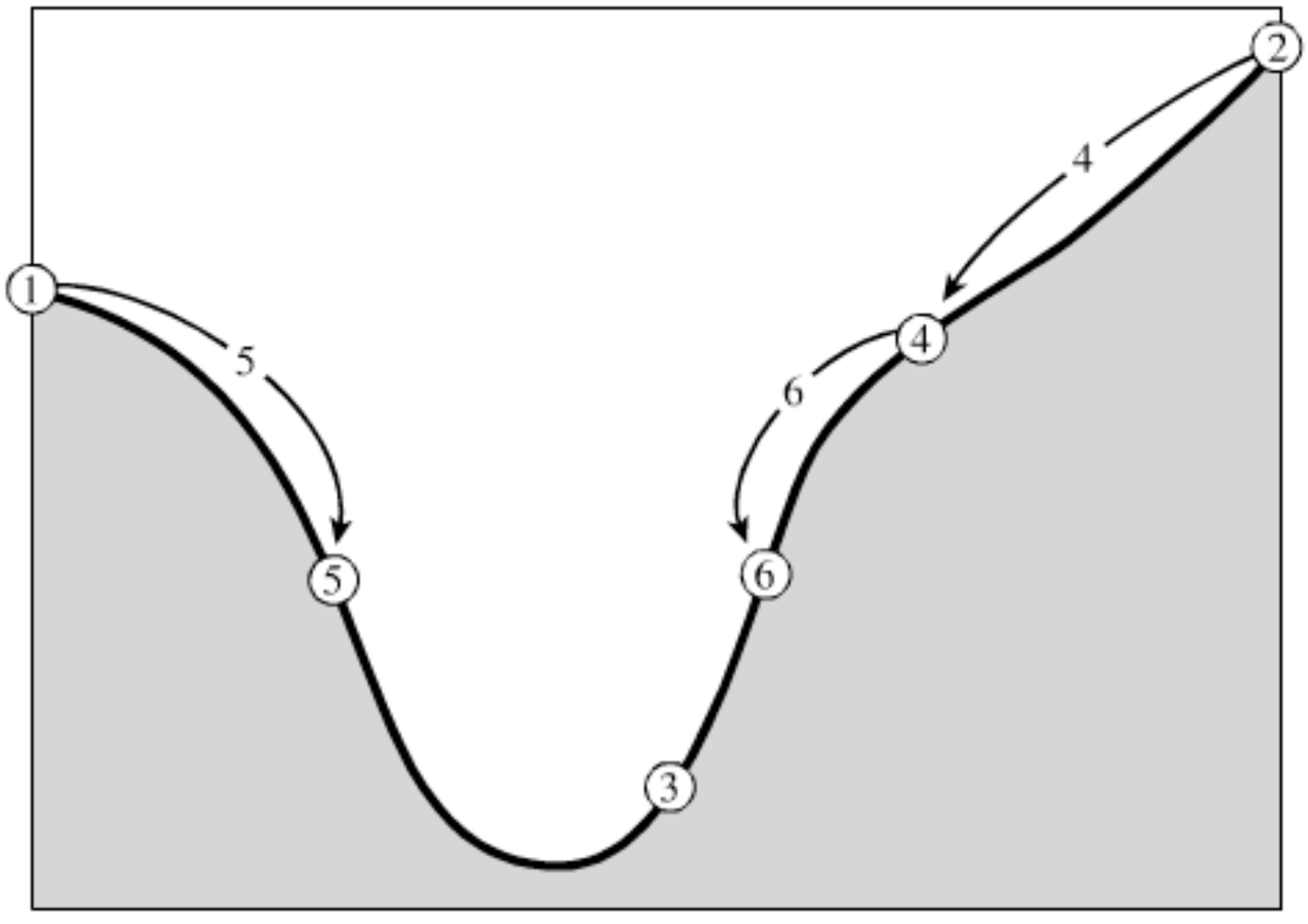


Figure 10.1.1. Successive bracketing of a minimum. The minimum is originally bracketed by points 1,3,2. The function is evaluated at 4, which replaces 2; then at 5, which replaces 1; then at 6, which replaces 4. The rule at each stage is to keep a center point that is lower than the two outside points. After the steps shown, the minimum is bracketed by points 5,3,6.

Minimization and maximization

- What about accuracy?

- Function near extremum will be

$$f(x) \approx f(b) + \frac{1}{2} f''(b)(x-b)^2$$

- We want the second term to be small compared to the first, which gives us:

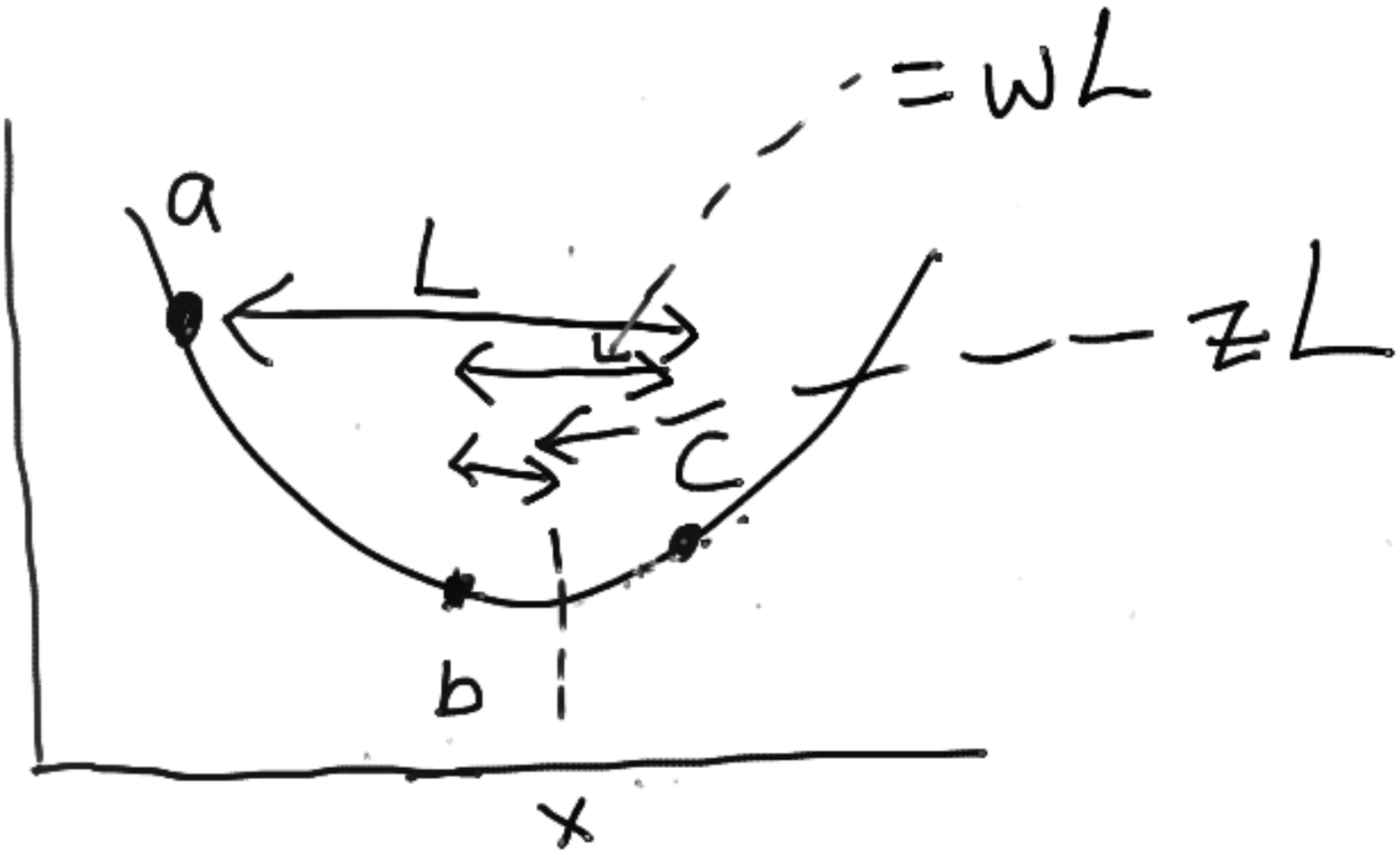
$$|x-b| < \sqrt{\epsilon}|b| \sqrt{\frac{2\,|f(b)|}{b^2 f''(b)}}$$

