CS 221 Lecture 9

Tuesday, 1 November 2011

Some slides in this lecture are from the publisher's slides for Engineering Computation: An Introduction Using MATLAB and Excel

Today's Agenda

1. Announcements
2. Passing functions as parameters to other functions
3. Solving equations
4. Numerical Root-finding methods (Text: Ch. 6)
   a. Bisection
   b. Newton's Method
   c. MATLAB's built-in methods: fzero() and roots()
5. Lab Quiz – What to expect

1. Announcements

- Problem Set 3 is out
  - Due Monday 7 November
- Lab Quiz 2 this Thursday
  - All-MATLAB
  - Example problems later & posted on Web page

2. Functions As Parameters

- Sometimes it's useful to pass a function as a parameter to another function
  - You want to use a function without knowing exactly what it is
  - This is useful for:
    - Plotting – see fplot()
    - Numerical (not symbolic) manipulations
      - Integration: finding the area under the curve of f(x)
      - Finding roots of equations f(x) = 0

Two Ways to Pass Functions As Parameters

1. Pass the name of the function as a string (in quotes)

   Examples:

   fplot('cos', [-pi, pi])

   g('f',1,2)

2. Pass the name of the function preceded by @
   - Syntax: @funcname
   - Semantics: @cos is a function handle – a pointer to the cosine function

   Examples

   fplot(@cos, [-pi, pi])

   g(@f,1,2)

Function Handles have many uses.

- You can assign a function handle to a variable

  ```
  func = @cos;
  ```

  Now func(x) returns the same thing as cos(x)
- Pass a parameter to a user-defined function

  Example: bisect() – coming soon

  – Use of function handles is required
- "function handle" is a real type in MATLAB
  – like "double" or "char"

3. Solving Equations

- We consider algebraic equations of the form:

$$f(x) = 0$$

  Single unknown: x
- Linear: multiply and add constants:

  $f(x) = 3x + 4.0321;$
- Polynomials:

  Sums of (non-negative) integer powers of x, multiplied by real constants

  $f(x) = 3x^5 + 2.5x^3 + x^2 + 100x - 10.5$
- General Nonlinear

  Non-integer powers of x, transcendentals, logs, etc:

  $f(x) = x^{3/2} + \log x + x \sin x$

Finding Roots of Equations

- We want the equation in the form f(x) = 0
  - This may may require some massaging
- Example:

$$7y = 6$$

- If we say that

$$f(y) = 7y - 6$$

then when $f(y) = 0$, the equation is satisfied

Solution Approaches

- Analytical (algebra)
  - What you’ve been doing since high school
  - Fine for linear and some polynomial equations
  - Often impractical in “real world” situations
- Graphical
  - Plot the function see where it crosses the x-axis
- Numerical

4. Numerical Root-Finding Methods

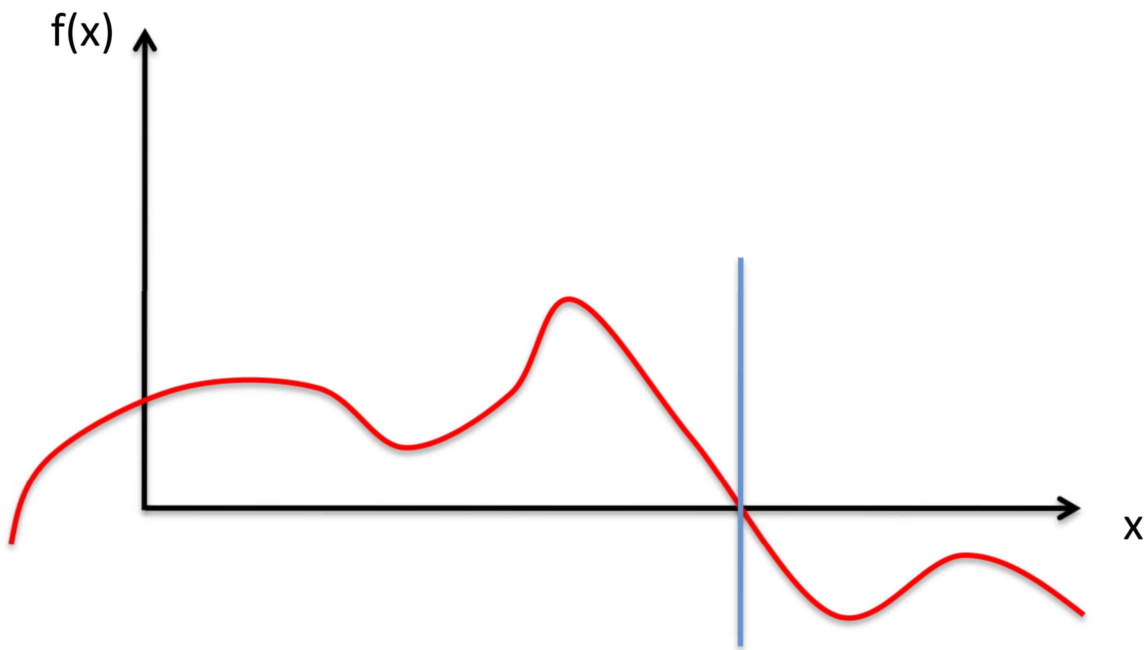
Finding Roots

- Problem: given f(x), find the roots of f
  - I.e., value(s) of x for which f(x) = 0
- Approaches:
  - Solve <u>analytically</u>
    - Quadratic equations, some polynomials, ...
  - Solve <u>graphically</u>
    - Low precision
  - Solve <u>numerically</u>
    - When the other methods aren't adequate
    - All you need is the function itself

      That is: you give it x, it gives back f(x)

Graphical Method

- Graph the function in the region of interest
- See where it crosses the y-axis



Finding Roots Numerically: General Approach

```
<Given function f plus starting_info>
<initialize>
estimate = <set based on starting_info>
while estimated_error > error_spec
   old_estimate = estimate;
    estimate = <refine estimate, using f>;
   estimated_error =        % update the error estimate
        (old_estimate – estimate)/estimate;
end
```

The Bisection Method

- Principle:

