Lecture 11

Rootfinding

David Semeraro

University of Illinois at Urbana-Champaign

October 1, 2013

Some Motivation

- Not all problems are as easy to solve as linear systems of equations
- Even the 5th-order polynomial has no closed form solution
- Many problems fall into this category:
  - Where should wireless access points be placed? How many?
  - Where do two curved surfaces intersect?
  - What is the subsurface geology in the Los Angeles basin?
- How do we solve these problems? Iterate, getting closer (we hope!) each time.

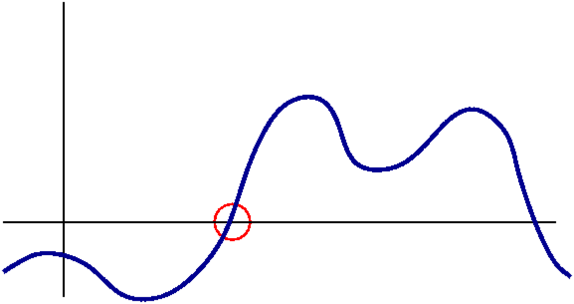
What Next?

We need to study some iterations.

- iteratively finding a root to an equation
- iteratively finding the solution to an algebraic system
- iteratively finding solutions to Ordinary Differential Equations (ODEs)
- ...

Root Finding

Given a function $f(x)$, find x so that $f(x) = 0$



Rootfinding

Goals:

- Find roots to equations
- Compare usability of different methods
- Compare convergence properties of different methods

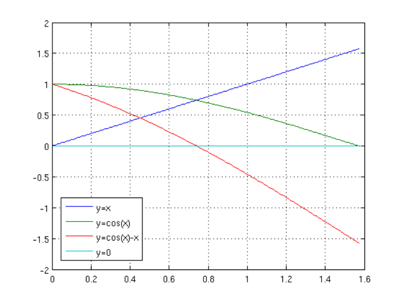
1. bracketing methods
2. Bisection Method
3. Newton's Method
4. Secant Method
5. (opt) fixed point iterations
6. (opt) special Case: Roots of Polynomials

Roots of $f(x)$

- Any single valued function can be written as $f(x) = 0$

Example

- Find x so that $\cos(x) = x$
- That is, find where $f(x) = \cos(x) - x = 0$



Analyze your Application

- Is the function complicated to evaluate?
  - lots of expresions?
  - singularities?
  - simplify? polynomial?
- How accuracte does our root need to be?
- How fast/robust should our method be?

!

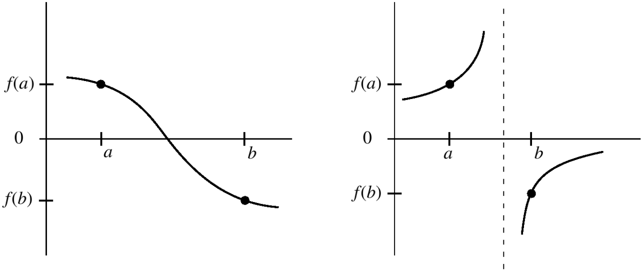
From this, you can pick the right method...

Basic Root Finding Strategy

1. Plot the function
   - Get an initial guess
   - Identify problematic parts
2. Start with the initial guess and iterate

Bracket Basics

- A root x is *bracketed* on $[a, b]$ if $f(a)$ and $f(b)$ have opposite sign.
- Changing signs does not guarantee bracketed, however: singularity



- Bracketing helps get an initial guess

Bracket Algorithm

Listing 1: Bracket Algorithm

```
given: $f(x)$, $x_{min}$, $x_{max}$, $n$

$dx = (x_{max} - x_{min})/n$
$x_{left} = x_{min}$
$i = 0$

while $i < n$
  $i = i + 1$
  $x_{right} = x_{left} + dx$
  if $f(x)$ changes sign in $[x_{left}, x_{right}]$
    save $[x_{left}, x_{right}]$ as an interval with a root
  end
  $x_{left} = x_{right}$
```

Testing Sign

$f(a) \times f(b) < 0$

Should we use?

```
fa = myfunc(a);
fb = myfunc(b);

if(fa*fb<0)
  (save)
end
```

Better Sign Test

!

Nope. Underflow...

sign()

Use Python's sign

```
fa = myfunc(a);
fb = myfunc(b);
import numpy as np

if(np.sign(fa) != np.sign(fb))
  (save)
end
```

Moving forward...