- If we pick epsilon as the machine precision, then we only get as good as sqrt(epsilon), which is a worse precision (since epsilon < 1)

Minimization and maximization

- Need to pick x given a,b,c
- What to do?

- Define :
  L = c-a
  w = (c-b)/L
  z = (x-b)/L



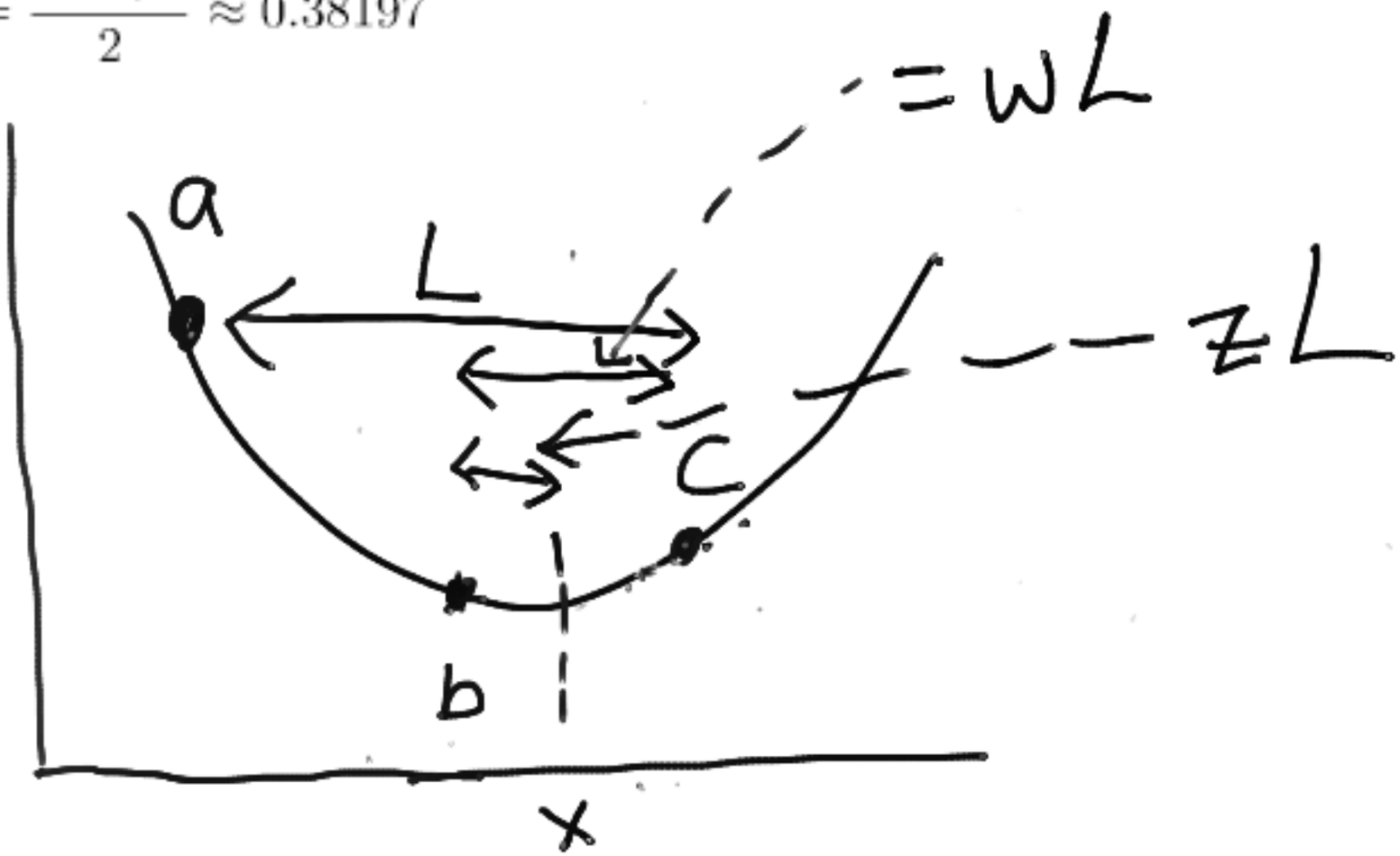
- Then the next bracketing segment will be either :
  w + z or 1 - w

- Choose z to make these equal!

Minimization and maximization

- Solving for z :

$$w^2 - 3w + 1 = 0 \qquad \text{yielding} \qquad w = \frac{3 - \sqrt{5}}{2} \approx 0.38197$$

