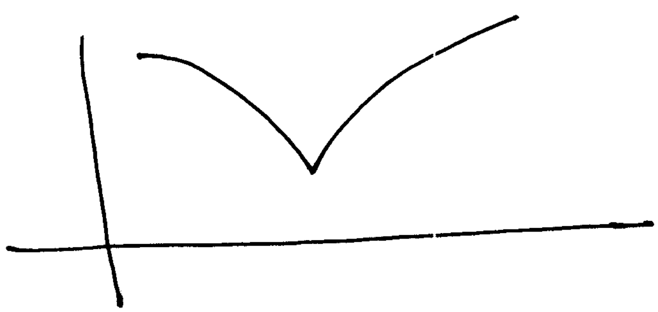
One-Dimensional Minimization

Seeing that the naive approach doesn't work well, we'd continue to study the problem.

We start with a function with a single local minimum (these are called unimodal):

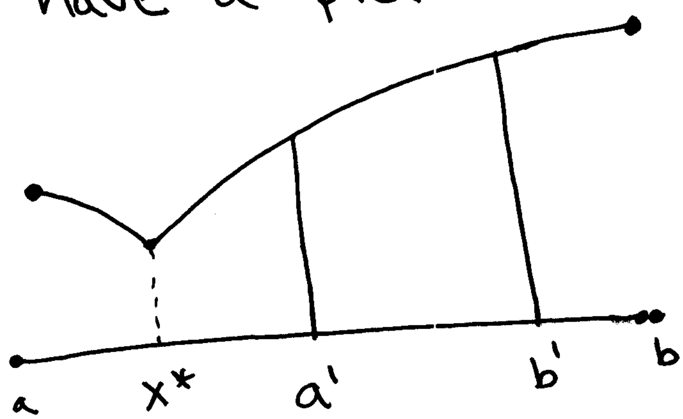


Lemma. A unimodal function is strictly decreasing up to the local min and strictly increasing afterward.

Proof. Easy, left to student.

Question. Given $[a, b]$ and f, how accurately can we compute ~~a~~ ~~to~~ the min of a univariate function ~~knowing~~ with n function evaluations?

Example. Given two evaluations at a', b' we have a picture like



Now if $f(a') \leq f(b')$, we know only one thing: the min is not between b' and b. If $f(a') > f(b')$, the min is not between a and a'.

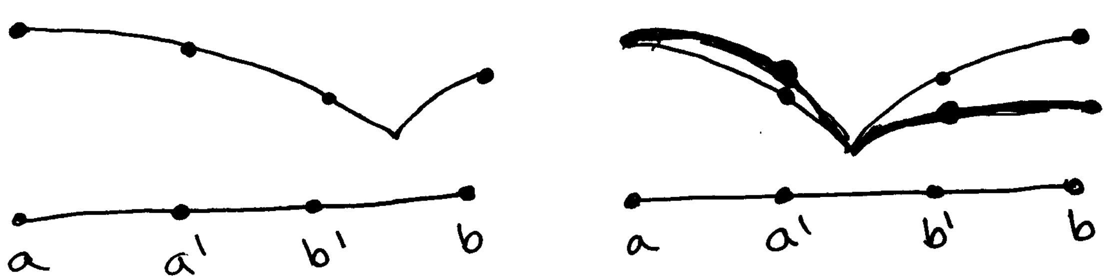
(Technically, if $f(a) = f(b)$ we know the min is between a' and b', but that case is essentially impossible in floating point arithmetic for a nontrivial function.)

Thus a' and b' should be ~~evaluated~~ as close to $\frac{a+b}{2}$ as roundoff error allows, for error at most $\frac{1}{4}(b-a)$.

Example 2. Suppose we have three evaluations of f. If we choose

$$a' = \frac{1}{3}(b-a) + a \qquad b' = \frac{2}{3}(b-a) + a$$

Suppose $f(a') \geq f(b')$. We have either



The difference between the two cases is the derivative at b'. We discover this with another evaluation near b'. This gives error bound $\frac{1}{6}(b-a)$.