Bracketing is fine. But we need to find the actual root:

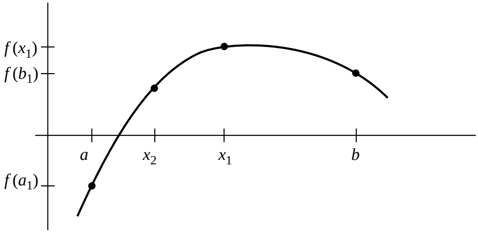
- Bisection
- Newton's Method
- Secant Method
- Fixed Point Iteration

Process:

1. use bracket.py to get a visual and brackets
2. search brackets with these methods

Bisection

Given a bracketed root, halve the interval while continuing to bracket the root



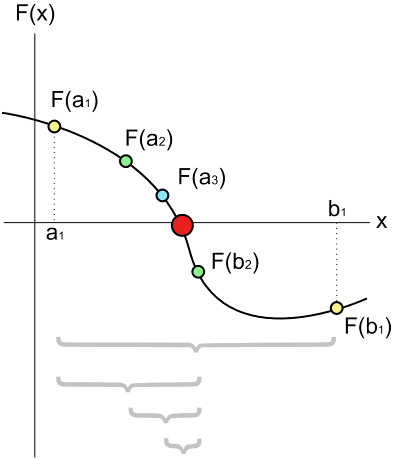
Bisection (2)

For the bracket interval $[a, b]$ the midpoint is

$$x_m = \frac{1}{2}(a + b)$$

idea:

1. split bracket in half
2. select the bracket that has the root
3. goto step 1



Bisection Algorithm

Listing 2: Bisection

```
initialize:  a = ..., b = ...
for k = 1,2,...
   x_m = a + (b − a)/2
   if sign(f(x_m)) = sign(f(x_a))
      a = x_m
   else
      b = x_m
   end
   if converged, stop
end
```

Bisection Example

Solve with bisection:

$$x - x^{1/3} - 2 = 0$$

k	a	b	x_{mid}	$f(x_{mid})$
0	3	4		
1	3	4	3.5	-0.01829449
2	3.5	4	3.75	0.19638375
3	3.5	3.75	3.625	0.08884159
4	3.5	3.625	3.5625	0.03522131
5	3.5	3.5625	3.53125	0.00845016
6	3.5	3.53125	3.515625	-0.00492550
7	3.51625	3.53125	3.5234375	0.00176150
8	3.51625	3.5234375	3.51953125	-0.00158221
9	3.51953125	3.5234375	3.52148438	0.00008959
10	3.51953125	3.52148438	3.52050781	-0.00074632

Analysis of Bisection

Let δ_n be the size of the bracketing interval at the n^{th} stage of bisection. Then

$$\delta_0 = b - a = \text{initial bracketing interval}$$
$$\delta_1 = \frac{1}{2}\delta_0$$
$$\delta_2 = \frac{1}{2}\delta_1 = \frac{1}{4}\delta_0$$
$$\vdots$$
$$\delta_n = \left(\frac{1}{2}\right)^n \delta_0$$

$$\Longrightarrow \qquad \frac{\delta_n}{\delta_0} = \left(\frac{1}{2}\right)^n = 2^{-n}$$

or $$n = \log_2\left(\frac{\delta_n}{\delta_0}\right)$$

Analysis of Bisection

$$\frac{\delta_n}{\delta_0} = \left(\frac{1}{2}\right)^n = 2^{-n} \qquad \text{or} \qquad n = \log_2\left(\frac{\delta_n}{\delta_0}\right)$$

n	$\frac{\delta_n}{\delta_0}$	function evaluations
5	3.1×10^{-2}	7
10	9.8×10^{-4}	12
20	9.5×10^{-7}	22
30	9.3×10^{-10}	32
40	9.1×10^{-13}	42
50	8.9×10^{-16}	52

Convergence Criteria

An automatic root-finding procedure needs to monitor progress toward the root and stop when current guess is close enough to the desired root.

- Convergence checking will avoid searching to unnecessary accuracy.
- Check how closeness of successive approximations

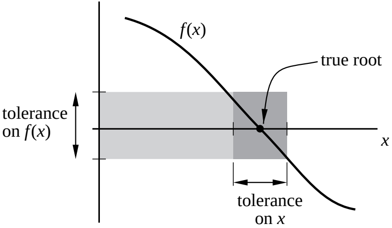
$$|x_k - x_{k-1}| < \delta_x$$

- Check how close $f(x)$ is to zero at the current guess.

$$|f(x_k)| < \delta_f$$

- Which one you use depends on the problem being solved

Convergence Criteria on x

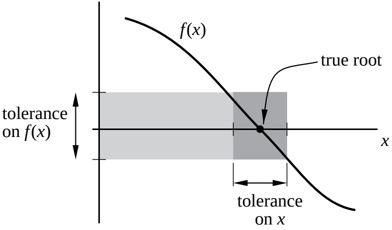


$$x_k = \text{current guess at the root}$$
$$x_{k-1} = \text{previous guess at the root}$$

