Root Finding

Mech 510

Fall, 2018

The Problems

We need to find one (or more) zeroes of:

1. A function of one variable, for which we can evaluate the function but not its derivative.
2. A function of one variable, for which we can evaluate the function and its derivative.
3. A function of more than one variable, for which we can evaluate the function and its derivative.

The Simplest Thing That Can Possibly Work: Bisection

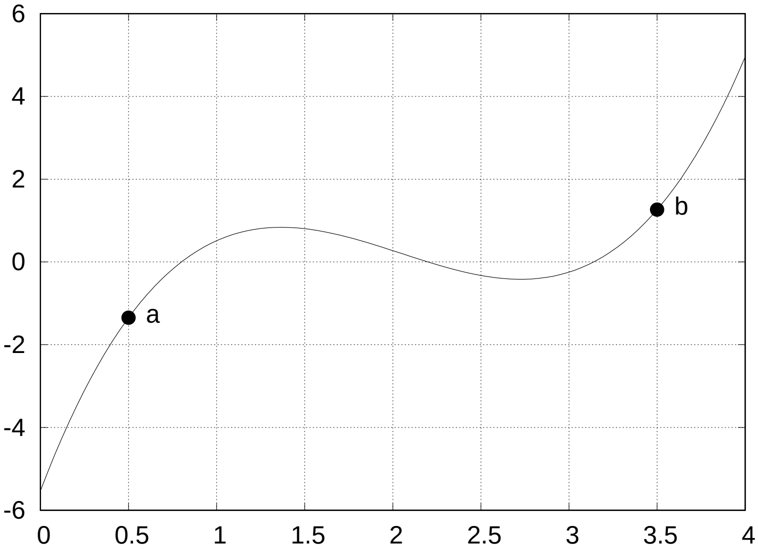
Given:

- A function $y = f(x)$.
- Two values a and b such that bracket a root: $f(a) \cdot f(b) < 0$.
- A convergence tolerance δ.

Bisect the range (a, b), keeping the root bracketed. Guaranteed to work, even for pathological functions.

```
evaluate and store f(a), f(b)
do
  c = (a+b)/2
  evaluate f(c)
  if f(a)*f(c) < 0
    b←c
  else
    a←c
until |a-b| < δ
```

Example



Approximating Slope: The Secant Method

Given: Same inputs.

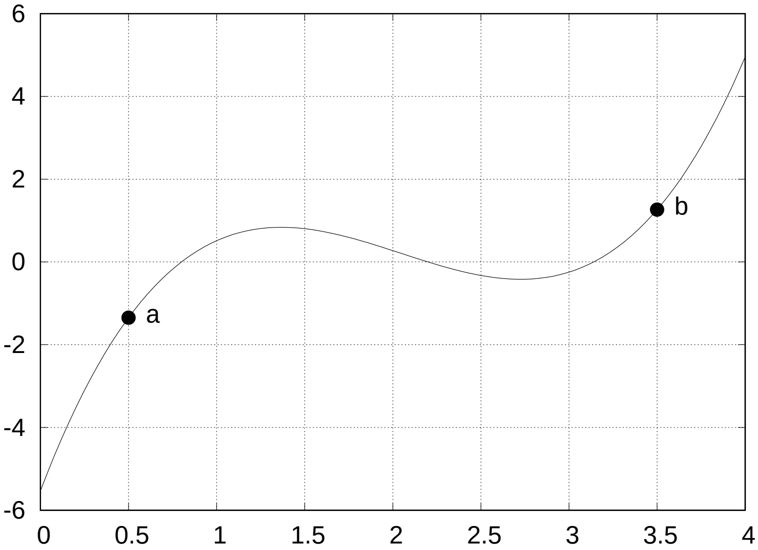
At each step, replace older point with linear interpolation / extrapolation to zero:

$$\begin{aligned} \text{slope} &= \frac{f(b) - f(a)}{b - a} \\ c &= b - \frac{f(b)}{\text{slope}} \\ &= b - (b - a)\frac{f(b)}{f(b) - f(a)} \end{aligned}$$

```
evaluate and store f(a), f(b)
do
  find c as above
  a←b
  b←c
until |a-b|<δ
```

Faster when it works but doesn't bracket the root. Can go crazy!

