<u>Optimization</u>: find minimum (or maximum) of a given function $f(x)$

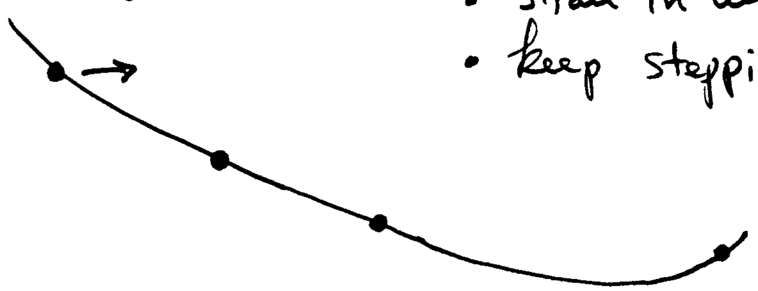
- Start with 1-D (much easier)
- Start with methods that don't use derivatives

<u>Strategy</u>
1) Narrow interval until <u>unimodal</u>.
2) Use iterative method to narrow interval, using unimodal property

---

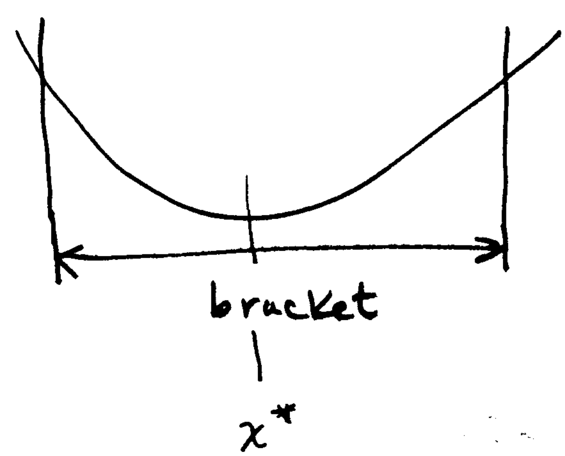
1) Bracketing a minimum:

- Start in decreasing direction
- keep stepping until function <u>increases</u>



usually steps are increased, or extrapolation is used.

Nothing sophisticated here. End up with

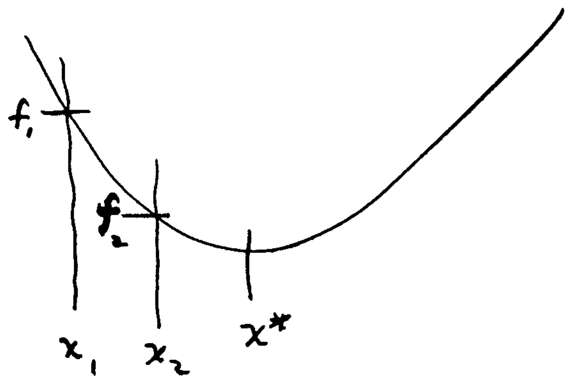


guaranteed that <u>minimum</u> exists

Mathematical Framework

Defn. a fctn. $f(x)$ is called unimodal on $[0,1]$ if there is an $x^* \in [0,1]$ such that

$$x_1 < x_2 < x^* \Rightarrow f_1 > f_2$$
$$x_2 > x_1 > x^* \Rightarrow f_1 < f_2$$



How do we choose points (experiments) to locate x^* to be within as small an interval as possible?