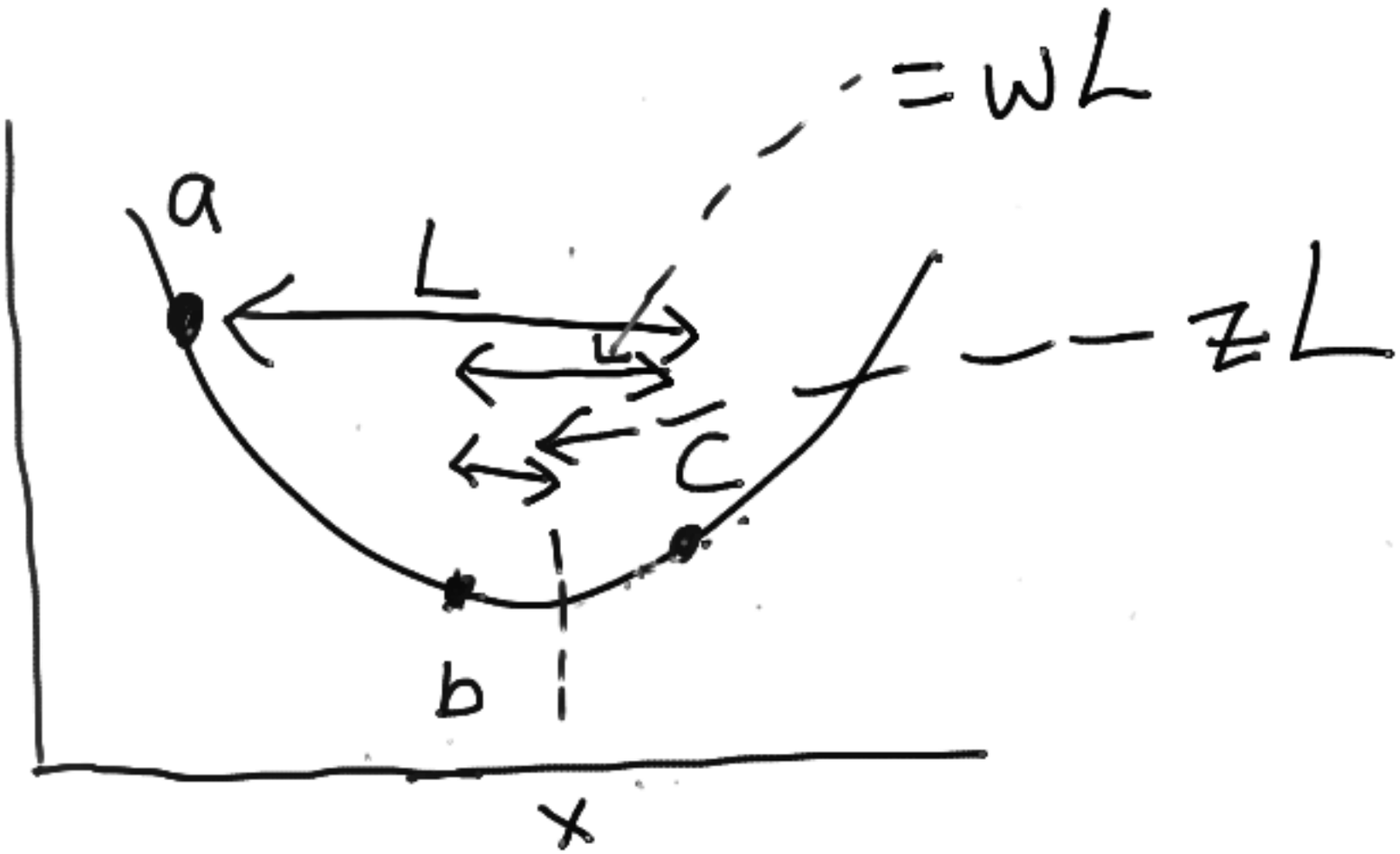


- Again with the golden section!

- Because of this, it’s called the “golden section search”

Minimization and maximization

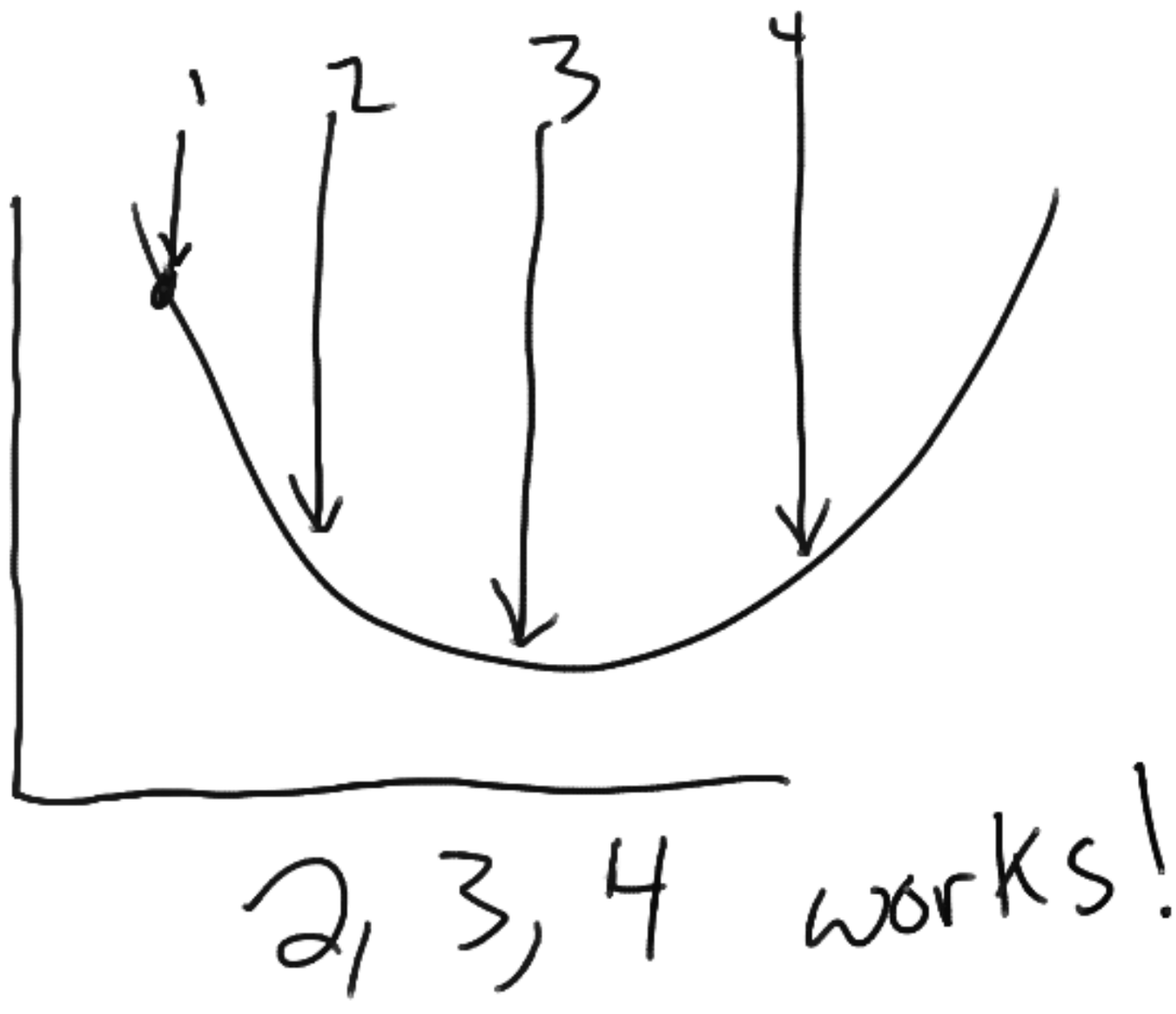
- Pick a,b,c
- Let x = 0.38197* max (c-b, b-a)



Minimization and maximization

- First : we need to know that we've actually chosen a, b, and c such that they bracket the minimum

- Simple strategy :
  - Start with a guess
  - Step "downhill" through the function
  - When you come back up again, then you have a candidate set of a,b,c



Minimization and maximization

- For the combination, then, we :
  - Guess a1 and b1
  - Step through to get candidate a,b,c that bracket extremum
  - Use golden search on a,b,c to get extremum

Minimization and maximization

- Is this too simple?
  - Can be somewhat intensive, yes
  - Next shot at this is to use parabolic interpolation instead of the above strategy
  - Makes sense, we're looking for something parabola-like
  - But! Assumes the function is parabola-like near your extremum and where you're evaluating things

