

Review of Taylor Series

Read Section 1.2

Power Series

- A **power series about c** is an infinite series of the form

$$\sum_{k=0}^{\infty} a_k (x - c)^k = a_0 + a_1(x - c) + a_2(x - c)^2 + a_3(x - c)^3 + \dots$$

- In many cases, $c = 0$, and the series takes the simpler form

$$\sum_{k=0}^{\infty} a_k x^k = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots$$

- Special power series of our interest are **Taylor series**.

Taylor Series

- The **Taylor series** of a function $f(x)$ about the point c

is the power series $\sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!} (x-c)^k$.

$$\begin{aligned} f(x) &\sim f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \frac{f'''(c)}{3!}(x-c)^3 + \dots \\ &= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!} (x-c)^k \end{aligned}$$

- If $f(x)$ equals its Taylor series in an interval I containing c , $f(x)$ is said to be **analytic** in I .

Taylor Series of e^x about $c = 0$

- $f(x) \sim \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!} (x-c)^k.$
- $\frac{d}{dx} e^x = e^x$, so $f^{(k)}(c) = e^c = e^0 = 1.$
- $e^x \sim \sum_{k=0}^{\infty} \frac{x^k}{k!}.$

Some Familiar Taylor Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad (|x| < \infty)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \quad (|x| < \infty)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots = \sum_{k=0}^{\infty} (-1)^{k-1} \frac{x^k}{k} \quad (|x| < 1)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \dots$$

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!}$$

What's the error in this approximation?

Taylor's Theorem

If a function is differentiable to the $(n + 1)$ th order in a closed interval $I = [a, b]$, then for any c and x in I ,

$$f(x) = \sum_{k=0}^n \frac{f^{(k)}(c)}{k!} (x - c)^k + E_{n+1}$$

where the error term

$$E_{n+1} = \frac{f^{(n+1)}(\mu)}{(n+1)!} (x - c)^{n+1}$$

for some point between x and c .

Taylor's Theorem for $f(c+h)$

Let $h = x - c$, then $x = c + h$, and Taylor's Theorem becomes:

$$f(c+h) = \sum_{k=0}^n \frac{f^{(k)}(c)}{k!} h^k + E_{n+1}$$

where the error term

$$E_{n+1} = \frac{f^{(n+1)}(\mu)}{(n+1)!} h^{n+1}$$

for some point between c and $c+h$.

Root Finding

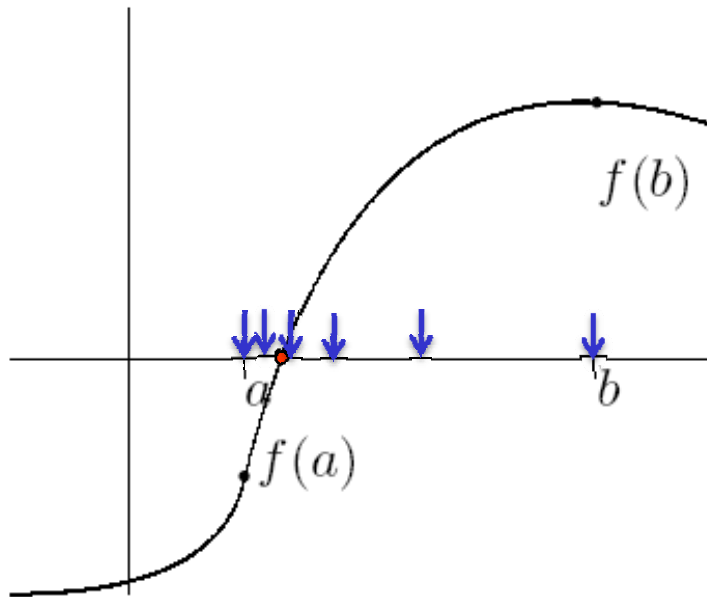
Read Chapter 3

Roots and Zeros

- Equation: $f(x) = 0$
- A number r for which $f(r) = 0$ is called
 - a **root** of the equation $f(x) = 0$
 - a **zero** of the function f
- Example: the equation $x^2 - 2x - 3 = 0$ has two roots, $x = -1$ and $x = 3$; the polynomial $x^2 - 2x - 3$ has two zeros, $x = -1$ and $x = 3$.

