

Lecture 9

Finding the Roots of $f(x) = 0$: Newton and Secant

L. Olson

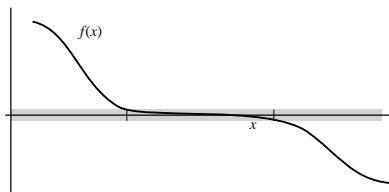
Department of Computer Science
University of Illinois at Urbana-Champaign

Slides based on NMM slides from Recktenwald

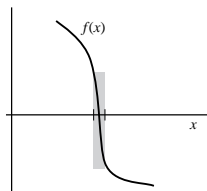
February 14, 2006

Convergence Criteria on $f(x)$

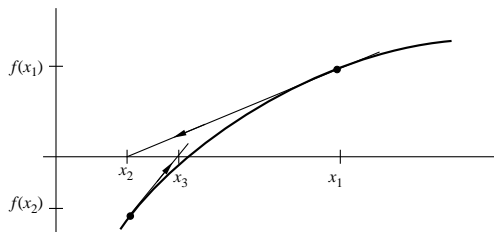
If $f'(x)$ is small near the root, it is easy to satisfy tolerance on $f(x)$ for a large range of Δx . The tolerance on Δx is more conservative



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



Newton's Method (1)



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 (2)

Use the slope at x_k , which is given by $f'(x_k)$ to get the next approximation to the root: x_{k+1} :

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

Thus

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

Newton's Method Algorithm

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

Newton's Method Example

Find the root of $f(x) = e^{-x} - x$ employing an initial guess of $x_0 = 0$.

First derivative is

$$f'(x) = -e^{-x} - 1$$

The iteration formula is

$$x_{k+1} = x_k - \frac{e^{-x_k} - x_k}{-e^{-x_k} - 1}$$

Newton's Method Example (1)

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 (2)

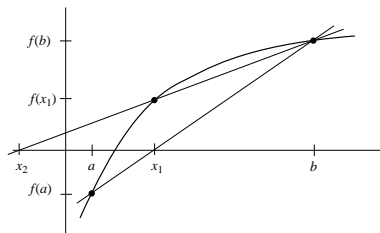
$$x_{k+1} = x_k - \frac{x_k - x_k^{1/3} - 2}{1 - \frac{1}{3}x_k^{-2/3}}$$

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.

Secant Method (1)



Given two guesses x_{k-1} and x_k , the next guess at the root is where the line through $f(x_{k-1})$ and $f(x_k)$ crosses the x axis.

Secant Method (2)

Given

x_k = current guess at the root

x_{k-1} = previous guess at the root

Approximate the first derivative with

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

Substitute approximate $f'(x_k)$ into formula for Newton's method

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

to get

$$x_{k+1} = x_k - f(x_k) \left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right]$$

Secant Method (3)

Two versions of this formula are (equivalent in exact math)

$$x_{k+1} = x_k - f(x_k) \left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right] \quad (\star)$$

and

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

Equation (\star) is better since it is of the form $x_{k+1} = x_k + \Delta$. Even if Δ is inaccurate the change in the estimate of the root will be small at convergence because $f(x_k)$ will also be small.

Equation $(\star\star)$ is susceptible to catastrophic cancellation:

- $f(x_k) \rightarrow f(x_{k-1})$ as convergence approaches, so cancellation error in denominator can be large.
- $|f(x)| \rightarrow 0$ as convergence approaches, so underflow is possible

Secant Algorithm

```
1 initialize:  $x_1 = \dots, x_2 = \dots$   
2 for  $k = 2, 3, \dots$   
3    $x_{k+1} = x_k - f(x_k)(x_k - x_{k-1}) / (f(x_k) - f(x_{k-1}))$   
4   if converged, stop  
5 end
```

Secant Example

Solve:

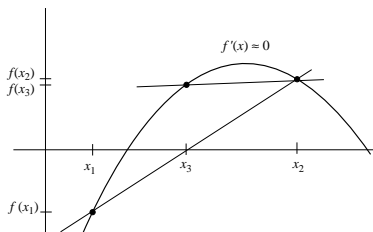
$$x - x^{1/3} - 2 = 0$$

k	x_{k-1}	x_k	$f(x_k)$
0	4	3	-0.44224957
1	3	3.51734262	-0.00345547
2	3.51734262	3.52141665	0.00003163
3	3.52141665	3.52137970	-2.034×10^{-9}
4	3.52137959	3.52137971	-1.332×10^{-15}
5	3.52137971	3.52137971	0.0

Conclusions:

- Converges almost as quickly as Newton's method.
- There is no need to compute $f'(x)$.
- The algorithm is simple.
- Two initial guesses are necessary
- Iterations are not guaranteed to stay inside an ordinal bracket.

Divergence of Secant Method



Since

$$x_{k+1} = x_k - f(x_k) \left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right]$$

the new guess, x_{k+1} , will be far from the old guess whenever $f'(x_k) \approx f(x_{k-1})$ and $|f(x)|$ is not small.

Summary

- Plot $f(x)$ before searching for roots
- Bracketing finds coarse interval containing roots and singularities
- Bisection is robust, but converges slowly
- Newton's Method
 - ▶ Requires $f(x)$ and $f'(x)$.
 - ▶ Iterates are not confined to initial bracket.
 - ▶ Converges rapidly.
 - ▶ Diverges if $f'(x) \approx 0$ is encountered.
- Secant Method
 - ▶ Uses $f(x)$ values to approximate $f'(x)$.
 - ▶ Iterates are not confined to initial bracket.
 - ▶ Converges almost as rapidly as Newton's method.
 - ▶ Diverges if $f'(x) \approx 0$ is encountered.

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton**
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant**
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation**
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 **When adding two positive numbers $a + b$, we should be careful when _____**
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____**
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 8 What is the relative error in the last iterate?
 - 9 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 1 What is the relative error in the last iterate?
 - 2 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 1 What is the relative error in the last iterate?
 - 2 What is the relative approximate error in the last iterate?

More Questions

- 1 Name one advantage of Newton over Secant
- 2 Name one advantage of Secant over Newton
- 3 Name one advantage of bisection over Newton and Secant
- 4 Name two sources of error in our computation
- 5 When adding two positive numbers $a + b$, we should be careful when _____
- 6 When subtracting two positive numbers $a - b$, we should be careful when _____
- 7 x_k is a sequence of iterates to approximate x . Suppose the sequence x_k is 2.92, 2.71, 2.58, 2.4, 2.2 and that $x_k = 2$.
 - 1 What is the relative error in the last iterate?
 - 2 What is the relative approximate error in the last iterate?

fzero Function (1)

fzero is a hybrid method that combines bisection, secant and reverse quadratic interpolation

Syntax:

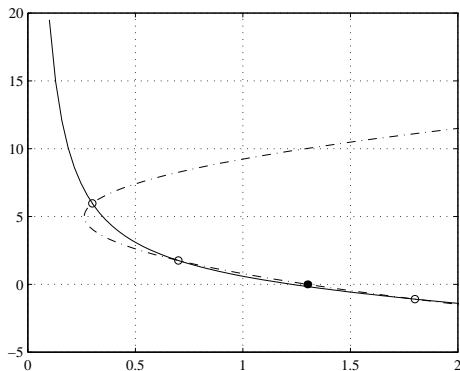
```
1 r = fzero('fun', x0)
2 r = fzero('fun', x0, options)
3 r = fzero('fun', x0, options, arg1, arg2, \ldots)
```

x_0 can be a scalar or a two element vector

- If x_0 is a scalar, **fzero** tries to create its own bracket.
- If x_0 is a two element vector, **fzero** uses the vector as a bracket.