When Do we stop - an important point:

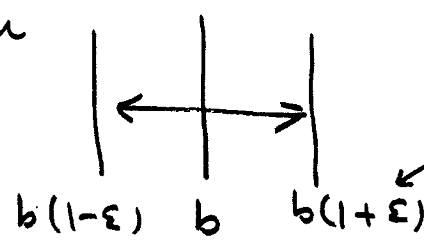
[Press et al.]

at a minimum $f'(x^*) = 0$

so at $x = b$ near x^*:

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

We want to stop when bracket reaches this:



relative accuracy limit in floating pt, typically double $\approx 10^{-15}$

$\therefore$ want $\left|\frac{1}{2}(x-b)^2 f''(b)\right| < \epsilon f(b)$

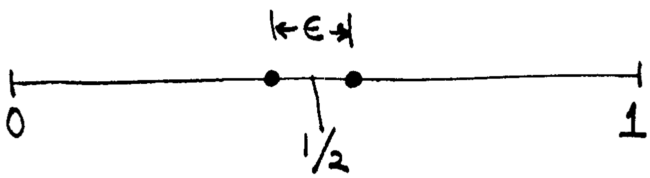
$$|x-b| < \sqrt{\epsilon}\,|b| \underbrace{\sqrt{\frac{2f(b)}{b^2 f''(b)}}}_{O(1)}$$

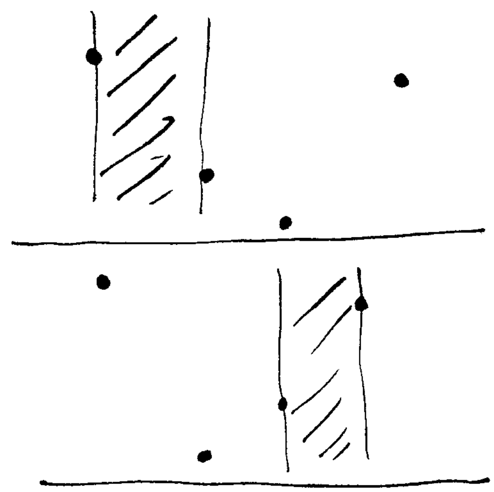
Rule of thumb, ask for fractional width about $\sqrt{\epsilon} \approx \sqrt{10^{-15}} \approx 3 \times 10^{-8}$

No reason to go further

Starting with methods that do not use derivatives —

<u>Dichotomous Search</u> ("bisection")





An interval of size $\frac{1}{2} - \frac{\epsilon}{2}$ is eliminated as possible location of min.

ϵ should be big enough to ensure that the decision

$$f\left(\tfrac{1}{2}+\tfrac{\epsilon}{2}\right) \overset{>}{\underset{?}{<}} f\left(\tfrac{1}{2}-\tfrac{\epsilon}{2}\right)$$

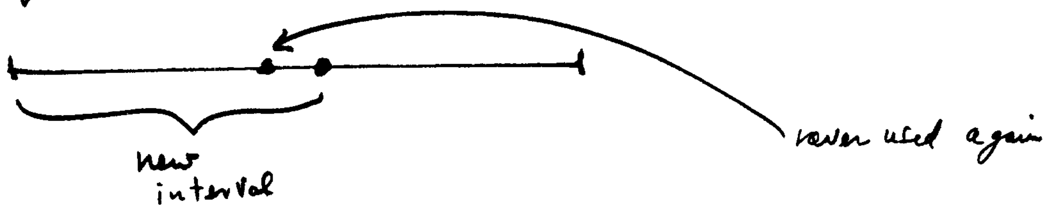
is reliable.

L_n = interval of uncertainty after n measurements $= \left(\frac{1+\epsilon}{2}\right)^{\left(\frac{n}{2}\right)}$

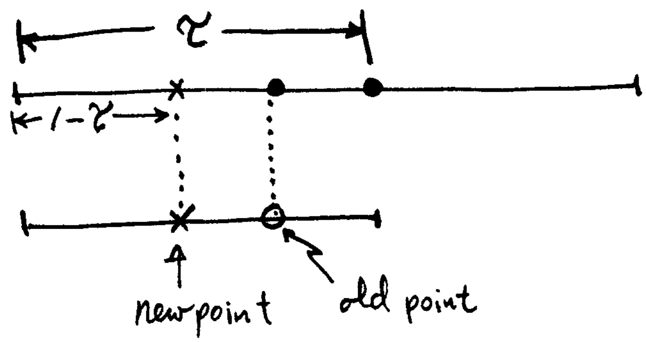
$$\approx \left(\frac{1}{\sqrt{2}}\right)^n = (.707)^n$$

Can we make this more efficient?