Minimization and maximization

- Parabolic interpolation

  – Assume you have a parabola f(x), and points a,b,c

  – Define x as the minimum:

$$x = b - \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]}$$

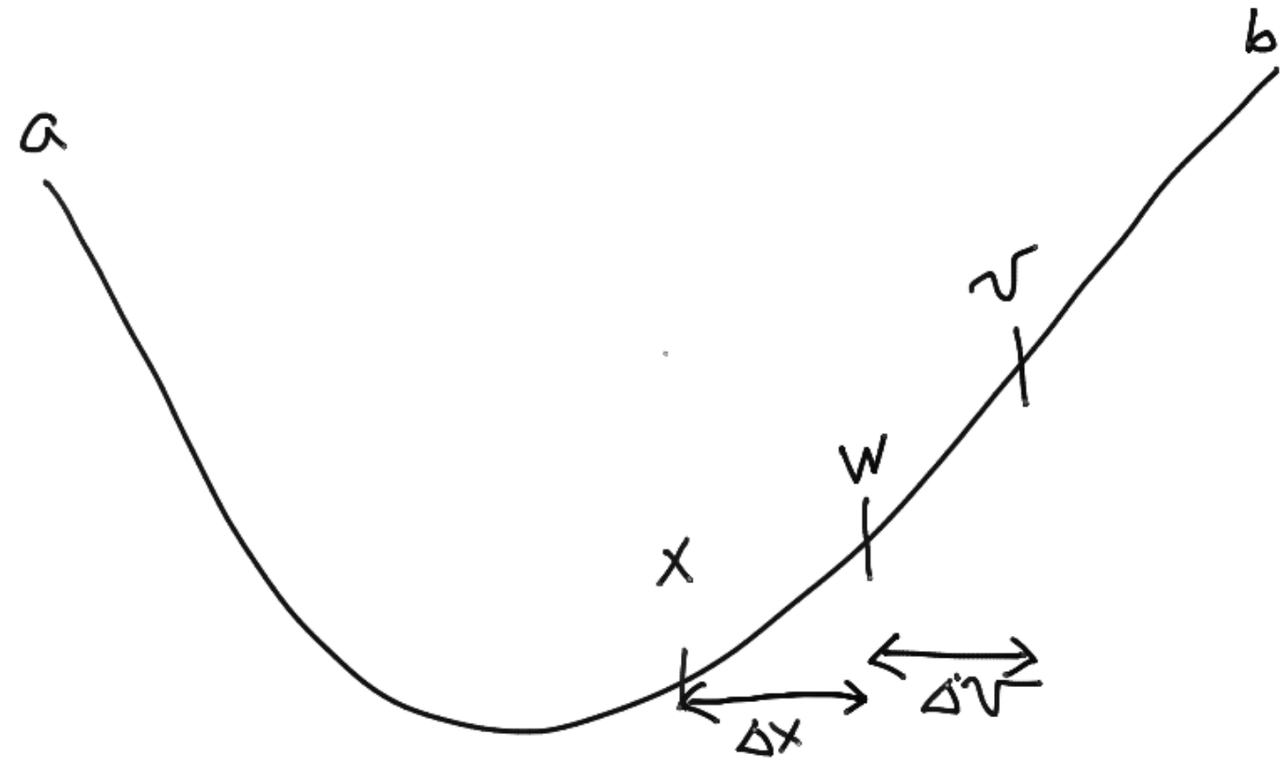
- This fails if the three points are colinear, so need to make sure that isn't the case!

- Also doesn't go so well if the function is not bracketing

- Need a more robust strategy here, then

Minimization and maximization

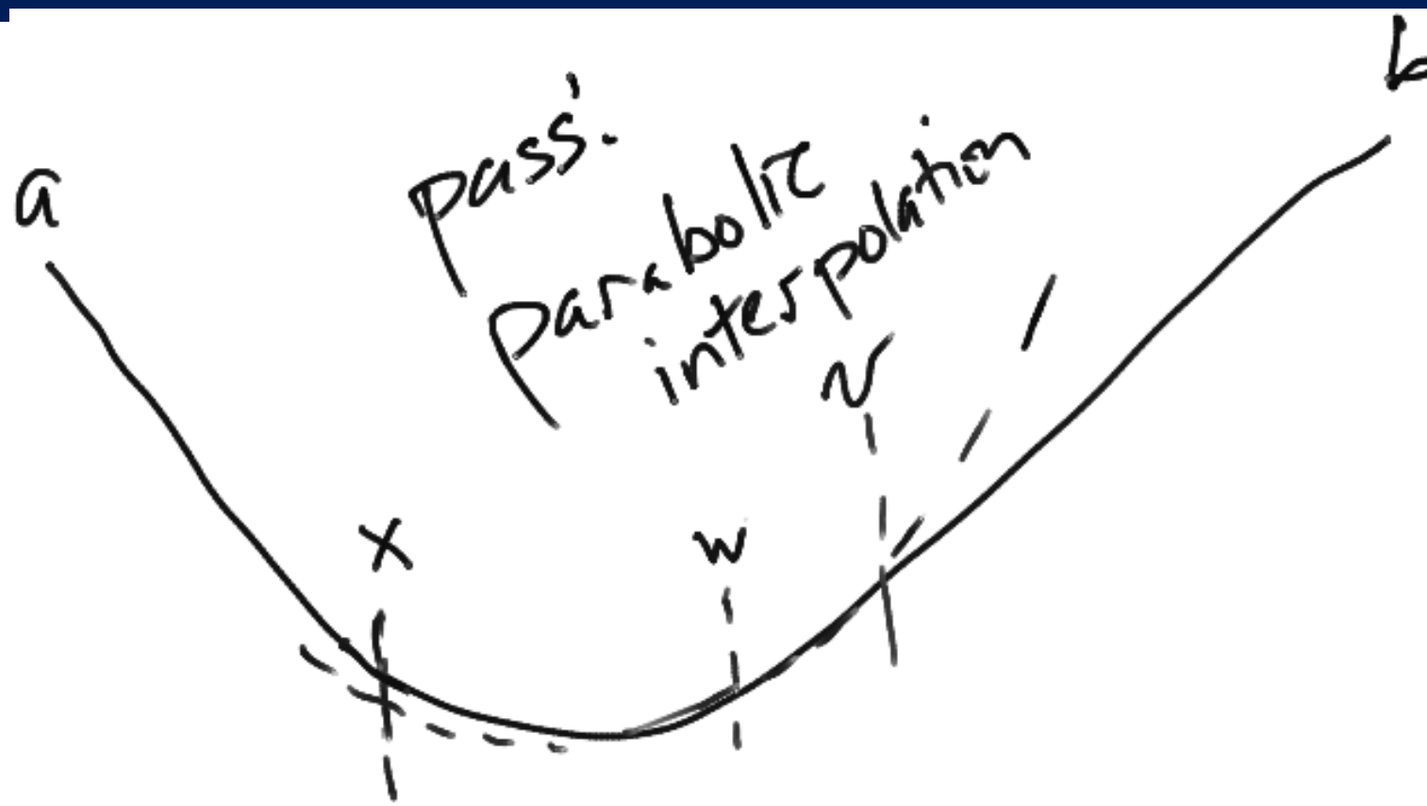
- Brent's method:
  - Pick bounds (a,b)
  - Find x = minimum of points "so far"
  - Let w = second-best minimum "so far"
  - Let v = previous value of w