Example



Approximating Slope: The Regula Falsi Method

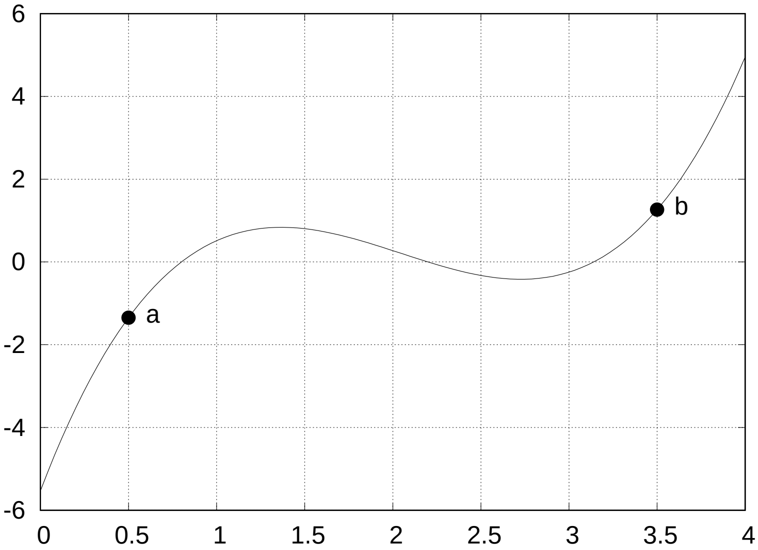
Given: Same inputs.

At each step, replace one point with linear interpolation / extrapolation to zero, but keep the root bracketed.

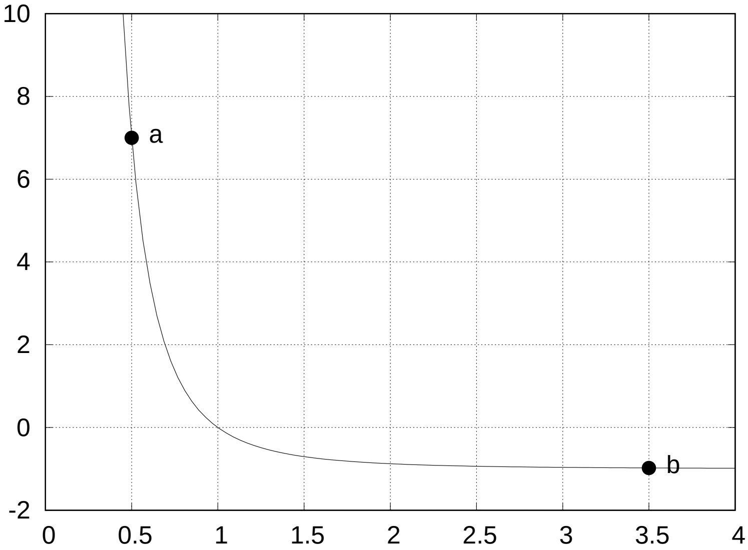
```
evaluate and store f(a), f(b)
do
  find c as for secant method
  evaluate f(c)
  if f(a)*f(c) < 0
    b←c
  else
    a←b
until |a-b|<δ
```

Slower than secant method but always brackets the root. Need to be careful about repeatedly keeping the same end point value!

Example



Pathological Example 1



Ridders' Method

The Idea: $f(x)$ is non-linear. So let's work instead with another function

$$h(x) = f(x)\, e^{Ax}$$

that interpolates $f(a)$, $f(b)$, and $f\left(\frac{a+b}{2}\right)$.

Why this helps: $h(x)$ has the same roots as $f(x)$. But regula falsi works better on $h(x)$ because it's closer to linear.