Notice: points are wasted



to plan ahead:



We would like the new interval to be partitioned in the same way:

$$\frac{\tau}{1} = \frac{1-\tau}{\tau}$$

$$\Rightarrow \quad \tau^2 + \tau - 1 = 0$$

$$\tau = \frac{\sqrt{5}-1}{2} = 0.6180339\ldots$$

$$L_n \approx (.618)^n$$

[cf $(.707)^n$ for Dichotomous →

Suppose we want to reach δ,

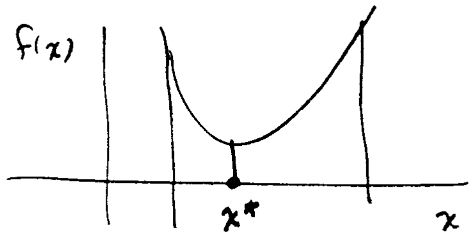
so $\quad (.707)^{n_1} = (.618)^{n_2}$

... about 40% more iterations] $\quad \dfrac{n_1}{n_2} = \dfrac{\ln(.618)}{\ln(.707)} \doteq 1.39$

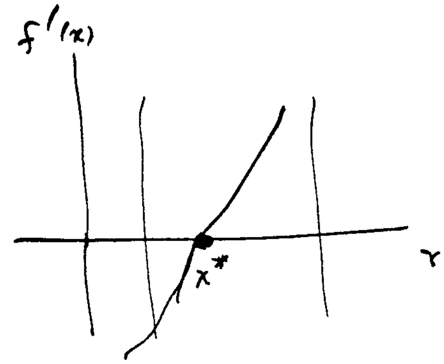
→ Works without regard to smoothness – as long as unimodal.

→ Linear convergence.

Newton – Raphson



apply to derivative



$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$ ← uses second derivative

Convergence rate known:

$$L_n \approx \underbrace{(\text{const.})}_{\left|\frac{f'''(\theta_n)}{2f''(\theta_n)}\right|} L_{n-1}^2 \Rightarrow L_n \sim \epsilon^{2^n}$$

quadratic convergence

If this runs into problems finding zeros – the problems in finding zeros of derivatives are even **more dangerous**!

Very fast and very unreliable.

How to get faster than linear convergence?

there are in fact many methods that attempt to improve on the rock-solid linear convergence of dichotomous or golden-section search with bracket — and they are all

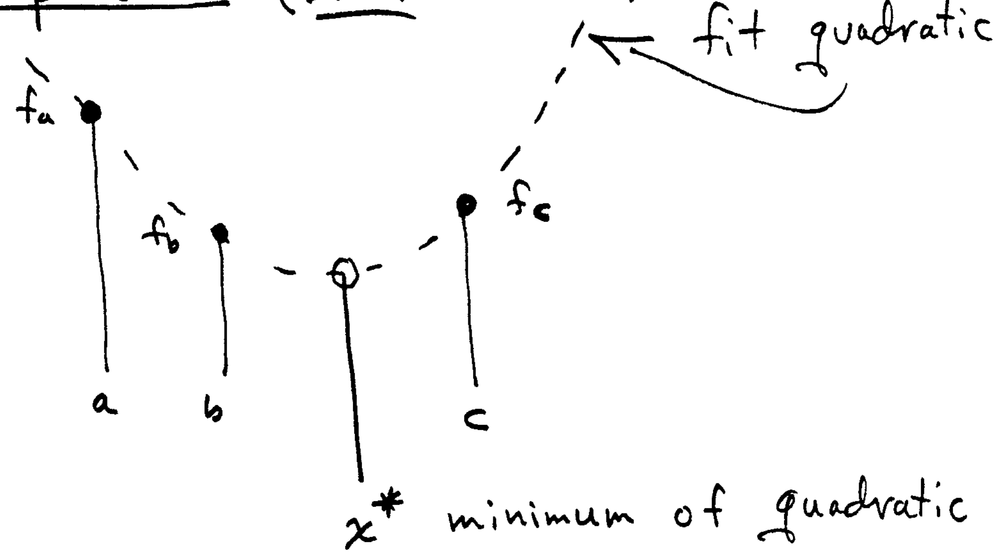
faster and more dangerous.

