# Outline

# Root finding

We want to solve the equation $f(x) = 0$
Functional form of $f(x)$ may not be known!
$\rightarrow$ Eg boundary value problem in ODE

Can be a hard problem:

- Most robust methods in 1 dim use bracketing:
  find $x_1, x_2 : f(x_1) < 0 < f(x_2)$
- May be difficult if $f(x) < 0$ only in very small region
- Double roots cannot be bracketed
- Rapidly oscillating functions are troublesome
- Robust and efficient algorithms exist in 1 dim
  — very different in many!

# Bracketing

All robust root-finding methods depend on bracketing:

Find $a, b : f(a) < 0 < f(b)$
Note: $a < b$ or $a > b$

If $f(x)$ does not have singularities, it must have a root in $\langle a, b \rangle$ (or $\langle b, a \rangle$)
There is no hard-and-fast way of bracketing, or even knowing that there is a root!

3 main 'methods'

1. make a (crude) plot
2. geometric expansion around initial range
3. subdivide interval until root is bracketed on at least one subinterval

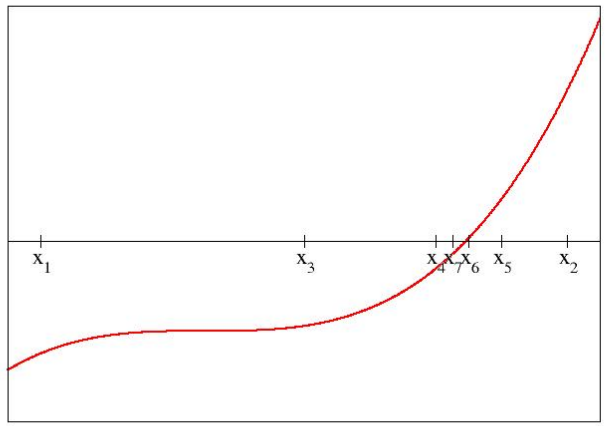
# Bisection

Bisection always works! — but can be slow

## Algorithm

1. Bracket the root: find $(x_1, x_2) : y_1 \equiv f(x_1) < 0, y_2 \equiv f(x_2) > 0$
2. Find midpoint: $z = (x_1 + x_2)/2$
3. If $f(z) < 0$ then set $x_1 = z$ else set $x_2 = z$
4. Repeat 2, 3 until $y = 0$ to desired precision (in $y$)
   or $|x_1 - x_2| <$ desired precision (in $x$)

Remember to check that the solution is actually a root!



# False position and secant

Bisection doesnt care how close we are to the root; requires a fixed number of iterations no matter what the function looks like.

## Try to be smarter

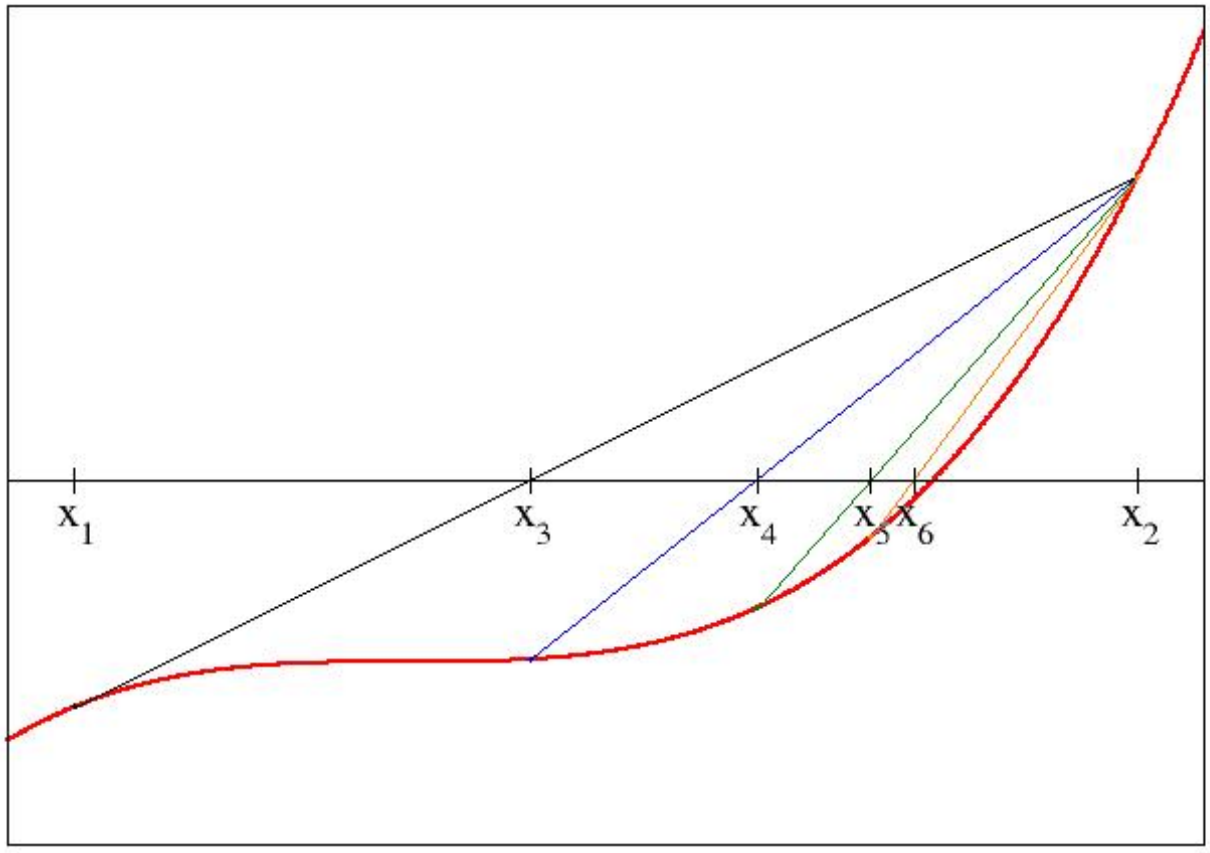
- Take a straight line from $(x_1, y_1)$ to $(x_2, y_2)$
- Find intersection with $y = 0$, use this as next guess:

$$z = x_2 - \frac{y_2}{y_2 - y_1}(x_2 - x_1)$$

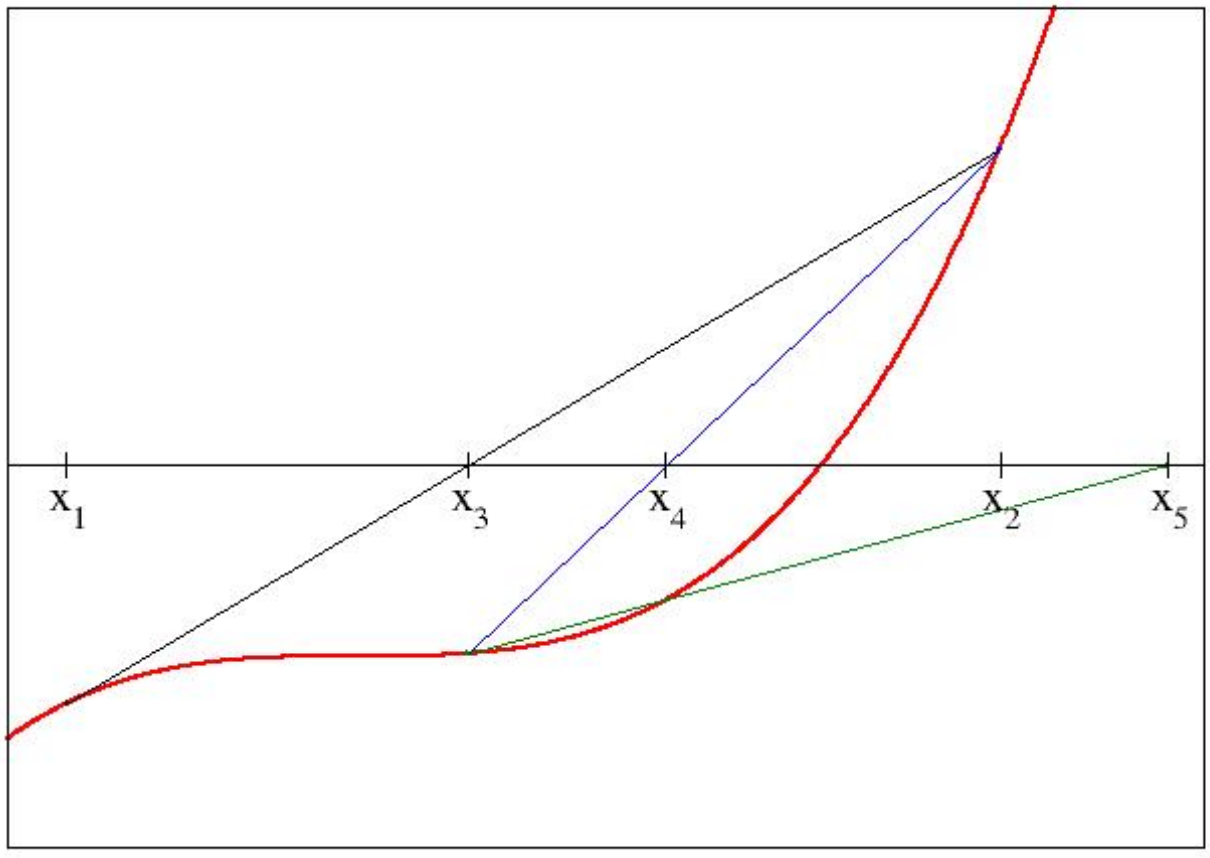
- Secant: Use $(x_2, z)$ as new bracket, ie throw away oldest $x$-value
- False position: Ensure new $(x_1, x_2)$ always bracket root