  - Attempt parabolic interpolation between (x,v,w)
  - success = a < x < b, and "delta x" < 0.5 * "delta v"
    - Second bit prevents "bouncing around"
  - if success : parabolic interpolation
  - else : golden section interpolation
  - In the worst case, this alternates (mathematically) between parabolic steps and golden sections

Minimization and maximization

- Brent's method:
  - Pick bounds (a,b)
  - Find x = minimum of points "so far"
  - Let w = second-best minimum "so far"
  - Let v = previous value of w



  - Attempt parabolic interpolation between (x,v,w)
  - success = a < x < b, and "delta x" < 0.5 * "delta v"
    - Second bit prevents "bouncing around"
  - if success : parabolic interpolation
  - else : golden section interpolation
  - In the worst case, this alternates (mathematically) between parabolic steps and golden sections

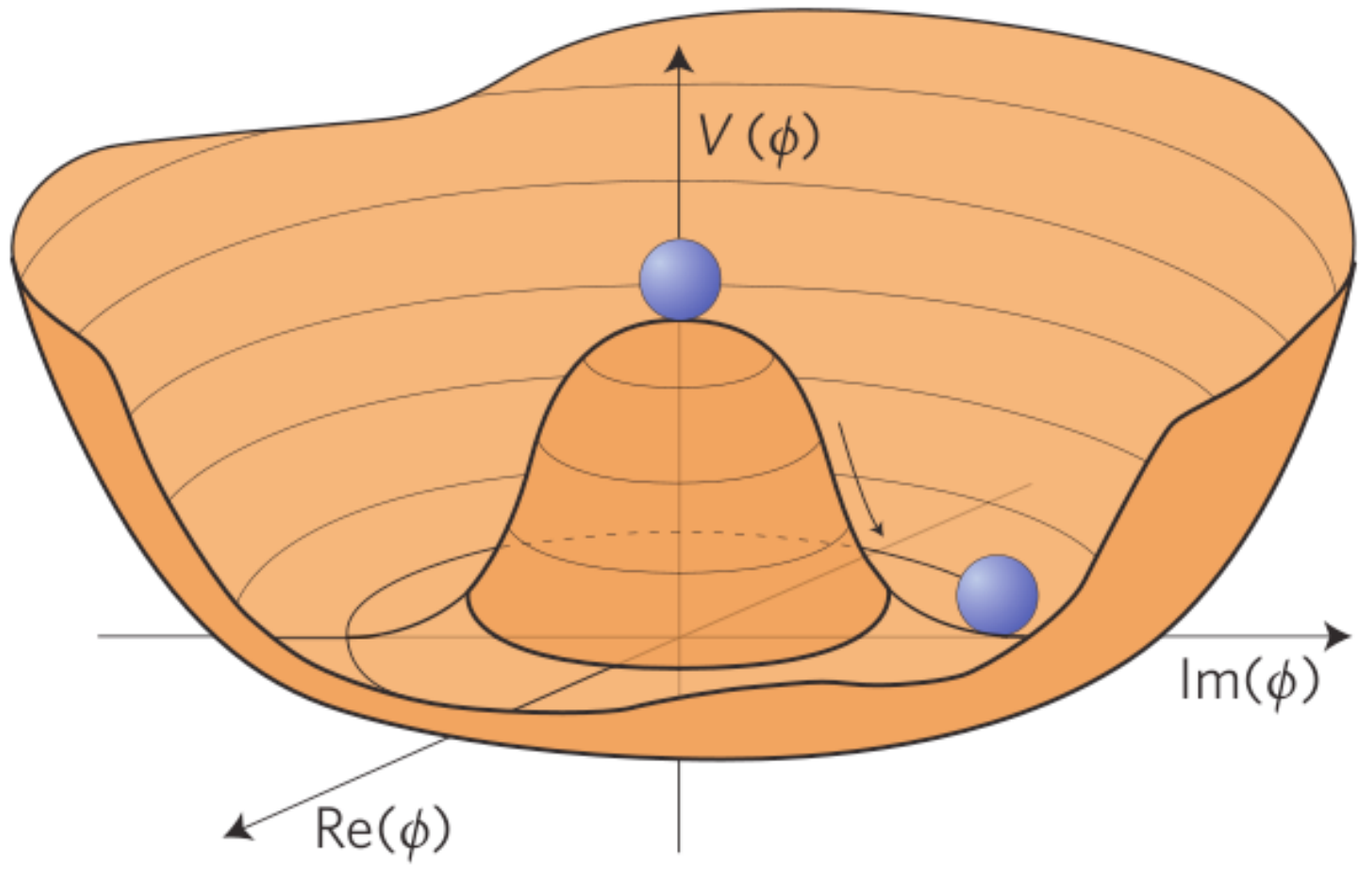
Minimization and maximization

- Very heuristically :
- Bracketing :
  - Iterate “downhill” until you went down and up again
- Minimization :
  - Use golden section alone, or with Brent’s method to find minimum

Application : Higgs potential!

- What is that thing?

- Example of a spontaneous symmetry breaking:
  - Bose-Einstein condensates
  - Higgs potential
  - Ferromagnet



- Field starts off at zero
  - Unstable!

- Decays to the true minimum at some finite value != 0

Application : Higgs potential!

- Simple 1-d Higgs potential:

$$V(x) = -\frac{1}{2}x^2 + \frac{1}{4}x^4$$

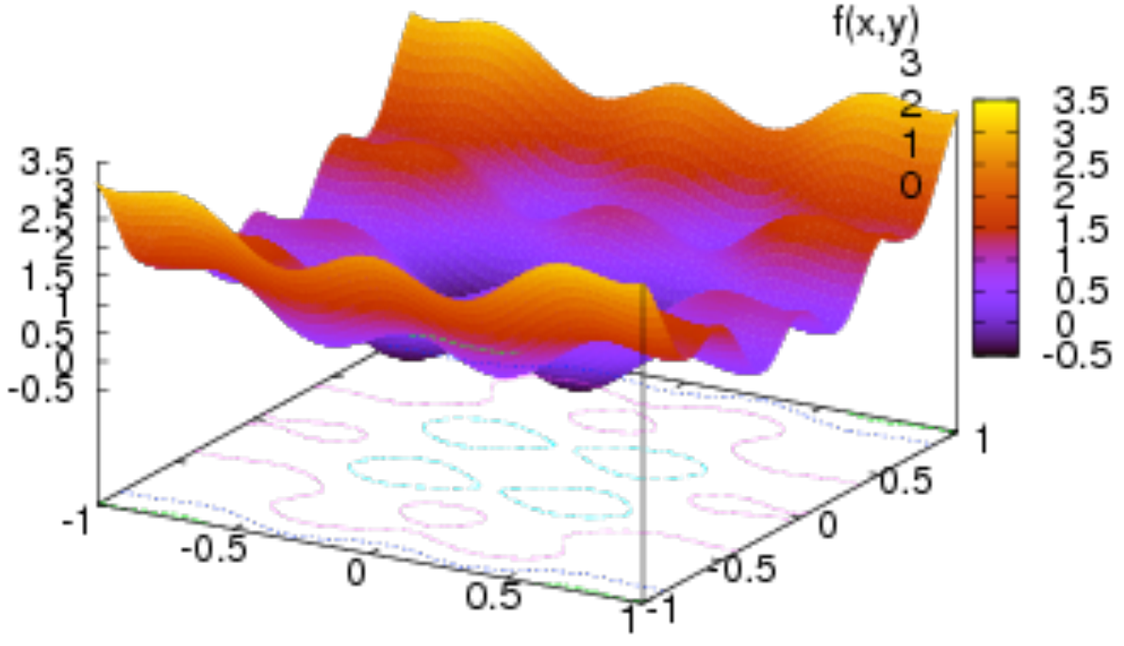
- Let's find the minima and maxima!
  - Can do some hands on now