How can we generalize? Using the somewhat complicated

Fibonacci search algorithm.

Let λ_n be the (n)th member of the Fibonacci sequence. We consider a sequence of intervals indexed by $k \in n, n-1, \ldots, 3$. If $I_k = [a,b]$, then let

$$\Delta = \left(\frac{\lambda_{k-2}}{\lambda_k}\right)(b-a)$$

$$a' = a + \Delta$$

$$b' = b + \Delta$$

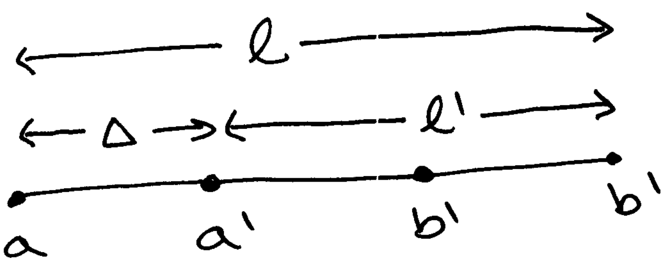
We have

$$I_{K-1} = \begin{cases} [a', b] & \text{if } f(a') \geq f(b') \\ [a, b'] & \text{if } f(a') \leq f(b') \end{cases}$$

For the step $K=2$, we set $a' = \frac{1}{2}(a+b) - \delta$, $b' = \frac{1}{2}(a+b) + \delta$ as in example 1.

Proof that FSA works.

At step K, assume we have located the min within $[a,b]$. We have



Now

$$\ell' = \ell - \Delta = \ell - \left(\frac{\lambda_{K-2}}{\lambda_K}\right)\ell$$

$$= \left(\frac{\lambda_K - \lambda_{K-2}}{\lambda_K}\right)\ell = \left(\frac{\lambda_{K-1}}{\lambda_K}\right)\ell.$$

Since $[a, b']$ and $[a', b]$ have the same length, we know that by our previous argument, we are picking the new interval so that it contains the min and so

$$\text{Length}(I_{k-1}) = \left(\frac{\lambda_{k-1}}{\lambda_k}\right) \text{Length}(I_k).$$

Now we have a new interval, and will step into it by

$$\Delta' = \left(\frac{\lambda_{k-3}}{\lambda_{k-1}}\right) \ell' = \left(\frac{\lambda_{k-3}}{\lambda_{k-1}}\right)\left(\frac{\lambda_{k-1}}{\lambda_k}\right) \ell$$

$$= \left(\frac{\lambda_{k-3}}{\lambda_k}\right) \ell.$$

We claim that this Δ' is exactly the distance between a' and b', and so we critically! can reuse one of our previous function evaluations.

We compute

$$
\begin{aligned}
b' - a' &= \ell - 2\Delta \\
&= \ell - 2\left(\frac{\lambda_{k-2}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_k - 2\lambda_{k-2}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_{k-1} + \cancel{\lambda_{k-2}} - \cancel{2}\lambda_{k-2}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_{k-1} - \lambda_{k-2}}{\lambda_k}\right)\ell \\
&= \left(\frac{\cancel{\lambda_{k-2}} + \lambda_{k-3} - \cancel{\lambda_{k-2}}}{\lambda_k}\right)\ell \\
&= \left(\frac{\lambda_{k-3}}{\lambda_k}\right)\ell = \Delta'.
\end{aligned}
$$