Absolute tolerance: $\left| x_k - x_{k-1} \right| < \delta_x$

Relative tolerance: $\left| \dfrac{x_k - x_{k-1}}{b - a} \right| < \hat{\delta}_x$

Convergence Criteria on $f(x)$



Absolute tolerance: $\left|f(x_k)\right| < \delta_f$

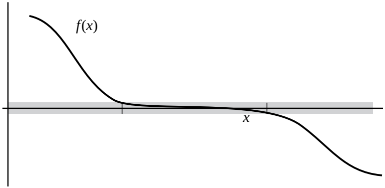
Relative tolerance:

$$|f(x_k)| < \hat{\delta}_f \max\left\{|f(a_0)|, |f(b_0)|\right\}$$

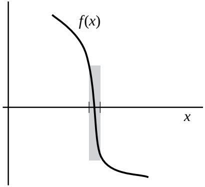
where a_0 and b_0 are the original brackets

Convergence Criteria Compared

If $f'(x)$ is small near the root, it is easy to satisfy tolerance on $f(x)$ for a large range of Δx. The tolerance on Δx is more conservative



If $f'(x)$ is large near the root, it is possible to satisfy the tolerance on Δx when $|f(x)|$ is still large. The tolerance on $f(x)$ is more conservative



Relationship Between Criteria

- How are the criteria on x and $f(x)$ related? Consider the ratio of the two criteria

$$\frac{f(x_b) - f(x_a)}{x_b - x_a}$$

- The limit of this as x_a and x_b converge to the exact answer x^* is just $f'(x^*)$.
- We can thus expect (this is not yet a proof) that

$$|f(x_b) - f(x_a)| \approx |f'(x^*)||x_b - x_a|$$

as x_a and x_b approach the solution x^*.

Convergence rate of a root finding iteration

- Let $e_n = x^* - x_n$ be the error.
- In general, a sequence is said to converge with rate r if

$$\lim_{k\to\infty} \frac{|e_{n+1}|}{|e_n|^r} = C$$

Special Cases:

- If $r = 1$ and $C < 1$, then the rate is *linear*
- If $r = 2$ and $C > 0$, then the rate is *quadratic*
- If $r = 3$ and $C > 0$, then the rate is *cubic*

Example

Convergence Rate

1. $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\ldots$
2. $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\ldots$
3. $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}\ldots$
4. $10^{-2}, 10^{-6}, 10^{-18}, \ldots$

Example

Convergence Rate

1. $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$... (linear with $C = 10^{-1}$)
2. $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$... (linear with $C = 10^{-2}$)
3. $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}$...(quadratic)
4. $10^{-2}, 10^{-6}, 10^{-18}$, ... (cubic)

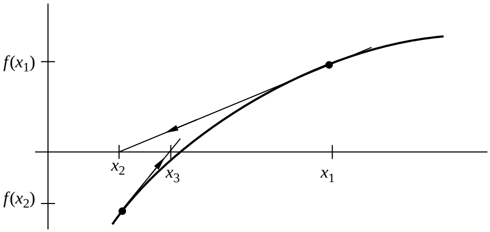
- Linear: Adds one digit of accuracy at each step
- Quadratic: Doubles the number of digits at each step

Performing Division

- Ever wondered how a computer process performs division?
- “Long” division requires lookup, subtraction, shifts
- Generates one digit and a time. Can we do better?

To answer this, we need to look at faster methods than bisection

Newton's Method



For a current guess x_k, use $f(x_k)$ and the slope $f'(x_k)$ to predict where $f(x)$ crosses the x axis.

Newton's Method

Expand $f(x)$ in Taylor Series around x_k

$$f(x_k + \Delta x) = f(x_k) + \Delta x \left.\frac{df}{dx}\right|_{x_k} + \frac{(\Delta x)^2}{2} \left.\frac{d^2 f}{dx^2}\right|_{x_k} + \ldots$$

Substitute $\Delta x = x_{k+1} - x_k$
and neglect 2^{nd} order terms to get

$$f(x_{k+1}) \approx f(x_k) + (x_{k+1} - x_k) f'(x_k)$$

where

$$f'(x_k) = \left.\frac{df}{dx}\right|_{x_k}$$

Newton's Method

Goal is to find x such that $f(x) = 0$.
Set $f(x_{k+1}) = 0$ and solve for x_{k+1}

$$0 = f(x_k) + (x_{k+1} - x_k) f'(x_k)$$

or, solving for x_{k+1}

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}$$

Newton's Method Algorithm

```
initialize:  x_1 = ...
for k = 2,3,...
   x_k = x_{k-1} - f(x_{k-1})/f'(x_{k-1})
   if converged, stop
end
```