Bisection Method

- Assume f is continuous.
- If $f(a)$ and $f(b)$ have different signs, then there is a root in $[a, b]$.
- The **bisection method** is similar to the **binary search**.



Bisection Method

Input: function f , interval $[a, b]$ where $f(a)f(b) < 0$, ε .

Algorithm:

1. Compute the midpoint $c = (a + b) / 2$;
2. if $f(c) = 0$ then return(c); /* unlikely */
3. if $\text{sign}(f(a)) \neq \text{sign}(f(c))$
 Recurse on $I := [a, c]$
 else /* $\text{sign}(f(c)) \neq \text{sign}(f(b))$ */
 Recurse on $I := [c, b]$
 endif
4. Repeat until $\frac{|I|}{2} < \varepsilon$; then return the midpoint of I .

Bisection Method

Bisection(f, a, b, ε, k)

/ Precondition: $\text{sign}(f(a)) \neq \text{sign}(f(b))$ */*

/ $k = \text{max \# of remaining iterations}$ */*

1. $c \leftarrow (a + b) / 2$

2. if $f(c) = 0$ then return(c);

3. if ($(b - a) / 2 < \varepsilon$ or $k < 1$) then return(c);

4. if $\text{sign}(f(a)) \neq \text{sign}(f(c))$ then

5. Bisection($f, a, c, \varepsilon, k - 1$);

6. else */* $\text{sign}(f(c)) \neq \text{sign}(f(b))$ */*

7. Bisection($f, c, b, \varepsilon, k - 1$);

8. endif

Convergence of Bisection Method

How many iterations are needed?

Let $[a, b] = [a_0, b_0]$ be the initial interval.

Let $[a_i, b_i]$ be the interval of the i th recursive call.

Length of interval $[a_i, b_i]$: $b_i - a_i = \frac{b - a}{2^i}$

#iterations (recursive calls) needed:

$$\frac{1}{2} \frac{b - a}{2^i} < \varepsilon \implies 2^i > \frac{(b - a)}{2\varepsilon}$$

$$i > \log_2 \frac{(b - a)}{2\varepsilon} = \log_2(b - a) - \log_2 2\varepsilon$$

Convergence of Bisection Method

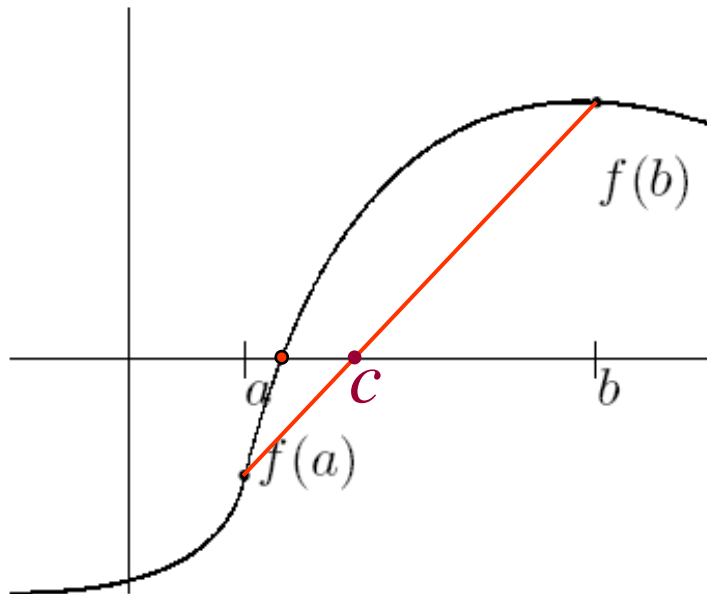
- $[a_0, b_0] = [a, b]$: the initial interval.
- $[a_i, b_i]$: the interval of the i th recursive call.
- Midpoints of these intervals c_0, c_1, c_2, \dots form a sequence of approximations to the root r . Let $e_i = |r - c_i|$.

$$\begin{array}{ll} c_0 = (a_0 + b_0) / 2 & e_0 \leq (b - a) / 2 \\ c_1 = (a_1 + b_1) / 2 & e_1 \leq (b - a) / 2^2 \\ \vdots & \vdots \\ c_i = (a_i + b_i) / 2 & e_i \leq (b - a) / 2^{i+1} \\ \vdots & \end{array}$$

- How fast does the sequence (c_0, c_1, c_2, \dots) converge to r ?
- **Approximately** 1 bit per iteration.