New point location:

$$c = \frac{a+b}{2}$$

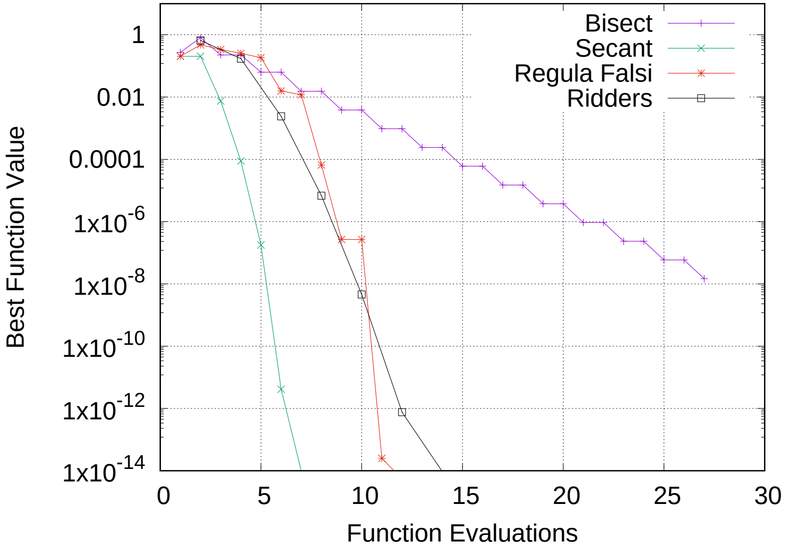
$$d = c + (c-a)\frac{\operatorname{sign}(f(a))\, f(c)}{\sqrt{f^2(c) - f(a)\, f(b)}}$$

Update: If $f(c)$ and $f(d)$ have opposite signs, keep c and d. Otherwise, keep d and one of a and b (always bracket!).

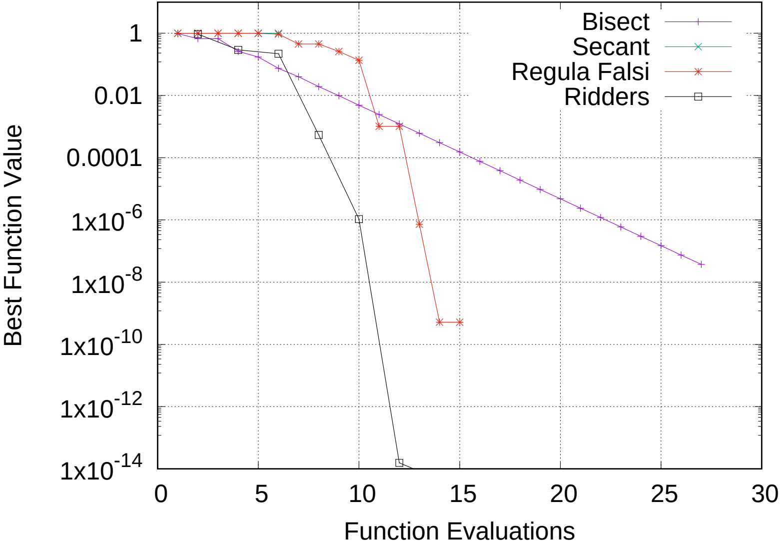
How good is this?

- Doubles the number of significant digits every iteration
- Requires two function evaluations per iteration

A Quantitative Comparison: Cubic Function



A Quantitative Comparison: A Pathological Function



Newton's Method

- Requires both the function and its derivative
- Assumes linear behavior of the function near current point:

$$x \leftarrow x - \frac{f(x)}{f'(x)}$$

- Works great when it works
- Like secant method, it can shoot off to ∞, so you need a good starting guess (or some way to globalize it).