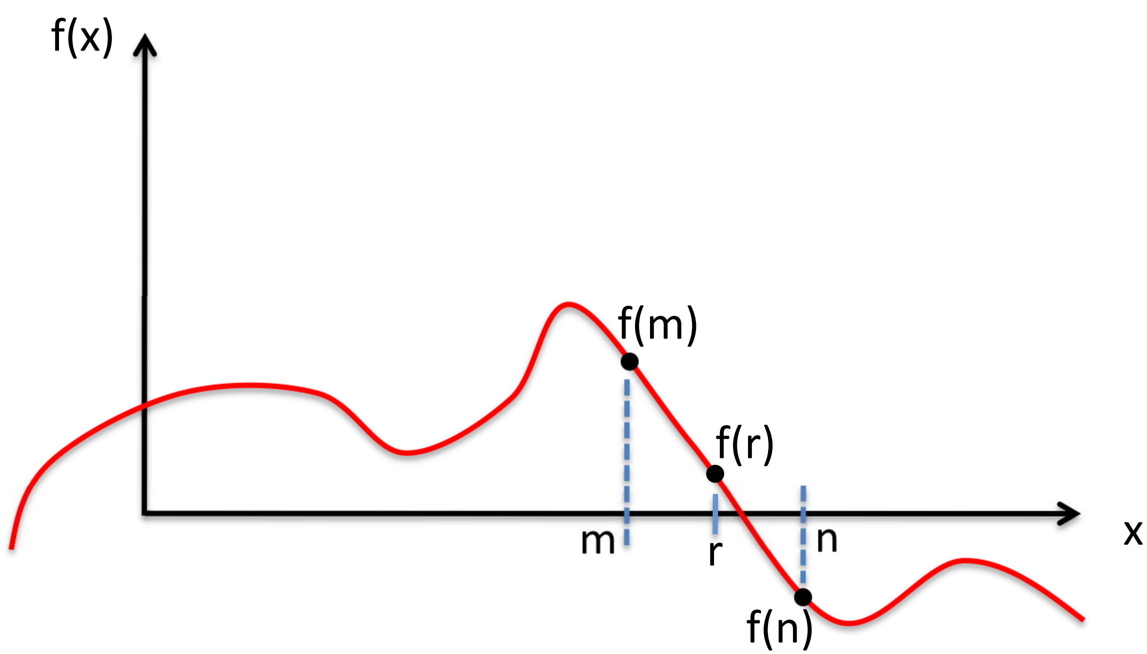
  If the signs of f(m) and f(n) differ, there exists an odd number of roots (at least 1) between m and n

  ...or there is a discontinuity between m and n

- Method:
  - Find m and n such that $f(m) \times f(n) < 0$
  - Compute $r = (m+n)/2$ and $f(r)$
  - If $f(r)$ has the same sign as $f(m)$, replace m by r
    else [$f(r)$ has same sign as $f(n)$] replace n by r

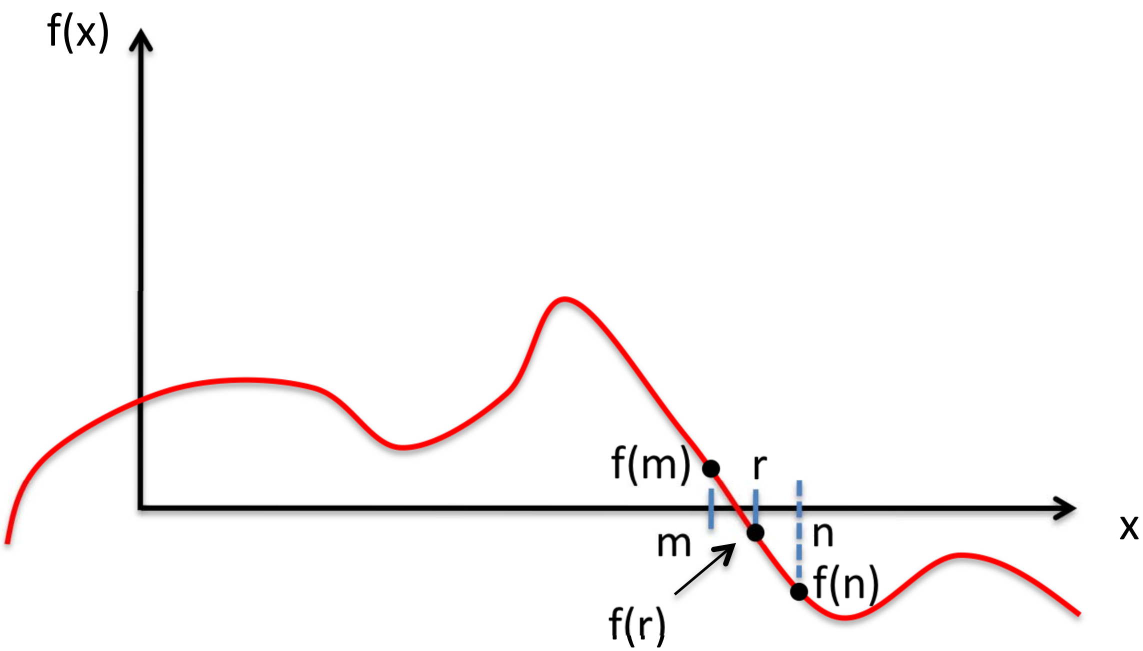
Bisection Method: Error estimation

- We know there is a root between m and n
- Estimate = (f(m) + f(n))/2
- Root must be within (f(m) – f(n))/2 of this estimate!
- The interval [m,n] shrinks with every iteration!



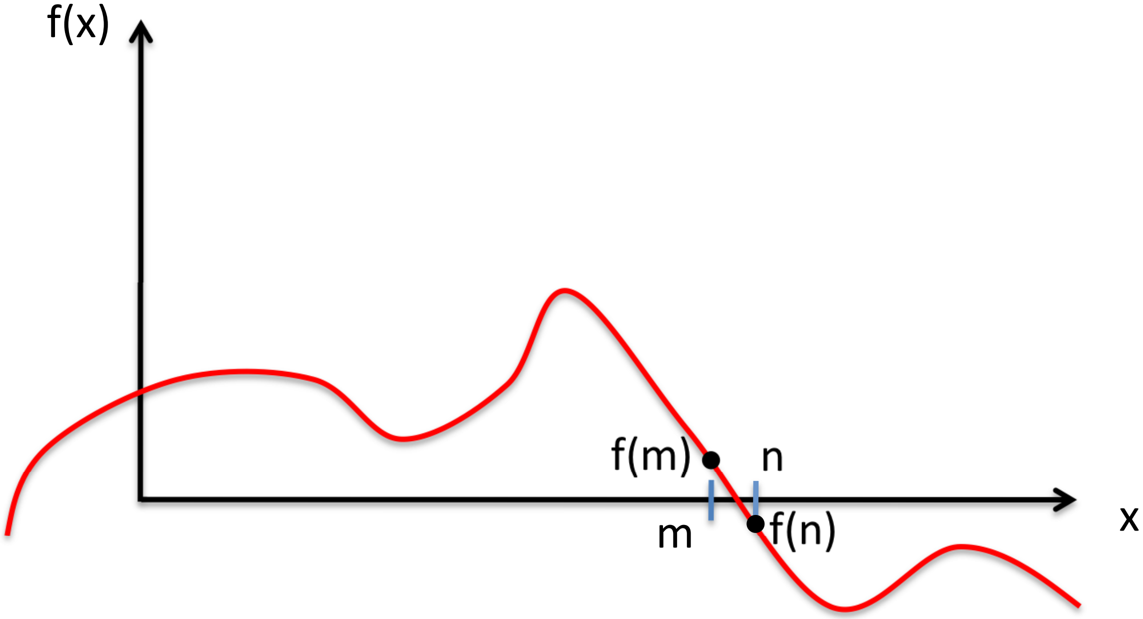
Bisection Method: Error estimation

- We know there is a root between m and n
- Estimate = (f(m) + f(n))/2
- Root <u>must</u> be within |m – n|/2 of this estimate!
- The interval [m,n] shrinks with every iteration