Regula Falsi: Improved Bisection Method

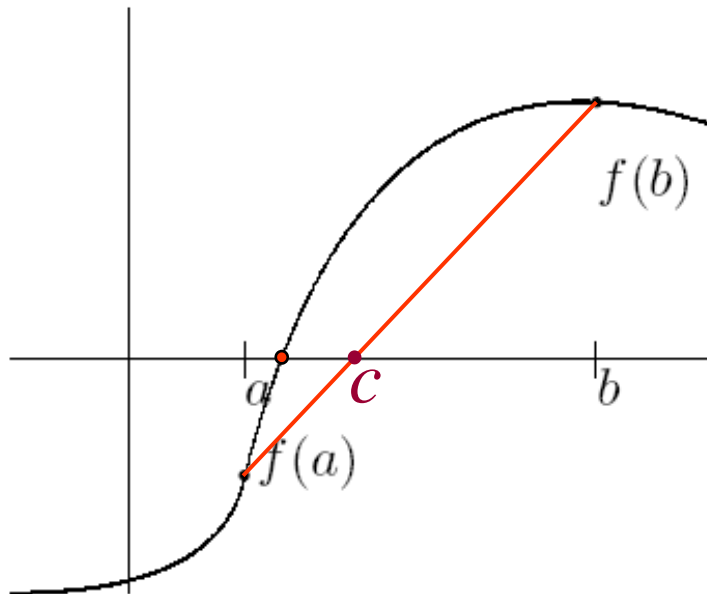
- Choose c to be the point where the secant line between $(a, f(a))$ and $(b, f(b))$ intersects the x axis.
- Regula Falsi often converges faster than bisection.



The equation of secant line:

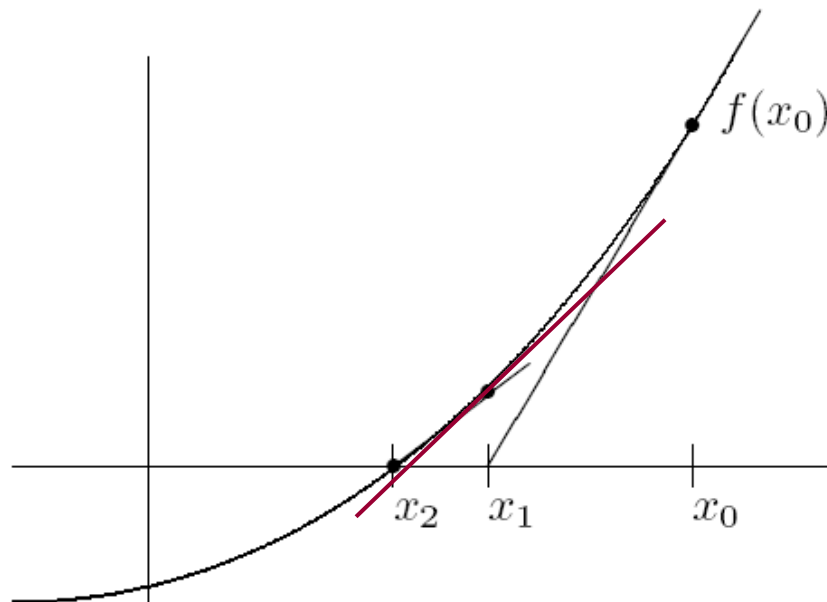
$$\frac{y - f(b)}{x - b} = \frac{f(a) - f(b)}{a - b}$$

When $y = 0$, $x = \frac{bf(a) - af(b)}{f(a) - f(b)}$, which is c .



Newton's Method

- Approximate $f(x)$ by a straight line $l(x)$, and use the root of $l(x)$ as an approximation to the root of $f(x)$.
- $l(x)$: a line tangent to the function at a current guess.