the general strategy is to combine

$$\boxed{\begin{array}{l}\text{golden-section bracketing} \\ \text{\& a faster method at final convergence}\end{array}}$$

See [Press et al.] for details of combination strategy. they combine with

Parabolic Interpolation (for Brent's Method)

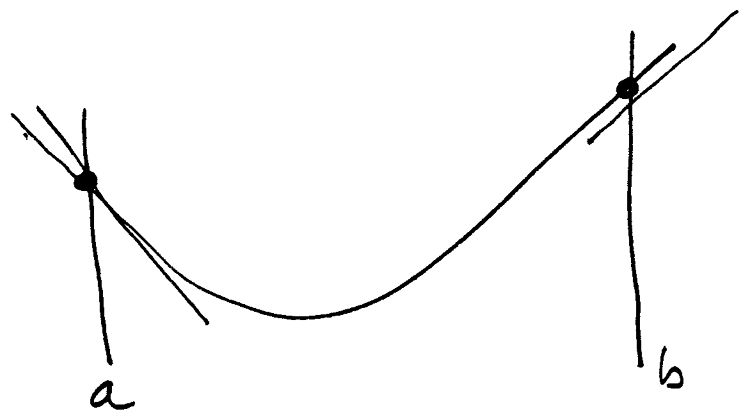


See [Press et al.] for formulas.

→ uses only function values

Moral combine linearly convergent method with bracket, with super-linearly convergent method.

BTW, here's a very fast & dangerous method that I like - goes back to Davidon & 60's-70's; uses first derivatives:



use $f(a)$, $f'(a)$, $f(b)$, $f'(b)$ & interpolate unique cubic. Algebra is easy and requires solving a quadratic for min of cubic polynomial.

Can also be combined with bracketing that is safe.

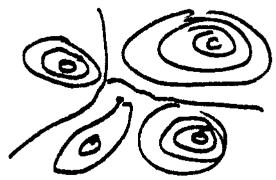
on to higher dimensions...

Higher Dimensional Optimization

→ Very important in many areas
- parameter estimation in statistics
- design of many many systems where a clear criterion of goodness must be maximized

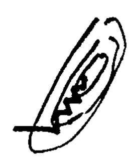
→ Very difficult (Bracketing is not possible!)

for example, even in 2 dimensions, we can have the following kinds of difficulties

1) multimodal  can have many different minima

2) saddle points max in one dimension, min in other 

3) Bad scaling  can be very compressed in one dimension – making search hard

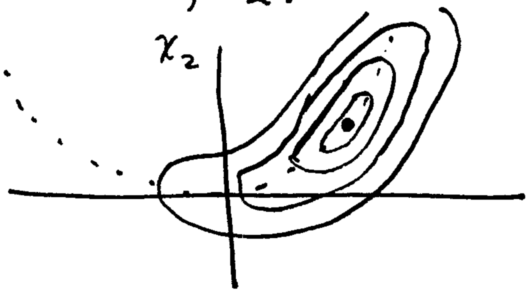
4) interaction of coordinates  can't change one variable at a time

5) strange shapes
   a) ridges
   b) curved valleys

Some Historically important Horror Stories:
(tests)

Rosenbrock's curved Valley [1960]