Newton's Method Example

Solve:

$$x - x^{1/3} - 2 = 0$$

First derivative is

$$f'(x) = 1 - \frac{1}{3}x^{-2/3}$$

The iteration formula is

$$x_{k+1} = x_k - \frac{x_k - x_k^{1/3} - 2}{1 - \frac{1}{3}x_k^{-2/3}}$$

Newton's Method Example

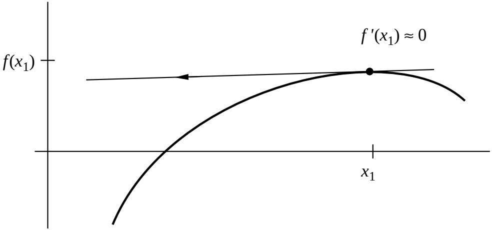
$$x_{k+1} = x_k - \frac{x_k - x_k^{1/3} - 2}{1 - \frac{1}{3}x_k^{-2/3}}$$

k	x_k	$f'(x_k)$	$f(x)$
0	3	0.83975005	-0.44224957
1	3.52664429	0.85612976	0.00450679
2	3.52138015	0.85598641	3.771×10^{-7}
3	3.52137971	0.85598640	2.664×10^{-15}
4	3.52137971	0.85598640	0.0

Conclusion

- Newton's method converges *much* more quickly than bisection
- Newton's method requires an analytical formula for $f'(x)$
- The algorithm is simple as long as $f'(x)$ is available.
- Iterations are not guaranteed to stay inside an ordinal bracket.

Divergence of Newton's Method



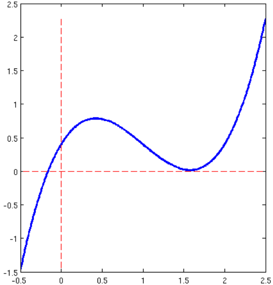
Since

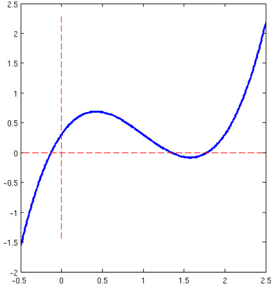
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

the new guess, x_{k+1}, will be far from the old guess whenever $f'(x_k) \approx 0$

Divergence of Newton's Method

Can you guess?





http://www.math.umn.edu/~garrett/qy/Newton.html

Newton's Method: Convergence

Recall

Convergence of a method is said to be of order r if there is a constant $C > 0$ such that

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

Newton's method is of order 2 (quadratic) when $f'(x_*) \neq 0$. For ξ between x_k and x_*

$$f(x_*) = f(x_k) + (x_* - x_k)f'(x_k) + \frac{1}{2}(x_* - x_k)^2 f''(\xi) = 0$$

So

$$\frac{f(x_k)}{f'(x_k)} + x_* - x_k + \frac{(x_* - x_k)^2}{2}\frac{f''(\xi)}{f'(x_k)} = 0$$

Then

$$x_* - x_{k+1} + \frac{1}{2}(x_* - x_k)^2 \frac{f''(\xi)}{f'(x_k)} = 0$$

Thus

$$\frac{|x_* - x_{k+1}|}{|x_* - x_k|^2} = \frac{1}{2}\frac{f''(\xi)}{f'(x_k)}$$

Reciprocal Approximation

- Consider the task of computing $1/q$ for some q without using division.
- We can write this as: find the root x of $f(x) = 1/(xq) - 1 = 0$.
- What is Newton's Method for this?
- $f'(x) = -1/(x^2 q)$. Thus

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

or

$$x_{n+1} = x_n - \frac{1/(x_n q) - 1}{-1/(x_n^2 q)}$$

Reciprocal Approximation

- Consider the task of computing $1/q$ for some q without using division.
- We can write this as: find the root x of $f(x) = 1/(xq) - 1 = 0$.
- What is Newton's Method for this?
- $f'(x) = -1/(x^2q)$. Thus

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

or

$$x_{n+1} = x_n - \frac{1/(x_nq) - 1}{-1/(x_n^2q)} \frac{x_n^2q}{x_n^2q}$$

$$x_{n+1} = x_n + x_n - x_n^2q = 2x_n - x_n^2q$$

Example: Compute 1/3

- Find the bracket:
- $1/2 > 1/3 > 1/4$

1. $x_0 = 1/4$
2. $x_1 = 2x_0 - x_0^2 q = 1/2 - 3/16 = 5/16$
3. $x_2 = 2 \times 5/2^4 - 3 \times 25/2^8 = (160 - 75)/2^8 = 85/2^8$
4. $x_3 = 2 \times 85/2^8 - 3 \times 85^2/2^{16} = 21845/2^{16}$

In 3 steps, computed 16 bits in $1/3$