Root Finding in Multiple Dimensions

- Suppose you have a system of non-linear equations to solve.
- Root bracketing methods fail: how can you be sure you've got a root in a region, in general?
- Which brings us to something like Newton's method:

$$\mathbf{f}(\mathbf{x}) = 0$$

The Newton update looks like:

$$\mathbf{x} \leftarrow \mathbf{x} - J^{-1}\mathbf{f}(\mathbf{x})$$

where

$$J = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$$

An Example of Newton's Method in 2D

- For definiteness, let's consider:

$$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} x^2 + y^2 - 4 \\ xy - \frac{1}{2} \end{pmatrix}$$

- Then the Jacobian is

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x & 2y \\ y & x \end{pmatrix}$$

and

$$J^{-1} = \frac{1}{2\left(x^2 - y^2\right)} \begin{pmatrix} x & -2y \\ -y & 2x \end{pmatrix}$$

- Let's choose a starting point and have a go:

Step	x	y	f_1	f_2	J_{11}	J_{12}	J_{21}	J_{22}	Δx	Δy
1	1	2	1	$\frac{3}{2}$	2	4	2	1	$-\frac{5}{6}$	$\frac{1}{6}$
2	$\frac{1}{6}$	$\frac{13}{6}$	$\frac{13}{18}$	$-\frac{5}{36}$	$\frac{1}{3}$	$\frac{13}{3}$	$\frac{13}{6}$	$\frac{1}{6}$	$\frac{13}{168}$	$-\frac{29}{168}$
3	$\frac{41}{168}$	$\frac{335}{168}$	0.0358	-0.0134	0.488	3.988	1.994	0.244	0.00792	-0.00994
4	0.25196	1.9841	1.615_{-4}	-7.869_{-5}	0.504	3.968	1.984	0.252	4.556_{-5}	-4.648_{-5}
5	0.25201	1.9841	4.237_{-9}	-2.118_{-9}	0.504	3.968	1.984	0.252	1.223_{-9}	-1.223_{-9}

Making Newton's Method More Robust: Line Search

- Our root finding problem is equivalent to finding the global minimum of
$$g(x) = \frac{1}{2}\mathbf{f}\cdot\mathbf{f}$$
- We know we started out right: along a steepest descent direction:
$$\begin{aligned}\nabla g &= \mathbf{f}^T J \\ \delta\mathbf{x} &= -J^{-1}\mathbf{f} \\ \nabla g \cdot \delta\mathbf{x} &= -\left(\mathbf{f}^T J\right)\left(J^{-1}\mathbf{f}\right) = -\mathbf{f}\cdot\mathbf{f} < 0\end{aligned}$$
So the function started back uphill on us at some point.
- The goal is to find a point where there is sufficient decrease in g:
$$g(\mathbf{x}_{\mathrm{new}}) \leq g(\mathbf{x}_{\mathrm{old}}) + \alpha \nabla g \cdot (\mathbf{x}_{\mathrm{new}} - \mathbf{x}_{\mathrm{old}})$$
α can be shockingly small: 10^{-4} is the number typically used.
- So we compute
$$\mathbf{x}_{\mathrm{new}} = \mathbf{x}_{\mathrm{old}} + \omega\,\delta\mathbf{x}$$
starting with $\omega = 1$. If we don't satisfy sufficient decrease, we reduce ω by a factor of 2 and try again. Eventually, we succeed.

References

- Wikipedia has good summaries of the methods described here:
  - Bisection Method
    `https://en.wikipedia.org/wiki/Bisection_method`
  - Regula Falsi Method
    `https://en.wikipedia.org/wiki/False_position_method`
  - Secant Method `https://en.wikipedia.org/wiki/Secant_method`
  - Ridders' Method:
    `https://en.wikipedia.org/wiki/Ridders%27_method`
  - Brent's Method is *the* state of the art in non-gradient based root finding methods, but it's complicated:
    `https://en.wikipedia.org/wiki/Brent%27s_method`
  - Newton's Method:
    `https://en.wikipedia.org/wiki/Newton%27s_method`
- *Numerical Recipes* has a good chapter on root finding.