Now if we try to compute the width of the ~~final~~ semifinal $(k=3)$ interval, we get

$$I_3 = \left(\frac{\lambda_3}{\cancel{\lambda_4}}\right)\left(\frac{\cancel{\lambda_4}}{\lambda_5}\right)\cdots\cdots\left(\frac{\lambda_{n-1}}{\lambda_n}\right)(b-a).$$

$$= \left(\frac{2}{\lambda_n}\right)(b-a)$$

The last step divides this error interval by 4 to get an error bound of

$$E_n = \frac{1}{2}\left(\frac{b-a}{\lambda_n}\right). \qquad \square$$

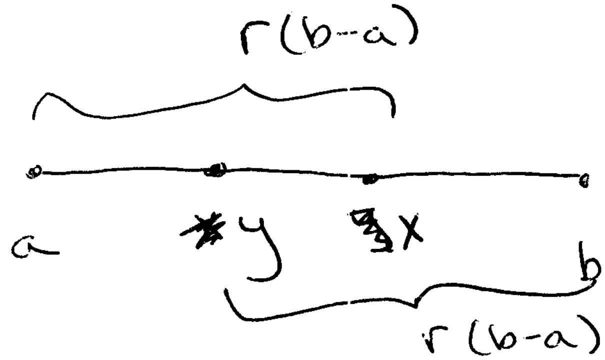
This algorithm is somewhat intricate, but has reasonable speed of convergence, since

$$\lambda_n \approx \varphi^n/\sqrt{5} = \frac{(1.618\ldots)^n}{\sqrt{5}}$$

We can get a simpler algorithm (at a slight cost in performance) by using φ directly...

One dimensional minimization.

Golden section search.



$$x = a + r(b-a)$$
$$y = a + r^2(b-a)$$

$$r = 1/\varphi$$
$$r^2 + r = 1.$$

We want x or y to be one of these points in the next subinterval.

Convergence: interval decreases by a factor of r, each time.

comparison to fibonacci?

$$r^{n-1} / \lambda_{n+1}^{-1} = 1/\varphi^{n-1} \; \varphi^{n+1}/\sqrt{5} = \varphi^2/\sqrt{5} \approx 1.17$$

~~Quadratic inte~~

How slow? For one digit, we want

$$\frac{1}{10} = r_1^{n}$$ and it turns out that $r = .5$

So it's about 5 steps per digit.

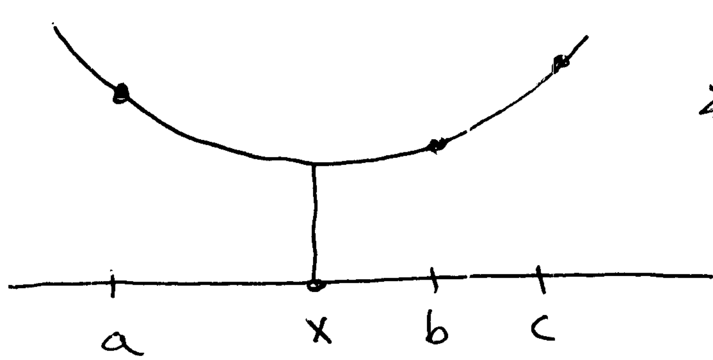
What if your fn is smoother?

$$F(x) = F(x^*) + (x-x^*)F'(x^*) + \frac{1}{2}(x-x^*)^2 F''(x^*)$$

and since x^* is the min,

$$F(x) \approx F(x^*) + \frac{1}{2}(x-x^*)^2 F''(x)$$

So we use quadratic interpolation!



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

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

$$\bar{a} = \frac{f(b)-f(a)}{b-a}$$

$$\bar{b} = \frac{f(c)-f(b)}{c-b} \qquad x = \frac{1}{2}\left[a+b-\frac{\bar{a}}{\bar{c}}\right]$$

$$\bar{c} = \frac{\bar{b}-\bar{a}}{c-a} \qquad q''(t) = 2\bar{c}.$$

derive with mathematica

---

cases: too far, actually a max.

Accuracy concerns: when $(x-x^*) < \epsilon$

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

we have

$$F(x) \approx f(x^*) + \frac{1}{\cancel{2}}\,\epsilon\,\cancel{(x^*)^2}\left(\frac{\cancel{2}|f(x^*)|}{\cancel{(x^*)^2}\,\cancel{f''(x^*)}}\right) \cancel{f''(x^*)}$$

$$\approx (1+\epsilon)\, f(x^*).$$