Bisection Method: Error estimation

- We know there is a root between m and n
- Estimate = (f(m) + f(n))/2
- Root <u>must</u> be within |m – n|/2 of this estimate!
- The interval [m,n] shrinks with every iteration



Bisection Method Example

- Consider the function f(x) = 7y – 6
- Consider an initial interval of x_{low} = -10 to x_{hi} = 10

$$f(-10) = -76$$

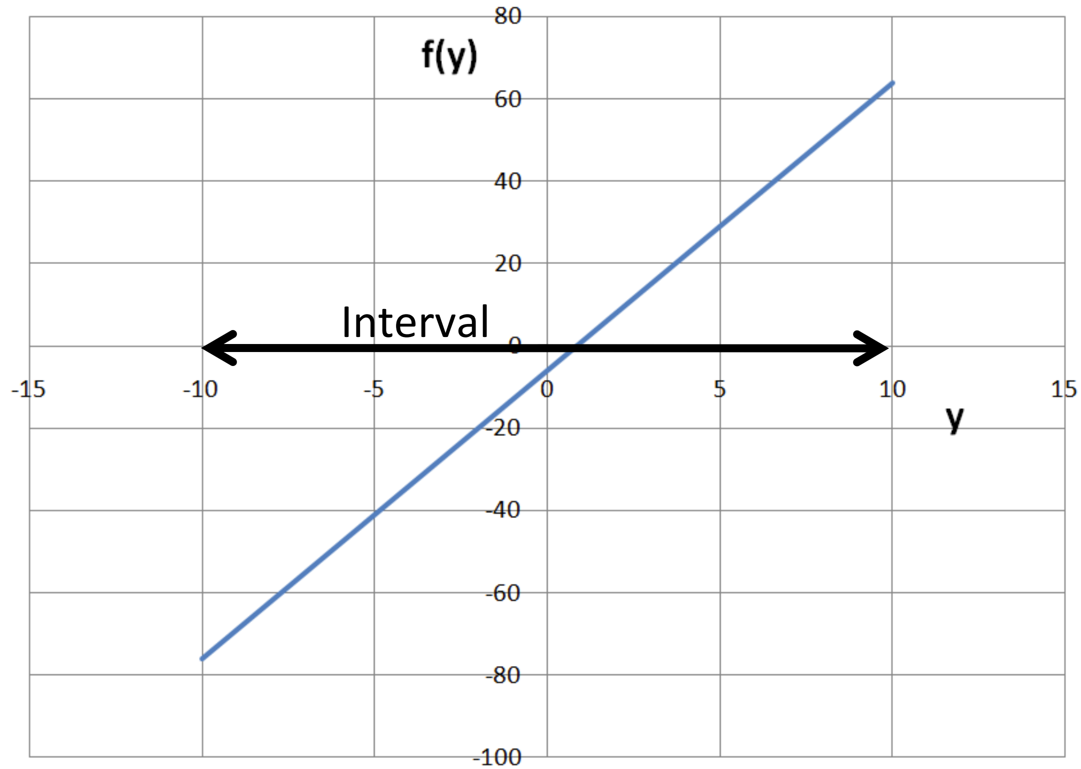
$$f(10) = 64$$

- Since the signs are opposite, we know that the method will converge to a root of the equation
- The value of the function at the midpoint of the interval is:

$$f(0) = -6$$

Bisection Method Example

- The method can be better understood by looking at a graph of the function:

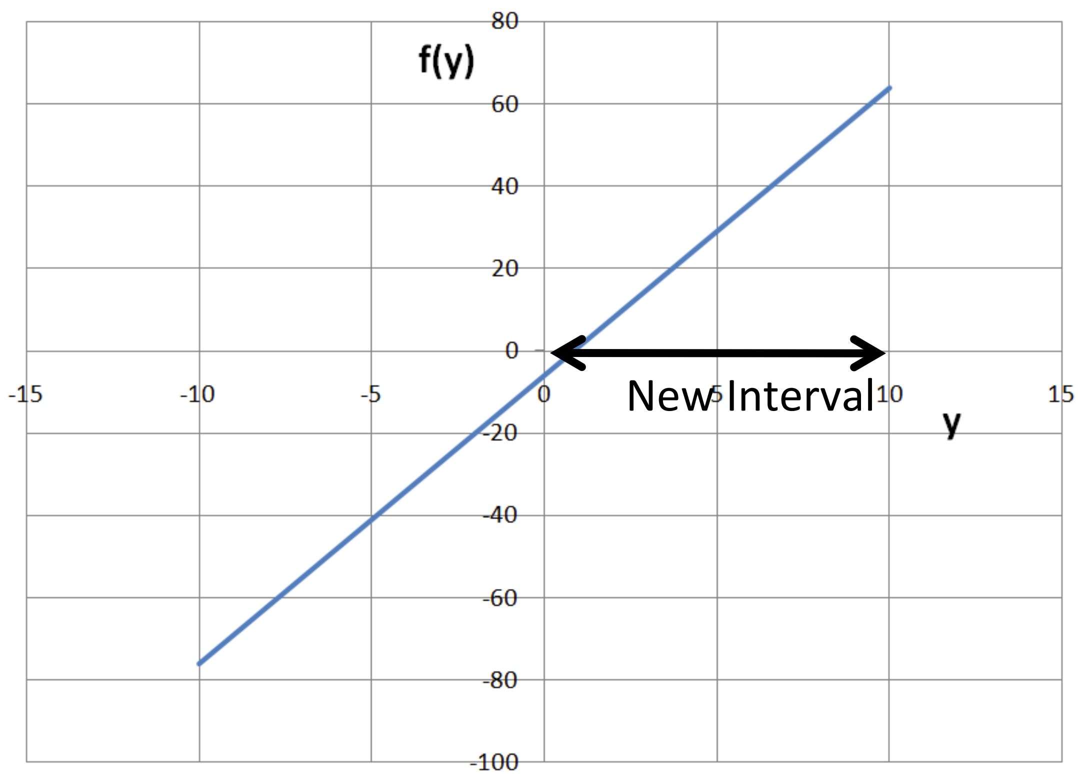


Bisection Method Example

- Now we eliminate half of the interval, keeping the half where the sign of *f*(midpoint) is opposite the sign of *f*(endpoint)
- In this case, since $f(x_{\text{mid}}) = -6$ and $f(x_{\text{upper}}) = 64$, we keep the upper half of the interval, since the function crosses zero in this interval

Bisection Method Example

- The interval has now been bisected, or halved:



Bisection Method Example

- New interval: $x_{\text{lower}} = 0$, $x_{\text{upper}} = 10$, $x_{\text{mid}} = 5$
- Function values:

$$f(0) = -6$$
$$f(5) = 29$$
$$f(10) = 64$$

- Since $f(x_{\text{lower}})$ and $f(x_{\text{mid}})$ have opposite signs, the lower half of the interval is kept

Bisection Method Example

- At each step, the difference between the high and low values of *x* is compared to 2*(allowable error)
- If the difference is greater, than the procedure continues
- Suppose we set the allowable error at 0.0005. As long as the width of the interval is greater than 0.001, we will continue to halve the interval
- When the width is less than 0.001, then the midpoint of the range becomes our answer