Newton's Method

- The **slope** of the tangent line is the derivative $f'(x_0)$. So:

$$y = f'(x_0)(x - x_0) + f(x_0)$$

- When $y = 0$, $x = x_0 - \frac{f(x_0)}{f'(x_0)}$.

- This gives us a sequence:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton's Method: Example

- Find a zero for $f(x) = x^3 + 2x^2 - 11x - 12$
- What is $f'(x)$:

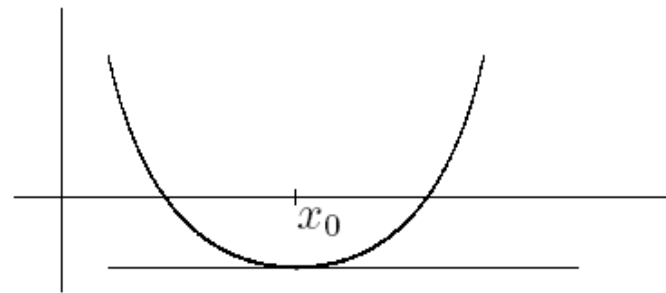
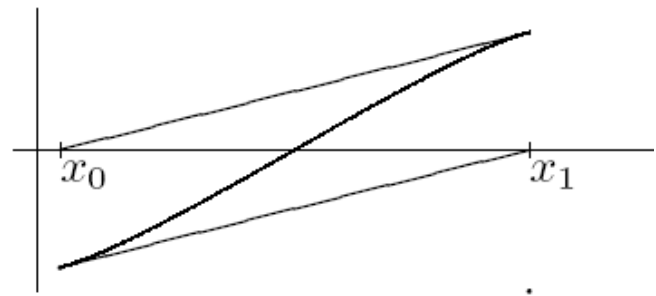
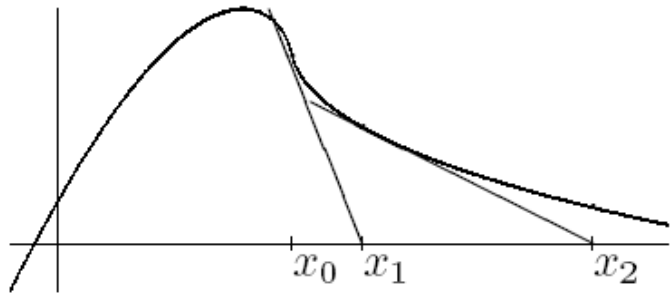
$$f'(x) = 3x^2 + 4x - 11$$

- So

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^3 + 2x_i^2 - 11x_i - 12}{3x_i^2 + 4x_i - 11}$$

- Start with $x_0 = 30$
- Does this method always converge?

Newton's Method: Bad Functions



Newton's Method: Convergence

- Doesn't always converge.
- Assume: f, f', f'' are continuous in a neighborhood of r (the root) and $f'(r) \neq 0$.
- The sequence (x_0, x_1, x_2, \dots) converges if x_0 is close enough to r .
- Converges quadratically:

$$|r - x_{i+1}| \leq c |r - x_i|^2$$

or

$$e_{i+1} \leq ce_i^2$$

where c is a positive constant.

Quadratic Convergence

- The error terms satisfy: $e_{i+1} \leq ce_i^2$
- Example: Suppose $c = 1$, and $|e_0| \leq 2^{-1}$.