Optimization in multiple dimensions

- Last time we talked about optimization in one dimension
- Now let's extend this!
- We took some shortcuts in 1-d :
  - There's only one derivative
  - There's only two kinds of extrema
- In N-d, these must be relaxed
  - We replace a derivative with a gradient
  - There are three kinds of extrema (max, min, saddle)

Optimization in multiple dimensions

- NR recommends using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method
  - http://en.wikipedia.org/wiki/BFGS_method
- Approximates Newton's method ("gradient descent")
- There are others, but this one is pretty robust and also quite fast



Optimization in multiple dimensions

- First : Newton's method in N-dimensions
- recall : in 1-dim:

$$x_{\rm new} = x_0 - \frac{f(x_0)}{f'(x_0)} \equiv x_0 + dx \ .$$

- or :

$$x_{\rm new} = x_0 - (f'(x_0))^{-1} f(x_0)$$

- Trivially switching to n-dim "schematically":

$$\mathbf{x}_{\rm new} = \mathbf{x}_0 - (\nabla f(\mathbf{x}_0))^{-1} f(\mathbf{x_0})$$

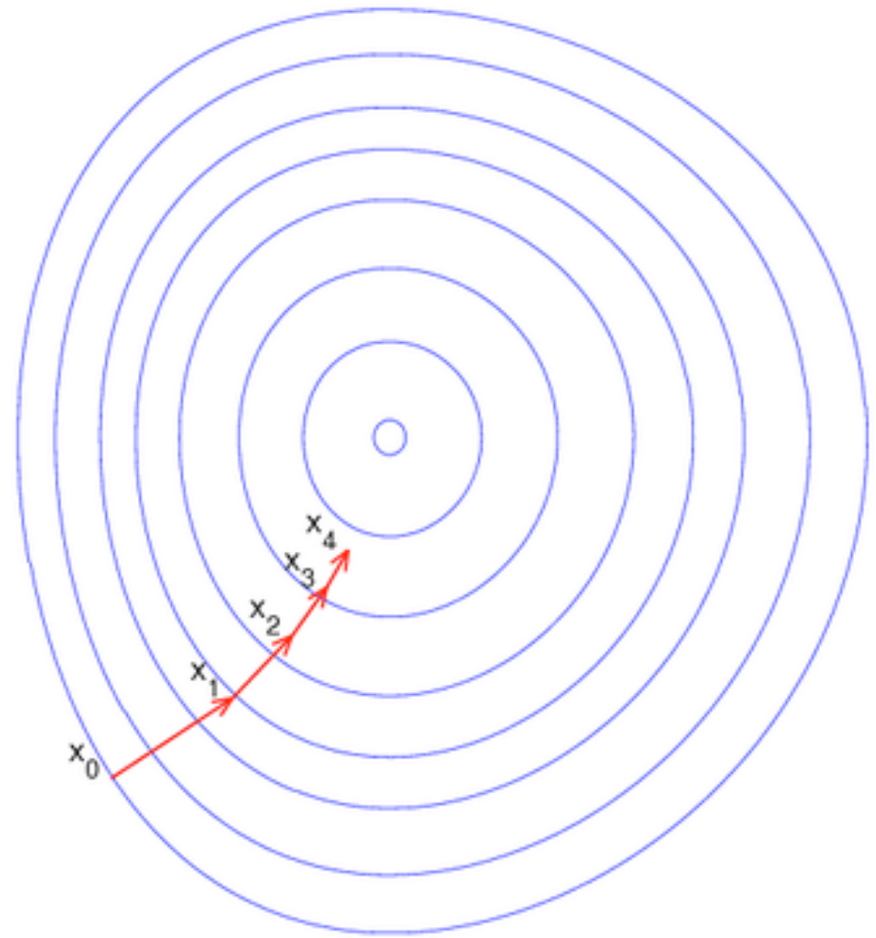
- The grad matrix is the Hessian:
  - http://en.wikipedia.org/wiki/Hessian_matrix

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

- We then have the steepest descent,
- and can linearly descend in 1-d!

Optimization in multiple dimensions

- Algorithm :
  - Compute gradient
  - Step along maximum gradient to minimum!



- But! This is intensive. We can do better because we can use an iterative approximation (H) to the Hessian matrix (A^{-1}) that's "good enough"

$$\lim_{i \to \infty} \mathbf{H}_i = \mathbf{A}^{-1}$$

Optimization in multiple dimensions

- Near the current point, to second order we have:

$$f(\mathbf{x}) = f(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla f(\mathbf{x}_i) + \tfrac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i)$$

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_i) + \mathbf{A} \cdot (\mathbf{x} - \mathbf{x}_i)$$

- In newton's method, grad-f = 0 so to get to the next point:

$$\mathbf{x} - \mathbf{x}_i = -\mathbf{A}^{-1} \cdot \nabla f(\mathbf{x}_i)$$

- Instead of the full Hessian matrix, we use an iterative approximation
- Modify the above at points i+1 and i, take the difference, and we get:

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{A}^{-1} \cdot (\nabla f_{i+1} - \nabla f_i)$$

Optimization in multiple dimensions

- If we then assumed H(i+1) were actually A^{-1}, then this would be :

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{A}^{-1} \cdot (\nabla f_{i+1} - \nabla f_i)$$

$$\mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{H}_{i+1} \cdot (\nabla f_{i+1} - \nabla f_i)$$

- Let's construct a formula of the form H(i+1) = H(i) + correction, so it would eventually converge to actual Hessian matrix
- Must satisfy the above, and be calculable from what we have "on hand".
- Candidate:

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i+1} - \mathbf{x}_i)}{(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\nabla f_{i+1} - \nabla f_i)} - \frac{[\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)] \otimes [\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)]}{(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)}$$

Outer product (a matrix)

Optimization in multiple dimensions

- Details aren't so interesting, but this does converge to the actual Hessian matrix

- An updated form of this converges with lower errors (BFGS) :