Bisection Method Example

- Or course, we know that the exact answer is 6/7 (0.857143)
- If we wanted our answer accurate to 5 decimal places, we could set the allowable error to 0.000005
- This increases the number of iterations only from 16 to 22 – the halving process quickly reduces the interval to very small values
- Even if the initial guesses are set to -10,000 and 10000, only 32 iterations are required to get a solution accurate to 5 decimal places

Bisection Method Example - Polynomial

- Now consider this example:

$$x^2 - 2x = 8$$

- Use the bisection method, with allowed error of 0.0001

Bisection Method Example - Polynomial

- If limits of -10 to 0 are selected, the solution converges to x = -2

	A	B	C	D	E	F	G	H
1	Lower	Upper	Difference	< 2*error?	Mid	f(low)	f(mid)	Product
2	-10	0	10	NO	-5	112	27	3024
3	-5	0	5	NO	-2.5	27	3.25	87.75
4	-2.5	0	2.5	NO	-1.25	3.25	-3.9375	-12.7969
5	-2.5	-1.25	1.25	NO	-1.875	3.25	-0.73438	-2.38672
6	-2.5	-1.875	0.625	NO	-2.1875	3.25	1.160156	3.770508
7	-2.1875	-1.875	0.3125	NO	-2.03125	1.160156	0.188477	0.218662
8	-2.03125	-1.875	0.15625	NO	-1.95313	0.188477	-0.27905	-0.05259
9	-2.03125	-1.95313	0.078125	NO	-1.99219	0.188477	-0.04681	-0.00882
10	-2.03125	-1.99219	0.039063	NO	-2.01172	0.188477	0.07045	0.013278
11	-2.01172	-1.99219	0.019531	NO	-2.00195	0.07045	0.011723	0.000826
12	-2.00195	-1.99219	0.009766	NO	-1.99707	0.011723	-0.01757	-0.00021
13	-2.00195	-1.99707	0.004883	NO	-1.99951	0.011723	-0.00293	-3.4E-05
14	-2.00195	-1.99951	0.002441	NO	-2.00073	0.011723	0.004395	5.15E-05
15	-2.00073	-1.99951	0.001221	NO	-2.00012	0.004395	0.000732	3.22E-06
16	-2.00012	-1.99951	0.00061	NO	-1.99982	0.000732	-0.0011	-8E-07
17	-2.00012	-1.99982	0.000305	NO	-1.99997	0.000732	-0.00018	-1.3E-07
18	-2.00012	-1.99997	0.000153	NO	-2.00005	0.000732	0.000275	2.01E-07
19	-2.00005	-1.99997	7.63E-05	YES	-2.00001	0.000275	4.58E-05	1.26E-08
20	-2.00001	-1.99997	3.81E-05	YES	-1.99999	4.58E-05	-6.9E-05	-3.1E-09
21	-2.00001	-1.99999	1.91E-05	YES	-2	4.58E-05	-1.1E-05	-5.2E-10

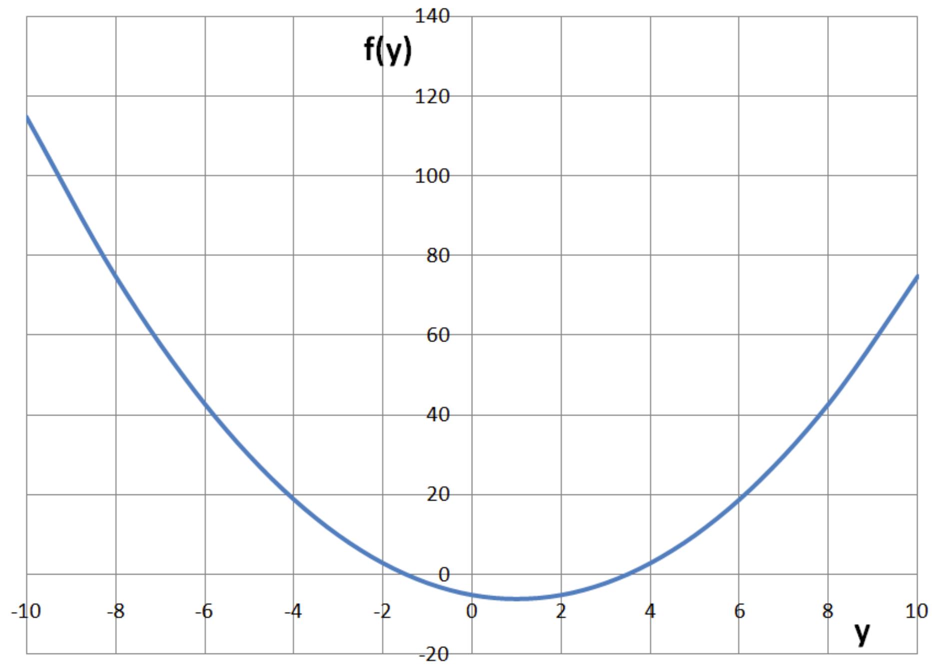
Bisection Method Example - Polynomial

- If limits of 0 to 10 are selected, the solution converges to $x = 4$

	A	B	C	D	E	F	G	H
1	Lower	Upper	Difference	< 2*error?	Mid	f(low)	f(mid)	Product
2	0	10	10	NO	5	-8	7	-56
3	0	5	5	NO	2.5	-8	-6.75	54
4	2.5	5	2.5	NO	3.75	-6.75	-1.4375	9.703125
5	3.75	5	1.25	NO	4.375	-1.4375	2.390625	-3.43652
6	3.75	4.375	0.625	NO	4.0625	-1.4375	0.378906	-0.54468
7	3.75	4.0625	0.3125	NO	3.90625	-1.4375	-0.55371	0.795959
8	3.90625	4.0625	0.15625	NO	3.984375	-0.55371	-0.09351	0.051775
9	3.984375	4.0625	0.078125	NO	4.023438	-0.09351	0.141174	-0.0132
10	3.984375	4.023438	0.039063	NO	4.003906	-0.09351	0.023453	-0.00219
11	3.984375	4.003906	0.019531	NO	3.994141	-0.09351	-0.03512	0.003284
12	3.994141	4.003906	0.009766	NO	3.999023	-0.03512	-0.00586	0.000206
13	3.999023	4.003906	0.004883	NO	4.001465	-0.00586	0.008791	-5.2E-05
14	3.999023	4.001465	0.002441	NO	4.000244	-0.00586	0.001465	-8.6E-06
15	3.999023	4.000244	0.001221	NO	3.999634	-0.00586	-0.0022	1.29E-05
16	3.999634	4.000244	0.00061	NO	3.999939	-0.0022	-0.00037	8.05E-07
17	3.999939	4.000244	0.000305	NO	4.000092	-0.00037	0.000549	-2E-07
18	3.999939	4.000092	0.000153	NO	4.000015	-0.00037	9.16E-05	-3.4E-08
19	3.999939	4.000015	7.63E-05	YES	3.999977	-0.00037	-0.00014	5.03E-08