Reverse Quadratic Interpolation

Find the point where the x axis intersects the sideways parabola passing through three pairs of $(x, f(x))$ values.



fzero Function (2)

fzero chooses next root as

- Result of reverse quadratic interpolation (RQI) if that result is inside the current bracket.
- Result of secant step if RQI fails, and if the result of secant method is inside the current bracket.
- Result of bisection step if both RQI and secant method fail to produce guesses inside the current bracket.

fzero Function (3)

Optional parameters to control **fzero** are specified with the **optimset** function.

Examples:

Tell **fzero** to display the results of each step:

```
1 >> options = optimset('Display','iter');  
2 >> x = fzero('myFun',x0,options)
```

Tell **fzero** to use a relative tolerance of 5×10^{-9} :

```
1 >> options = optimset('TolX',5e-9);  
2 >> x = fzero('myFun',x0,options)
```

Tell **fzero** to suppress all printed output, and use a relative tolerance of 5×10^{-4} :

```
1 >> options = optimset('Display','off','TolX',5e-4);  
2 >> x = fzero('myFun',x0,options)
```

fzero Function (4)

Allowable options (specified via optimset):

Option type	Value	Effect
'Display'	'iter'	Show results of each iteration
	'final'	Show root and original bracket
	'off'	Suppress all print out
'TolX'	tol	Iterate until $ \Delta x < \max[\text{tol}, \text{tol} * a, \text{tol} * b]$ where $\Delta x = (b - a)/2$, and $[a, b]$ is the current bracket.

The default values of 'Display' and 'TolX' are equivalent to

```
options = optimset('Display','iter','TolX',eps)
```

fzero example

Take

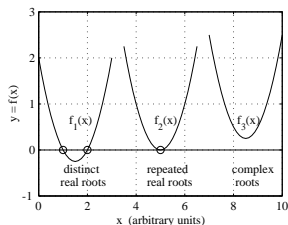
$$f(x) = x^{10} - 1$$

```
1 >> f = @(x)x.^10 - 1;  
2 >> options = optimset('display','iter');  
3 >> [x,fx]=fzero(f,0.5,options)
```

Roots of Polynomials

Complications arise due to

- Repeated roots
- Complex roots
- Sensitivity of roots to small perturbations in the polynomial coefficients (conditioning).



Algorithms for Finding Polynomial Roots

- Bairstow's method
- Müller's method
- Laguerre's method
- Jenkin's–Traub method
- Companion matrix method

roots Function (1)

The built-in **roots** function uses the companion matrix method

- No initial guess
- Returns *all* roots of the polynomial
- Solves eigenvalue problem for companion matrix

Write polynomial in the form

$$c_1x^n + c_2x^{n-1} + \dots + c_nx + c_{n+1} = 0$$

Then, for a *third* order polynomial

```
1 >> c = [c1 c2 c3 c4];  
2 >> r = roots(c)
```

roots Function (2)

The eigenvalues of

$$A = \begin{bmatrix} -c_2/c_1 & -c_3/c_1 & -c_4/c_1 & -c_5/c_1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

are the same as the roots of

$$c_5\lambda^4 + c_4\lambda^3 + c_3\lambda^2 + c_2\lambda + c_1 = 0.$$

The statements

```
1 c = ... % vector of polynomial coefficients
2 r = roots(c);
```

are equivalent to

```
1 c = ...
2 n = length(c);
3 A = diag(ones(1,n-2), -1); % ones on first subdiagonal
4 A(1,:) = -c(2:n) ./ c(1); % first row is -c(j)/c(1), j=2..n
5 r = eig(A);
```

roots Examples

Roots of

$$f_1(x) = x^2 - 3x + 2$$

$$f_2(x) = x^2 - 10x + 25$$

$$f_3(x) = x^2 - 17x + 72.5$$

are found with

```
1 >> roots([1 -3 2])
2 ans =
3     2
4     1
5
6 >> roots([1 -10 25])
7 ans =
8     5
9     5
10
11 >> roots([1 -17 72.5])
12 ans =
13     8.5000 + 0.5000i
14     8.5000 - 0.5000i
```