$$e_0 \leq 2^{-1}$$

$$e_1 \leq 2^{-2}$$

$$e_2 \leq 2^{-4}$$

$$e_3 \leq 2^{-8}$$

$$e_4 \leq 2^{-16}$$

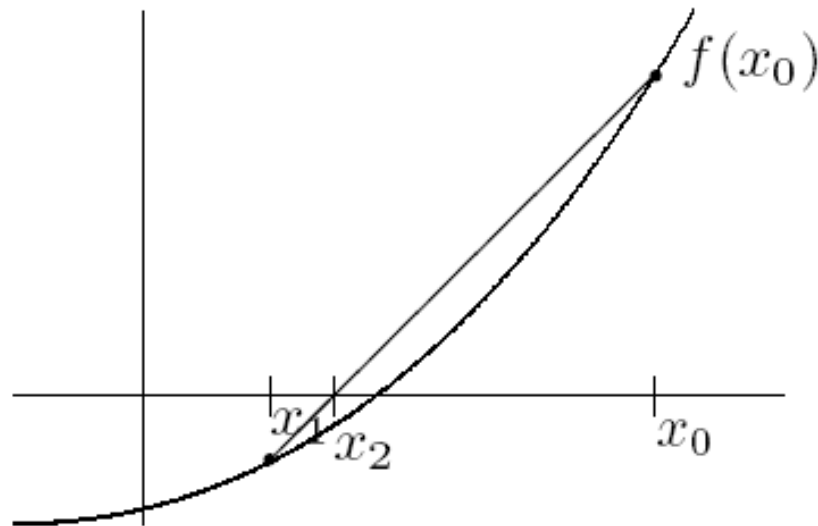
- The number of correct digits **doubles** with each iteration.
- Five or six iterations are often sufficient to yield full machine precision.

Newton's Method: Remarks

- We must be able to evaluate $f'(x)$, which is not always the case.
- Does not guarantee convergence.
- But it is very fast, if the starting point is close enough to the root.

Secant Method

- As in Newton's method, approximate the function with a line, but this time a **secant line**.
- Two starting points: x_0 and x_1 .



Secant Method

- Let the intersection point be $(x_2, 0)$. Equate the slopes:

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{0 - f(x_1)}{x_2 - x_1}$$

Rearrange: $x_2 = x_1 - f(x_1) \frac{x_1 - x_0}{f(x_1) - f(x_0)}$

- Do the same thing for x_3 using x_1 and x_2 .

- In general: $x_{i+1} = x_i - f(x_i) \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$

Secant Method vs Newton's Method

- Secant method is similar to Newton's method except that

$$f'(x_i) \text{ is approximated by } f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

- So how does it compare to Newton's method?
- How many function evaluations are needed per iteration?
- Will it converge?
- How about speed of convergence?

$$e_{i+1} \leq ce_i^\alpha, \quad \alpha \approx 1.62, \text{ the golden ratio.}$$

- Not quite as fast as Newton's method but still superlinear and much faster than bisection. But if you consider number of evaluations it could be faster than Newton's.

Root Finding Comparison

- Bisection Method
 - Always converges, but does so slowly,
 - Needs two starting values on opposite sides of the root.
- Newton's Method
 - May not converge, but converges quickly near the root.
 - Needs two function evaluations per iteration.
 - Needs to be able to evaluate f' .
- Secant Method
 - May not converge, but converges quickly near the root.
 - Needs one function evaluation per iteration.
 - Doesn't need f' .

Hybrid Methods

- Most popular
- Combine Bisection with Newton's or Secant.
- Apply Bisection for a few steps and then switch to Newton or Secant.

Root Finding with Matlab

- Matlab has a function `fzero` that returns a root of a given function.

Define a Function

- The following Matlab script defines a function

$$f(x) = x^5 - x^3 + \sin(x) + 7.$$

- Store the script in a file named **f.m**

```
function y = f(x)
```

```
% define a function named "f"
```

```
y = x^5 - x^3 + sin(x) + 7;
```

Plot the Function

- `linspace(a,b)` defines a 100 equally-spaced points interval $[a,b]$.
- `linspace(a,b)` draw the diagram. Try `plot(x,y,'r*')`, `plot(x,y,'g+')`.
- Use `help plot` to get more information.

```
x = linspace(-pi, pi);  
% an array of 100 equally-spaced points in [-pi,pi]  
for i = 1:100  
    y(i) = f(x(i))  
end  
plot(x,y)
```

Find the Root

- `fzero('fname', [a b])` tries to find a zero of the function `fname` in the interval `[a,b]`.
- `fzero('fname', a)` tries to find a zero of the function `fname` near `a`.

```
x = linspace(-pi,pi);  
for i = 1:100  
    y(i) = f(x(i));  
end;  
root = fzero('f',[-2 2]);  
plot(x,y, root, 0, 'ro');  
root, f_val_at_root = f(root)
```