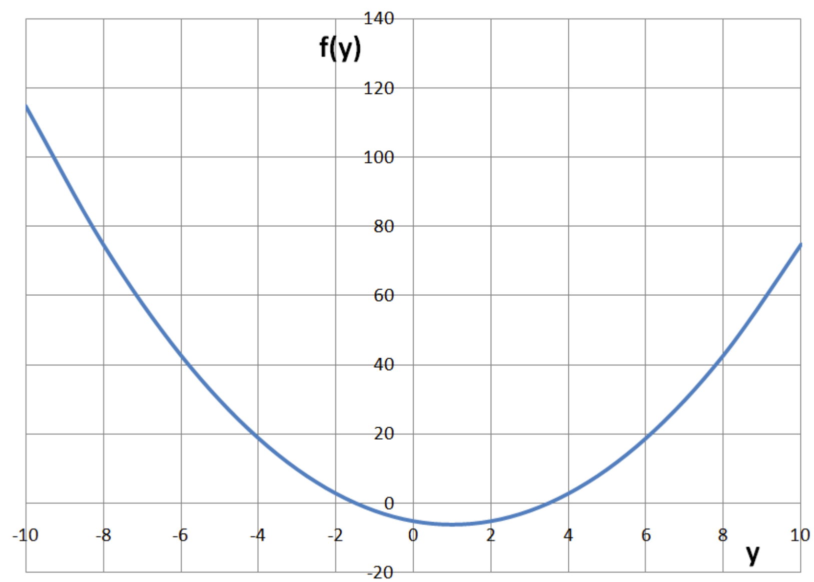
Bisection Method Example - Polynomial

- If limits of -10 to 10 are selected, which root is found?
- In this case $f(-10)$ and $f(10)$ are both positive, and $f(0)$ is negative



Bisection Method Example - Polynomial

- Which half of the interval is kept?
- Depends on the algorithm used – in our example, if the function values for the lower limit and midpoint are of opposite signs, we keep the lower half of the interval



Bisection Method Example - Polynomial

- Therefore, we converge to the negative root

	A	B	C	D	E	F	G	H
1	Lower	Upper	Difference	< 2*error?	Mid	f(low)	f(mid)	Product
2	-10	10	20	NO	0	112	-8	-896
3	-10	0	10	NO	-5	112	27	3024
4	-5	0	5	NO	-2.5	27	3.25	87.75
5	-2.5	0	2.5	NO	-1.25	3.25	-3.9375	-12.7969
6	-2.5	-1.25	1.25	NO	-1.875	3.25	-0.73438	-2.38672
7	-2.5	-1.875	0.625	NO	-2.1875	3.25	1.160156	3.770508
8	-2.1875	-1.875	0.3125	NO	-2.03125	1.160156	0.188477	0.218662
9	-2.03125	-1.875	0.15625	NO	-1.95313	0.188477	-0.27905	-0.05259
10	-2.03125	-1.95313	0.078125	NO	-1.99219	0.188477	-0.04681	-0.00882
11	-2.03125	-1.99219	0.039063	NO	-2.01172	0.188477	0.07045	0.013278
12	-2.01172	-1.99219	0.019531	NO	-2.00195	0.07045	0.011723	0.000826
13	-2.00195	-1.99219	0.009766	NO	-1.99707	0.011723	-0.01757	-0.00021
14	-2.00195	-1.99707	0.004883	NO	-1.99951	0.011723	-0.00293	-3.4E-05
15	-2.00195	-1.99951	0.002441	NO	-2.00073	0.011723	0.004395	5.15E-05
16	-2.00073	-1.99951	0.001221	NO	-2.00012	0.004395	0.000732	3.22E-06
17	-2.00012	-1.99951	0.00061	NO	-1.99982	0.000732	-0.0011	-8E-07
18	-2.00012	-1.99982	0.000305	NO	-1.99997	0.000732	-0.00018	-1.3E-07
19	-2.00012	-1.99997	0.000153	NO	-2.00005	0.000732	0.000275	2.01E-07
20	-2.00005	-1.99997	7.63E-05	YES	-2.00001	0.000275	4.58E-05	1.26E-08

Bisection Method – Summary

- A “bracketing” method
- Guaranteed to converge
- May take a long time to converge if root is near one of the bounds
- Provides guaranteed upper bound on the absolute (not relative!) error

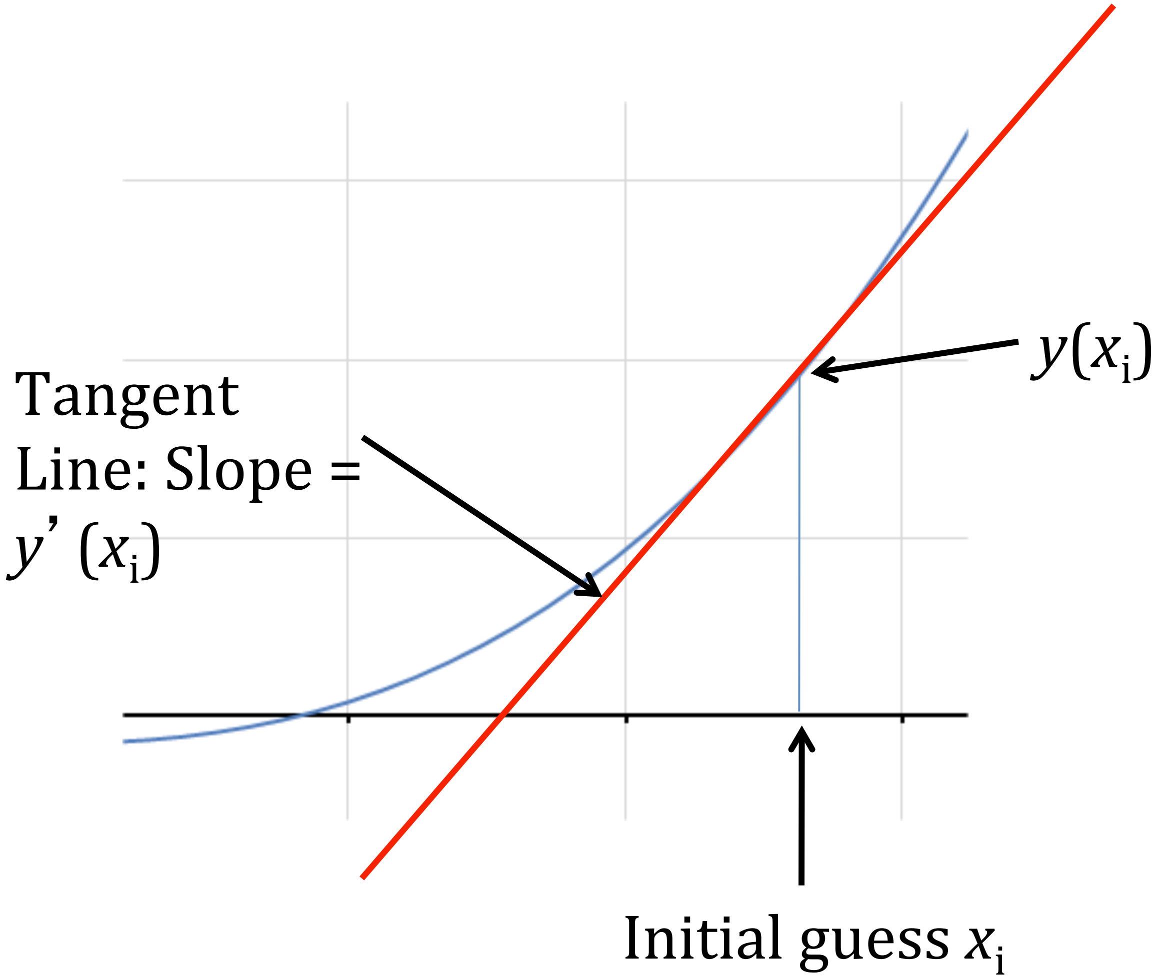
“Open” Root-finding Methods

- Bisection method requires two input x-values for which the sign of $f(x)$ differs
  - They thus bracket the root
  - These are called “closed” methods
- “Open” methods do *not* require knowledge of where the root lies
  - Some only need one initial guess
  - BUT they may not converge