Secant converges faster near root but may fly off to $\infty$!
Both may be slower than bisection if function is "ill-behaved".

# False position



# Secant



# Ridders' method

Improve on false position by factoring out (exponential) curvature

- Take $x_1, x_2$ + midpoint $x_3 = (x_1 + x_2)/2$
- Find factor $e^Q$ : $f(x_1) - 2f(x_3)e^Q + f(x_2)e^{2Q} = 0$
  $\Leftrightarrow$ $e^{Qx_i} f(x_i)$ lies on straight line
- Use false position on $e^{Qx_i} f(x_i)$

**Result**

$$z = x_3 + (x_3 - x_1)\frac{\mathrm{sgn}(y_1 - y_2)y_3}{\sqrt{y_3^2 - y_1 y_2}}$$

Very robust, nearly always much faster than bisection

Almost as good as far more complicated, advanced algorithms

[See Numerical Recipes]

# Convergence rates

What does it mean that one method is faster than another?

Assume that at step $n$ the root is known to be within an interval of size $\epsilon_n$.
When $\epsilon_n < \delta$ (our chosen precision) we have converged to the root.
How fast can we reduce $\epsilon_n$?

- Bisection: $\epsilon_{n+1} = \epsilon_n/2$
- Secant: $\epsilon_{n+1} \approx c\epsilon_n^{1.618}$
- False position: No closed expression
- Ridders: $\epsilon_{n+1} \approx c\epsilon_n^2$ but 2 function evaluations per iteration

## Linear and superlinear convergence

| | |
|---|---|
| Linear convergence: | $\epsilon_{n+1} = c\epsilon_n$ |
| Superlinear convergence: | $\epsilon_{n+1} = c\epsilon_n^m, \quad m > 1$ |

Bisection is linear, secant is superlinear.
False position and Ridders are usually superlinear.

# Newton–Raphson

Use Taylor expansion to estimate the root

- Very good if you are close enough
- Most useful if you have analytical expression for $f'(x)$
- Much less useful in 1-d if you must use numerical derivative: Secant is better, Ridders is much better!
- Almost only general method on the market in n-dim
- Can go horribly wrong!

**Simple principle:**

$$f(x+\varepsilon) = f(x) + \varepsilon f'(x) + \frac{\varepsilon^2}{2} f''(x) + \dots$$

Ignore higher order terms, find $\varepsilon : f(x+\varepsilon) = 0$

$$\Longrightarrow \quad \varepsilon = -f(x)/f'(x)$$

# Newton–Raphson in more dimensions

We have $N$ equations for $N$ variables:

$$f_i(x_1, \ldots, x_N) = 0\,, \quad i = 1, \ldots, N$$

Taylor expansion:

$$f_i(\vec{x} + \vec{\delta}) = f_i(\vec{x}) + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j} \delta_j + \mathcal{O}(\delta^2)$$

In vector–matrix notation:

$$F(x + \delta) = F(x) + J \cdot \delta\,, \quad J_{ij} \equiv \frac{\partial f_i}{\partial x_j} = \text{Jacobian}$$

Solve matrix eq $J\delta = -F$ for the vector $\delta$.

# Summary

- Start by bracketing the root
- Bisection: halve the interval — always works!
- False position: interpolate to guess next point
- Secant: similar to false position, but does not rely on bracketing. May fly off to outer space!
- Ridders: uses 'sophisticated' interpolation — robust and fast
- Newton–Raphson: use the local gradient to guess root
  - Can go horribly wrong
  - Only general-purpose method in more than one dimension!