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i) \otimes (\mathbf{x}_{i+1} - \mathbf{x}_i)}{(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\nabla f_{i+1} - \nabla f_i)}$$

$$- \frac{[\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)] \otimes [\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)]}{(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)}$$

$$+ \left[(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)\right] \mathbf{u} \otimes \mathbf{u}$$

- where :

$$\mathbf{u} \equiv \frac{(\mathbf{x}_{i+1} - \mathbf{x}_i)}{(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\nabla f_{i+1} - \nabla f_i)} - \frac{\mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)}{(\nabla f_{i+1} - \nabla f_i) \cdot \mathbf{H}_i \cdot (\nabla f_{i+1} - \nabla f_i)}$$

Optimization in multiple dimensions

- So, this is the basis for our implementation
  - Implemented in C++
  - Also implemented in scipy
- Basically :
  - Input function AND gradient
  - Initialize approximate Hessian inverse (hessin) by unit matrix, and initial direction to some random value
  - while error is too big :
    - update the line direction using hessin
    - update gradient
    - compute difference in gradient, update hessin
    - calculate big complicated formula
    - continue
- Other details are not interesting

Optimization in multiple dimensions

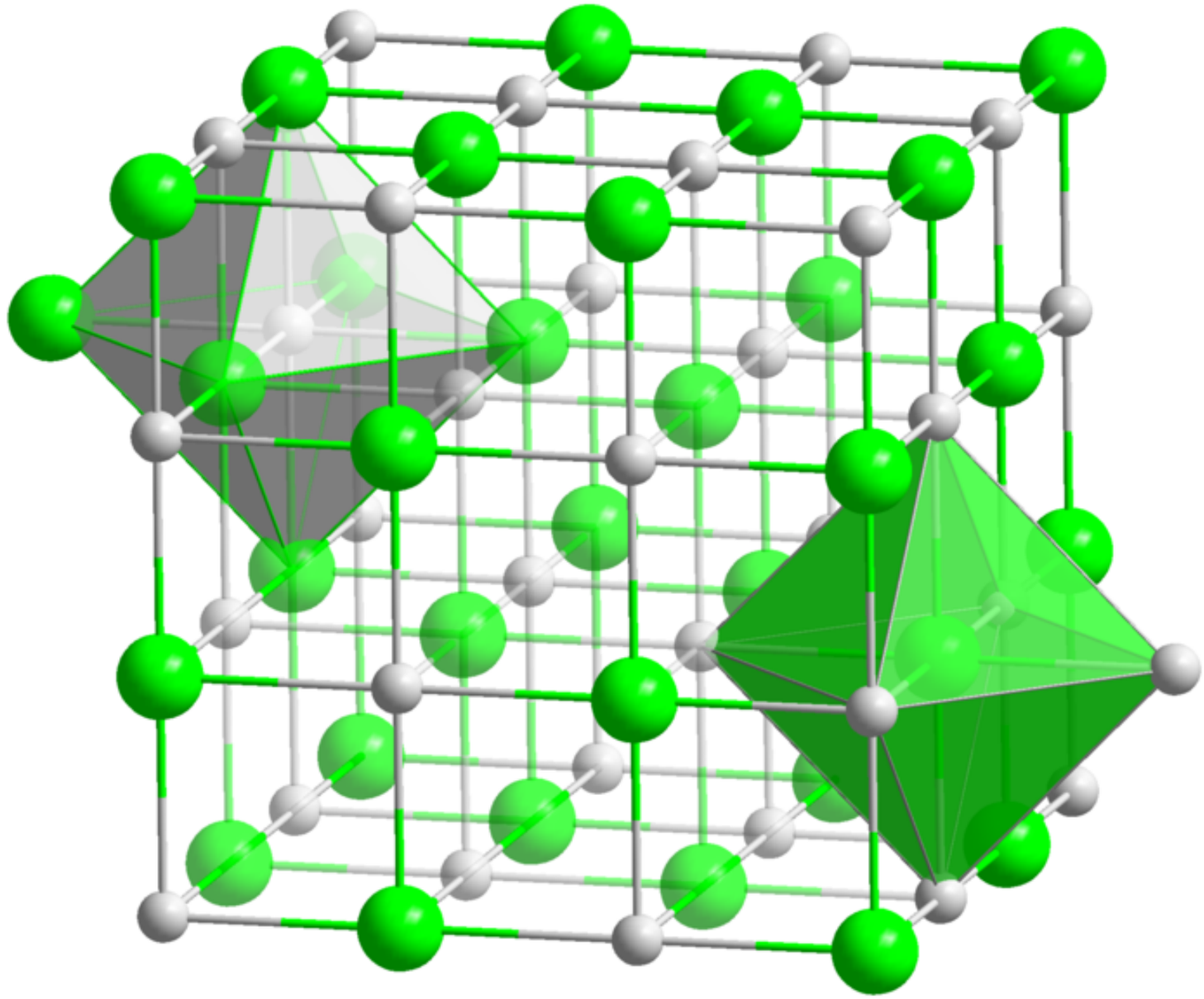
- Strategies for avoiding local minima :
  - First find “coarse” minima with stable algorithm, initialize from there, use BFGS to find minima with high precision
  - Compute an ensemble of “pseudo experiments” (or “toys”) where the initial value is randomly varied, take the ensemble mean (or median)
  - Pick “correct” initial conditions from first principles

Equilibria and modes of Molecules

- Let's consider an example of molecules in a regular format (lattice-like)

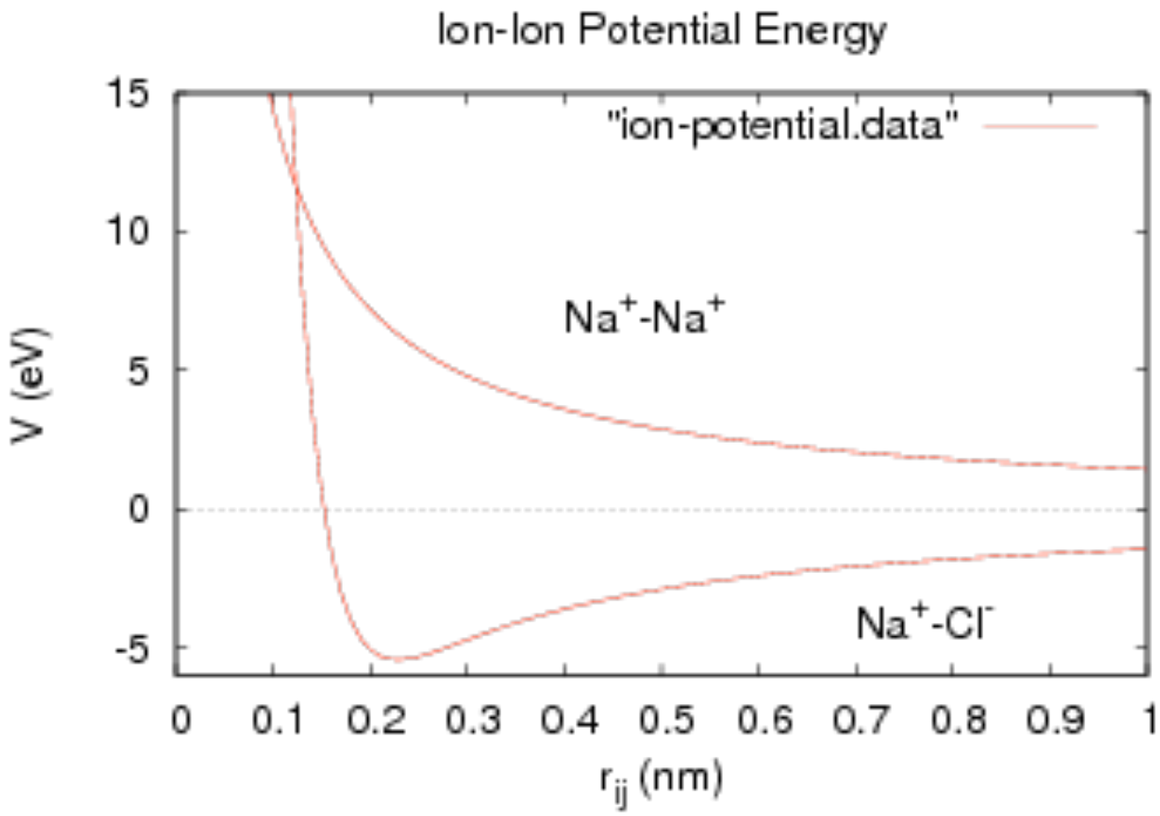
- For instance, salt (sodium chloride) http://en.wikipedia.org/wiki/Sodium_chloride