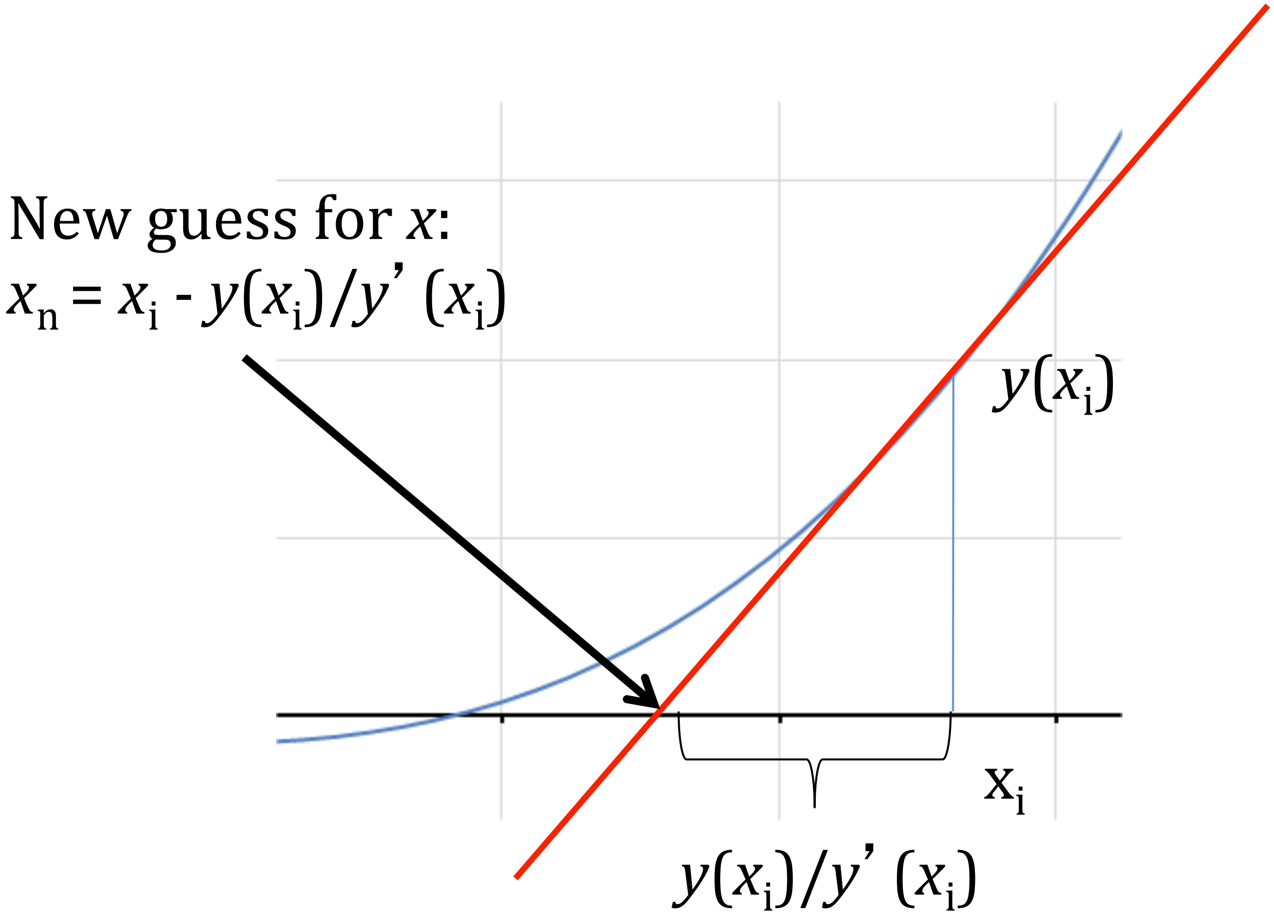
Newton's Method

- Newton's Method (also know as the Newton-Raphson Method) is another widely-used algorithm for finding roots of equations
- In this method, the slope (derivative) of the function is calculated at the initial guess value and projected to the *x*-axis
- The corresponding *x*-value becomes the new guess value
- The steps are repeated until the answer is obtained to a specified tolerance

Newton's Method



Newton's Method



Newton's Method Example

- Find a root of this equation:

$$y = 3x^3 - 15x^2 - 20x + 50$$

- The first derivative is:

$$y' = 9x^2 - 30x - 20$$

- Initial guess value: $x = 10$

Newton's Method Example

- For x = 10:

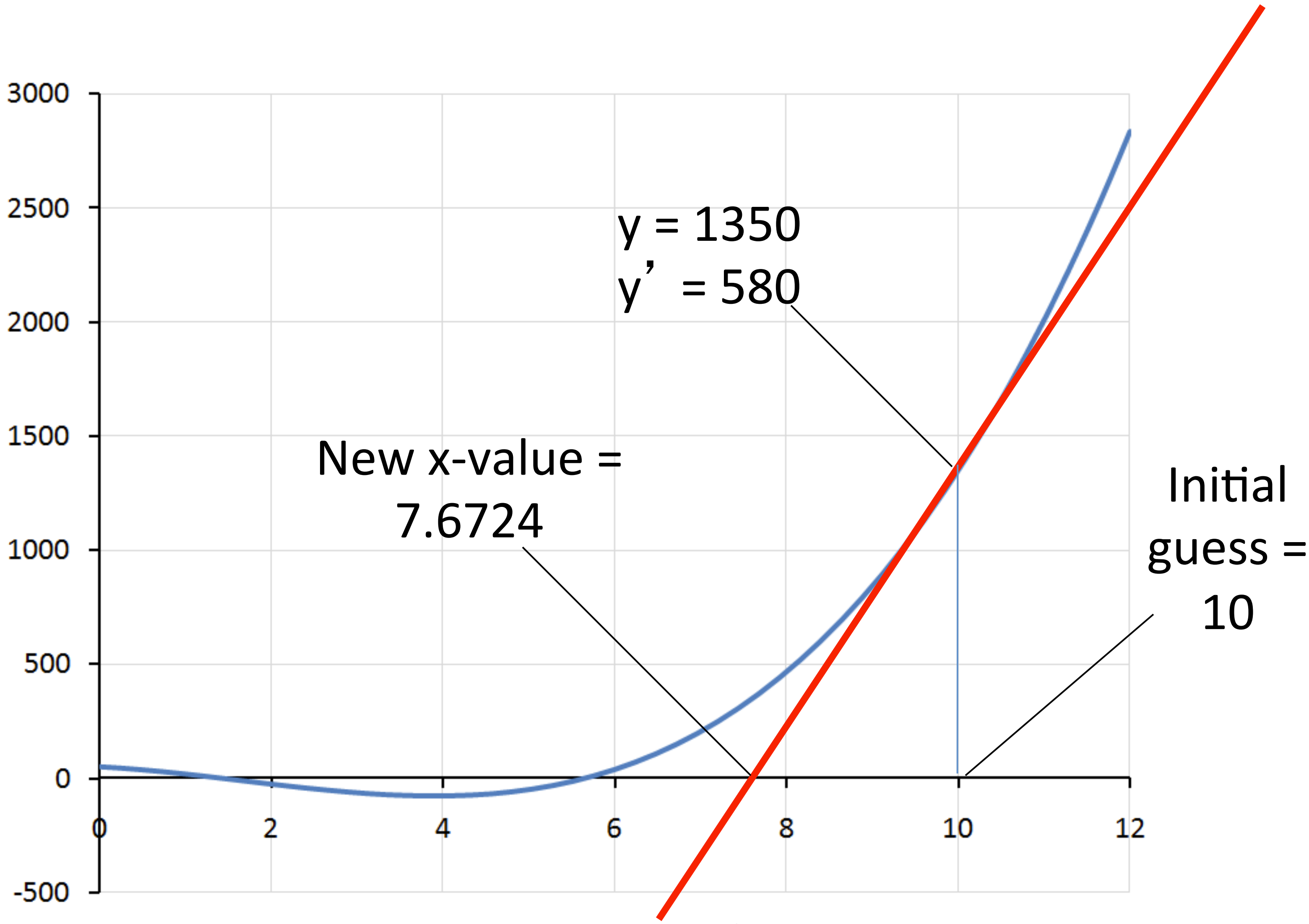
$$y = 3(10)^3 - 15(10)^2 - 20(10) + 50 = 1350$$

$$y' = 9(10)^2 - 30(10) - 20 = 580$$

$$x_n = 10 - {}^{1350}\!/_{580} = 7.6724$$

- This is the new value of x

Newton's Method Example



Newton's Method Example

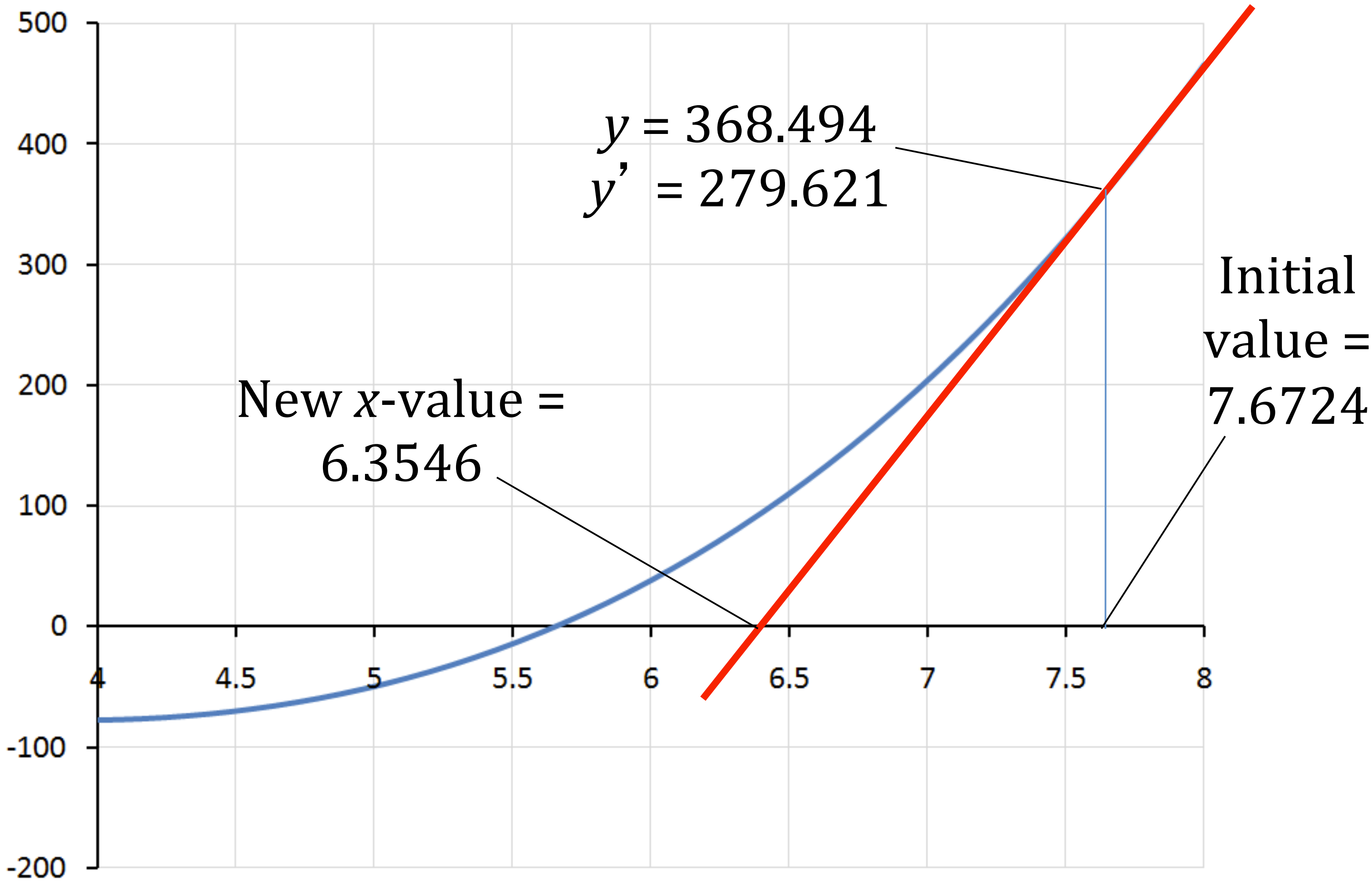
- For $x = 7.6724$:

$$y = 3(7.6724)^3 - 15(7.6724)^2 - 20(7.6724) + 50 = 368.494$$

$$y' = 9(7.6724)^2 - 30(7.6724) - 20 = 279.621$$

$$x_n = 7.6724 - {}^{368.494}/_{279.621} = 6.3546$$

Newton's Method Example



Newton’s Method Example

- Continue iterations:

x-initial	y(x)	y'(x)	x-new
10.0000	1350.0000	580.0000	7.6724
7.6724	368.4941	279.6210	6.3546
6.3546	87.0049	152.7887	5.7851
5.7851	13.1273	107.6559	5.6632
5.6632	0.5457	98.7502	5.6577
5.6577	0.0011	98.3529	5.6577
5.6577	0.0000	98.3521	5.6577