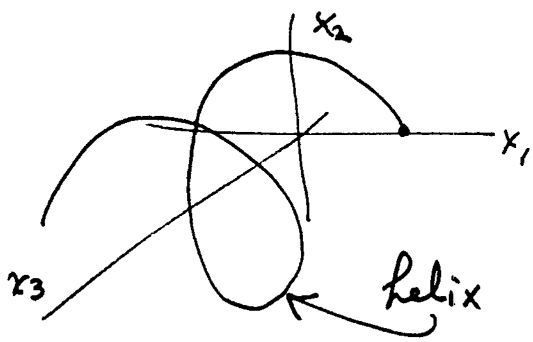
$$U(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



Fletcher & Powell's Helical Valley [1963]

$$f(x_1, x_2, x_3) = 100\left\{ [x_3 - 10\,\theta(x_1, x_2)]^2 \right\} + \left\{ [r(x_1, x_2) - 1]^2 \right\}$$

where
$$\begin{cases} 2\pi\theta(x_1, x_2) = \arctan(x_2/x_1) & x_1 > 0 \\ \qquad\qquad\quad\;\; \pi + \arctan(x_2/x_1) & x_1 < 0 \\ r(x_1, x_2) = (x_1^2 + x_2^2)^{1/2} \end{cases}$$



Methods without derivatives are too slow for hard problems —

but see Nelder & Mead's method (1965) described in [Press et al.]

in 2-d: a crawling creeping triangle that expands, contracts, reflects, etc.

Most Practical Methods are based on gradients 2.3.10

$$\nabla f = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \end{bmatrix}$$

Points in direction of maximally increasing $f(\underline{x})$

$\underline{x}$ is n-dim. Vector

1) Simplest (Bad)

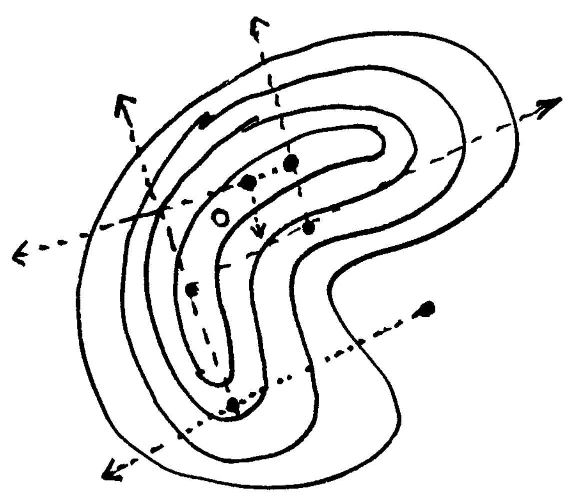
<u>Steepest</u> <u>Descent</u>

More along gradient – usually we try a 1-d minimization in this direction

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \nabla f(\underline{x}_i)$$

adjust to find min

Can be miserably slow in curved valley

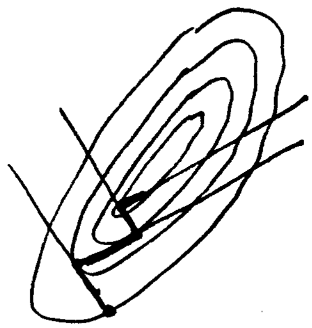


2.3.11

Another major loser

2) Successive Relaxation

optimize in one coordinate at a time



zig-zag

Another loser, except in very smooth cases

3) Newton-Raphson in n-dimensions

$$\underline{x}_{i+1} = \underline{x}_i + \underline{\underline{H}}^{-1} \nabla f(\underline{x}_i)$$

matrix of second derivatives

Each step is expensive. Usually also used with 1-dim. search in direction

State-of-the-Art methods try to mimic the convergence of Newton-Raphson with first derivatives only.

<u>Strategy</u> Find sets of directions that are "conjugate"

- minimizing in these directions is exact for really quadratic fctn, & convergence quadratic like Newton-Raphson.

See [Press et al.] & references there.

We haven't even touched <u>constraints</u>, <u>linear programming</u>.