June 11, 2015
Review from Last Class

- Floating Point and Machine Representation of Numbers (0.3)
Review from Last Class

- Floating Point and Machine Representation of Numbers (0.3)
- Newton’s Method (1.4)
Fixed Point Iteration and Convergence
MATLAB Example

Find the root \( r = 0.6823278\ldots \) of \( f(x) = x^3 + x - 1 = 0 \) using Fixed Point Iteration and Newton’s Method. Examine the convergence rate.
Newton’s Method

We need 3 things:

1. \( f(x) \)
2. \( f'(x) \)
3. \( x_0 \)

\[
\begin{align*}
f(x) & = x^3 + x - 1 \\
f'(x) & = 3x^2 + 1 \\
x_0 & = -0.7
\end{align*}
\]
Newton’s Method

What’s the theoretical convergence rate, $M$?

\[ f'(x) = 3x^2 + 1 \]
\[ f''(x) = 6x \]
\[ r = 0.6823278 \]

So we can calculate

\[
M = \left| \frac{f''(r)}{2f'(r)} \right| = \left| \frac{6(0.6823278)}{2(3(0.6823278)^2 + 1)} \right| = 0.85407
\]
Fixed Point Iteration

We need 3 things:

1. \( g(x) \)
2. \( x_0 \)

\[
\begin{align*}
f(x) &= x^3 + x - 1 \\
g(x) &= (1 - x)^{\frac{1}{3}} \\
x_0 &= 0.5
\end{align*}
\]
Fixed Point Method

What’s the theoretical convergence rate, $S$?

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

$$r = 0.6823278$$

So we can calculate

$$S = |g'(r)| = \left| -\frac{1}{3}(1-0.6823278)^{-\frac{2}{3}} \right|$$

$$S = 0.716$$
The drawback of Newton’s Method is...
The drawback of Newton’s Method is...the derivative!
The drawback of Newton’s Method is...the derivative!

BUT derivative = more information = faster convergence.
The drawback of Newton’s Method is...the derivative!

BUT derivative = more information = faster convergence.

What if the derivative is not available? We still want something that’s better than Fixed Point Iteration and Bisection Method.
Secant Method

We can approximate the derivative as

\[ f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \]

This is known as a *finite difference approximation*.
Secant Method

We can derive the Secant Method from Newton’s Method

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \]
Secant Method

We can derive the Secant Method from Newton’s Method

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

\[
x_{i+1} = x_i - \frac{f(x_i)}{f(x_i) - f(x_{i-1})}
\]

\[
x_{i+1} = x_i - \frac{f(x_i)}{x_i - x_{i-1}}
\]

\[
x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}
\]
Secant Method

We can derive the Secant Method from Newton’s Method

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

\[ x_{i+1} = x_i - \frac{f(x_i)}{f(x_i) - f(x_{i-1})} \]

\[ x_{i+1} = x_i - \frac{f(x_i) (x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \]

We just need 2 initial guesses, \( x_0 \) and \( x_1 \), instead of 1
Secant Method has better than linear convergence, but not quite quadratic convergence...
Section 1.5

\( f'(r) \neq 0: \)

Secant Method has better than linear convergence, but not quite quadratic convergence...

\[
e^{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right|^{0.62} e_i^{1.62}
\]

It has **superlinear** convergence
Section 1.5

\( f'(r) \neq 0 \):

Secant Method has better than linear convergence, but not quite quadratic convergence...

\[
e_{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right|^{0.62} e_i^{1.62}
\]

It has **superlinear** convergence

\( f'(r) = 0 \): linear convergence
Section 1.5

Other root-finding algorithms without derivatives:

1. False Position Method- Combination of Bisection Method and Secant Method
2. Muller’s Method- Generalization of Secant Method with parabolas
3. Inverse Quadratic Interpolation (IQI)- Generalization of Secant Method with parabolas
4. Brent’s Method- Hybrid method that combines Bisection, Secant and IQI. Known as *fzero* in MATLAB (p 65)
Chapter 2: Systems of Equations
Let's get familiar with matrices in MATLAB
We have the system of equations

\begin{align*}
x_1 - 2x_2 + x_3 &= -2 \\
2x_1 + x_2 - 3x_3 &= 1 \\
-2x_1 + 2x_2 - x_3 &= 1
\end{align*}
We have the system of equations

\[
\begin{align*}
    x_1 - 2x_2 + x_3 &= -2 \\
    2x_1 + x_2 - 3x_3 &= 1 \\
    -2x_1 + 2x_2 - x_3 &= 1
\end{align*}
\]

\(n\) equations in \(n\) unknowns. Solve for \(x_1, x_2, \ldots, x_n\)
We can represent the system in matrix form $Ax = b$

\[
\begin{bmatrix}
1 & -2 & 1 \\
2 & 1 & -3 \\
-2 & 2 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= 
\begin{bmatrix}
-2 \\
1 \\
1
\end{bmatrix}
\]
We can represent the system in tableau form

\[
\begin{bmatrix}
1 & -2 & 1 & \vert & -2 \\
2 & 1 & -3 & \vert & 1 \\
-2 & 2 & 1 & \vert & 1
\end{bmatrix}
\]
Section 2.1

We can represent the system in tableau form

\[
\begin{bmatrix}
1 & -2 & 1 & | & -2 \\
2 & 1 & -3 & | & 1 \\
-2 & 2 & 1 & | & 1 \\
\end{bmatrix}
\]

Solve this in 2 parts

1. Elimination step (get 0’s in lower triangle)
2. Back substitute to find \(x_n, x_{n-1}, \ldots x_1\)
Step 1: Elimination

\[
\begin{bmatrix}
1 & -2 & 1 & | & -2 \\
2 & 1 & -3 & | & 1 \\
-2 & 2 & 1 & | & 1 \\
\end{bmatrix} \rightarrow \text{subtract } 2 \times \text{row 1 from row 2} \rightarrow \begin{bmatrix}
1 & -2 & 1 & | & -2 \\
0 & 5 & -5 & | & 5 \\
-2 & 2 & 1 & | & 1 \\
\end{bmatrix} \rightarrow \text{subtract } -2 \times \text{row 1 from row 3} \rightarrow \begin{bmatrix}
1 & -2 & 1 & | & -2 \\
0 & 5 & -5 & | & 5 \\
0 & 0 & -1 & | & -1 \\
\end{bmatrix} \rightarrow \text{subtract } -\frac{2}{5} \times \text{row 2 from row 3} \rightarrow \begin{bmatrix}
1 & -2 & 1 & | & -2 \\
0 & 5 & -5 & | & 5 \\
0 & 0 & -1 & | & -1 \\
\end{bmatrix}
\]
Step 2: Back substitution

\[-1x_3 = -1\]
\[5x_2 - 5x_3 = 5\]
\[x_1 - 2x_2 + 1x_3 = -2\]
Section 2.1

Step 2: Back substitution

\[-1x_3 = -1\]
\[5x_2 - 5x_3 = 5\]
\[x_1 - 2x_2 + 1x_3 = -2\]

\[-1x_3 = -1 \rightarrow x_3 = 1\]
\[5x_2 - 5(1) = 5 \rightarrow 5x_2 = 10 \rightarrow x_2 = 2\]
\[x_1 - 2(2) + 1(1) = -2 \rightarrow x_1 = 4 - 1 - 2 = 1\]
Solution

\[ \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \]
I want to do this in MATLAB...
I want to do this in MATLAB... Think of the above system as

\[ a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \]
\[ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \]
\[ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \]
I want to do this in MATLAB... Think of the above system as

\[
\begin{align*}
    a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\
    a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\
    a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3
\end{align*}
\]

We’ll get back to operation counts in Gaussian elimination later.
Homework

- Read 1.5, 2.1, 2.3
- Exercises
  - 1.5: 1, 7
  - 2.1: 1, 3
- Project 2