Method quickly converges to this root

Newton's Method Example

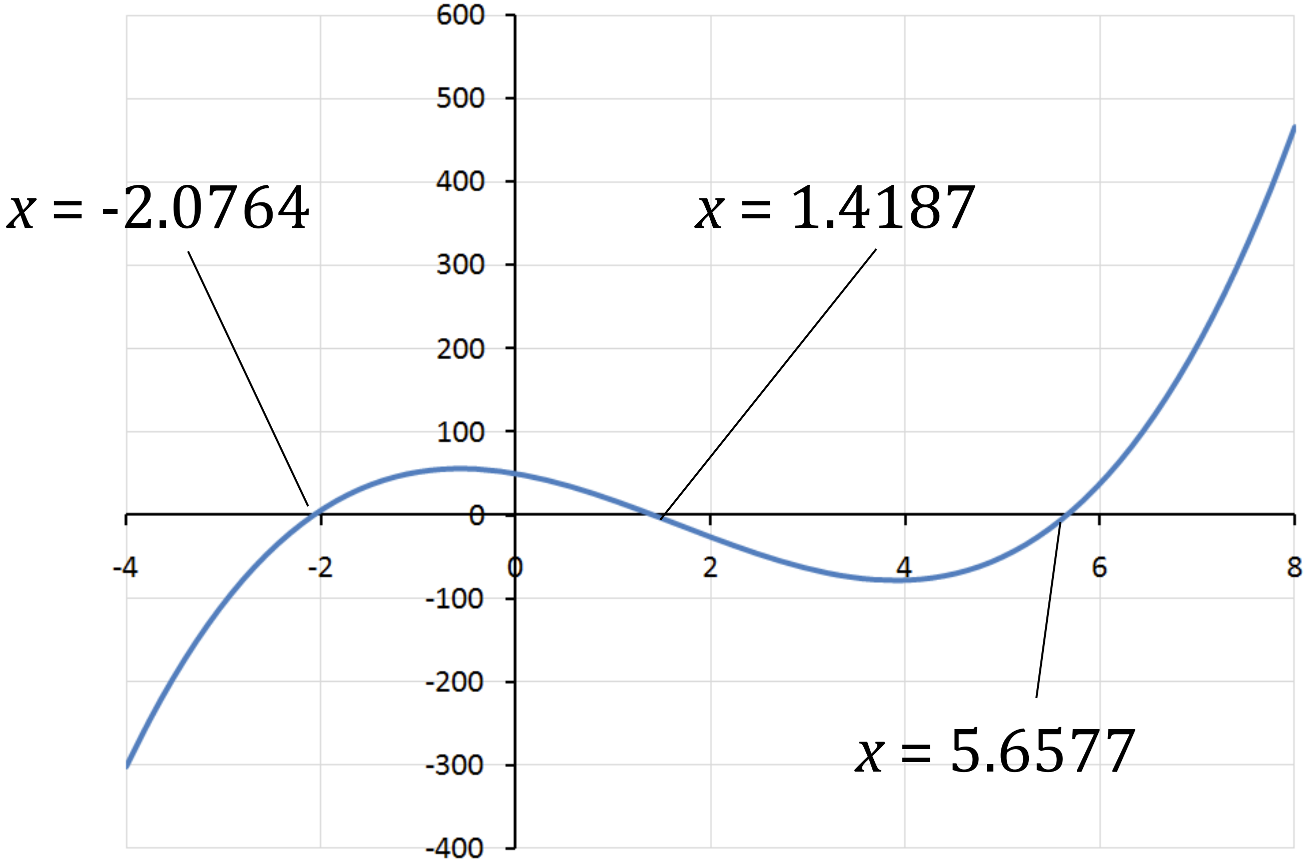
- To find the other roots, use different initial guess values

x-initial	y(x)	y'(x)	x-new
0.0000	50.0000	-20.0000	2.5000
2.5000	-46.8750	-38.7500	1.2903
1.2903	5.6645	-43.7253	1.4199
1.4199	-0.0503	-44.4518	1.4187
1.4187	0.0000	-44.4468	1.4187

x-initial	y(x)	y'(x)	x-new
-5.0000	-600.0000	355.0000	-3.3099
-3.3099	-156.9105	177.8923	-2.4278
-2.4278	-32.7877	105.8823	-2.1181
-2.1181	-3.4445	83.9231	-2.0771
-2.0771	-0.0572	81.1421	-2.0764
-2.0764	0.0000	81.0946	-2.0764

Newton's Method Example

- Three roots found:



Newton's Method - Comments

- Usually converges to a root much faster than the bisection method
- In some cases, the method will not converge (discontinuous derivative, initial slope = 0, etc.)
- In most cases, however, if the initial guess is relatively close to the actual root, the method will converge
- Don't necessarily need to calculate the derivative: can approximate the slope from two points close to the *x*-value. This is called the *Secant Method*

Newton-Raphson Method

- Advantages:
  - Fast convergence
  - Error on i+1st iteration proportional to square of error on ith iteration
    - Number of correct decimal places roughly doubles each iteration (assuming convergence!)
- Disadvantages:
  - Derivative must be known
  - No general convergence criterion
- Implementation suggestions:
  - Include a limit on number of iterations
  - Check for $f'(x) = 0$ during computation

MATLAB’s Built-in Root-finding Tools

fzero()

- root = fzero(@func,[low,high]);
- root = fzero(@func,guess);
- Uses a combination of methods

  (See the help under “finding roots”)

roots()

- – Solves for all roots of a polynomial

roots Example

- For polynomials, the MATLAB function *roots* finds all of the roots, including complex roots
- The argument of *roots* is an array containing the coefficients of the equation
- For example, for the equation

$$y = 3x^3 - 15x^2 - 20x + 50$$

the coefficient array is [3, -15, -20, 50]<

roots Example

```
>> A = [3, -15, -20, 50];
>> roots(A)
ans =
     5.6577
    -2.0764
     1.4187
```

roots Example

- Now find roots of

$$y = 3x^3 - 5x^2 - 20x + 50$$

```
>> B = [3, -5, -20, 50];
>> roots(B)
ans =
  -2.8120
   2.2393 + 0.9553i
   2.2393 - 0.9553i
```

$\left.\begin{array}{l} \\ \end{array}\right\}$ Two complex roots

5. Lab Quiz – What to Expect

- One problem will involve correcting a script
  - it will likely involve fprintf(), logic, iteration
- One problem will involve writing a function from scratch
  - it will involve computing with arrays
  - it will involve iteration (while- and/or for-loops)

Example

Here's a function that is supposed to return a vector whose elements are the differences between adjacent elements in the input vector. Correct it:

```
function x = vecdiff(invec)
    len = size(invec);
    for i=1:len
        x(i) = invec(i) – invec(i+1);
    end
end
```

Example, cont.

- For input: [0, 3, -20, 8, 10, 7]
- Correct return value is: [3, -23, 28, 2, -3]

```
function x = vecdiff(invec)
    len = size(invec);
    for i=1:len
        x(i) = invec(i) – invec(i+1);
    end
end
```

Example

- Write a function that takes a 2-dimensional matrix and returns the maximum difference between any two elements