- Let's consider examples like NaCl, Na_2Cl+, Na_2Cl_2, Na_3Cl_2+



Equilibria and modes of Molecules

- Potential is :



Coulomb term

Repulsive term to account for Pauli exclusion

Numerical "trick" to stabilize things at low r (negligible at equil.)

$$V(r_{ij}) = \begin{cases} -\frac{e^2}{4\pi\epsilon_0 r_{ij}} + \alpha e^{-r_{ij}/\rho} + b\left(\frac{c}{r_{ij}}\right)^{12} & \text{for } \mathrm{Na^+}\text{–}\mathrm{Cl^-} \text{ pairs} \\ +\frac{e^2}{4\pi\epsilon_0 r_{ij}} + b\left(\frac{c}{r_{ij}}\right)^{12} & \text{for } \mathrm{Na^+}\text{–}\mathrm{Na^+} \text{ or } \mathrm{Cl^-}\text{–}\mathrm{Cl^-} \end{cases}$$

Equilibria and modes of Molecules

- Numbers involved :

$$\frac{e^2}{4\pi\epsilon_0} = 1.44 \text{ eV} - \text{nm}$$

$$\alpha = 1.09 \times 10^3 \text{eV}$$

$$\rho = 0.0321 \text{ nm}$$

$$b = 1 \text{ eV}, c = 0.01 \text{ nm}$$

Equilibria and modes of Molecules

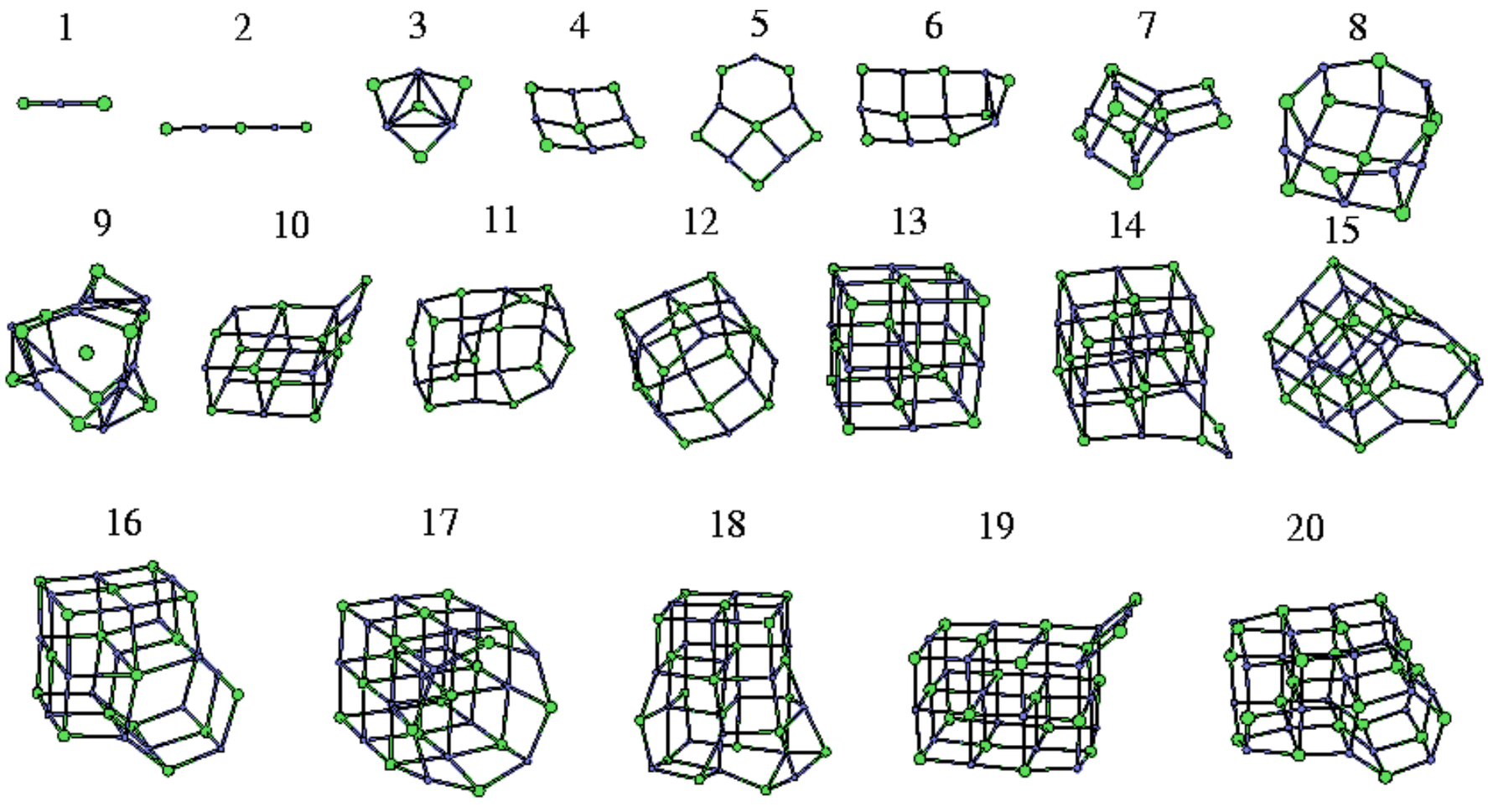
- Further reading :
  - K. Michaelian, "Evolving few-ion clusters of Na and Cl", Am. J. Phys. 66, 231 (1998), which uses a genetic algorithm to study clusters of ions.

Equilibria and modes of Molecules

- Equilibrium structures are global minima of the potential energy:

$$U(\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{n-1}) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} V(r_{ij}) \ ,$$

- A few cases are shown here :



http://www-wales.ch.cam.ac.uk/CCD.html