Rootfinding: Secant Method

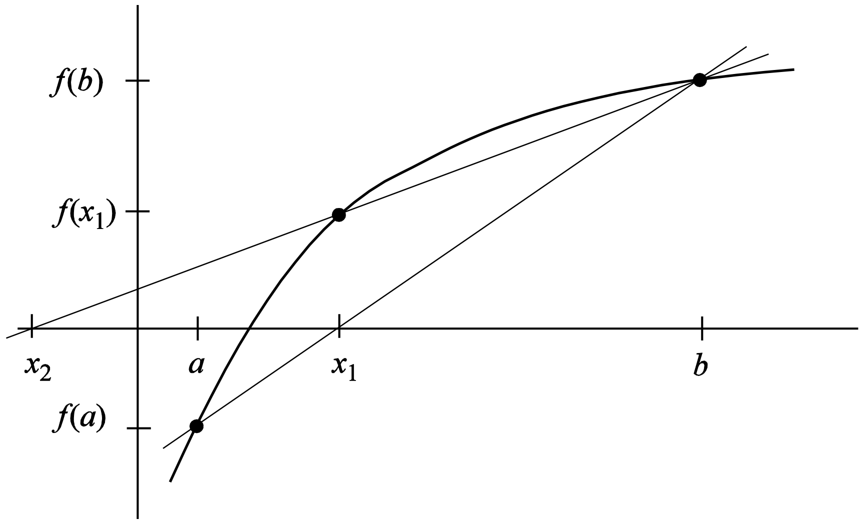
Lecture 12

October 8, 2013

David Semeraro

Secant Method

- 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

Given:

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

Approximate the first derivative with:

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

Substitute $f'(x_k)$ into Newton's method.

Secant Method

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

Becomes:

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

Secant Method

Two versions of formula are the equivalent in exact math:

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

Which is better computationally?

- Cancellation?

Secant Algorithm

Initialize $x_1 = *, x_2 = *$

for $k = 2,3,\ldots$

$x_{k+1} = x_k - f(x_k)\,(x_k - x_{k-1})/\big(f(x_k) - f(x_{k-1})\big)$

if converged stop

end

Secant Example

Solve:

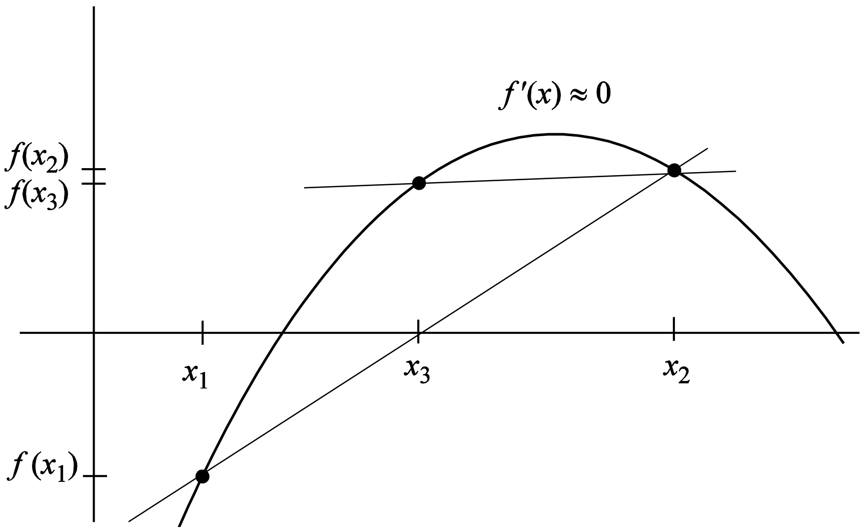
$$x - x^{\frac{1}{3}} - 2 = 0$$

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

Secant Example

- Conclusions:
  - Converges almost as quickly as Newton's method ($r = 1.62$).
  - 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 $|f(x_k)|$ 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 ($r = 2$).
  - Diverges if $f'(x) = 0$ 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 ($r = 1.62$).
  - Diverges if $f'(x) = 0$ is encountered.

Systems of Equations

- Single valued function of 1 variable.
- What about higher dimensions?

$$
\begin{gathered}
f_1(x_1, x_2, \cdots, x_n) = 0 \\
f_2(x_1, x_2, \cdots, x_n) = 0 \\
\vdots \\
f_n(x_1, x_2, \cdots, x_n) = 0
\end{gathered}
$$

Systems of Equations

$$F(X) = 0$$

Where

$$F = [f_1, f_2, \ldots, f_n]^T$$
$$X = [x_1, x_2, \ldots, x_n]^T$$

Newton's method becomes:

$$X^{(k+1)} = X^{(k)} - \left[F'\left(X^{(k)}\right)\right]^{-1} F\left(X^{(k)}\right)$$

Systems of Equations

- $F'\left(x^{(k)}\right)$ is the Jacobian Matrix.
- Made up of the partial derivatives of F evaluated at $X^{(k)}$
- $X^{(0)}$ is the initial solution *vector*
- The inverse of the Jacobian Matrix is not computed but rather the related system of equations solved.

Example

- 3 equations in 3 variables

$$f_1(x_1, x_2, x_3) = 0$$
$$f_2(x_1, x_2, x_3) = 0$$
$$f_3(x_1, x_2, x_3) = 0$$

- Use Taylor series expansion

$$f_i(x_1 + h_1, x_2 + h_2, x_3 + h_3)$$
$$= f_i(x_1, x_2, x_3) + h_1 \frac{\partial f_i}{\partial x_1} + h_2 \frac{\partial f_i}{\partial x_2} + h_3 \frac{\partial f_i}{x_3}$$
$$+ \cdots$$

Example

- Let $X^{(0)} = \left[x_1{}^{(0)}, x_2{}^{(0)}, x_3{}^{(0)}\right]^T$ be the initial approximate solution.
- Let $H = [h_1, h_2, h_3]^T$ be a correction vector such that $X^{(0)} + H$ is a better approximate solution.
- Discard the higher order terms in the Taylor expansion and …

Example

$$0 \approx F\left(X^{(0)} + H\right) \approx F\left(X^{(0)}\right) + F'\left(X^{(0)}\right)H$$

Where:

$$F'\left(X^{(0)}\right) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}$$

Example

- Assume $F'\left(X^{(0)}\right)$ is nonsingular.
- Solving for $H = -\left[F'\left(X^{(0)}\right)\right]^{-1} F\left(X^{(0)}\right)$
- $X^{(1)} = X^{(0)} + H$ is a better approximation.
- In general

$$X^{(k+1)} = X^{(k)} - \left[F'\left(X^{(k)}\right)\right]^{-1} F\left(X^{(k)}\right)$$

Example

1. Solve

$$\left[F'(X^{(k)})\right]H^{(k)} = -F\left(X^{(k)}\right)$$

2. Update

$$X^{(k+1)} = X^{(k)} + H^{(k)}$$

Numerical Example

- $f_1(x, y) = 2x^2 + 3x - 4 - y = 0$
- $f_2(x, y) = x^2 + 2x + 3 - y = 0$

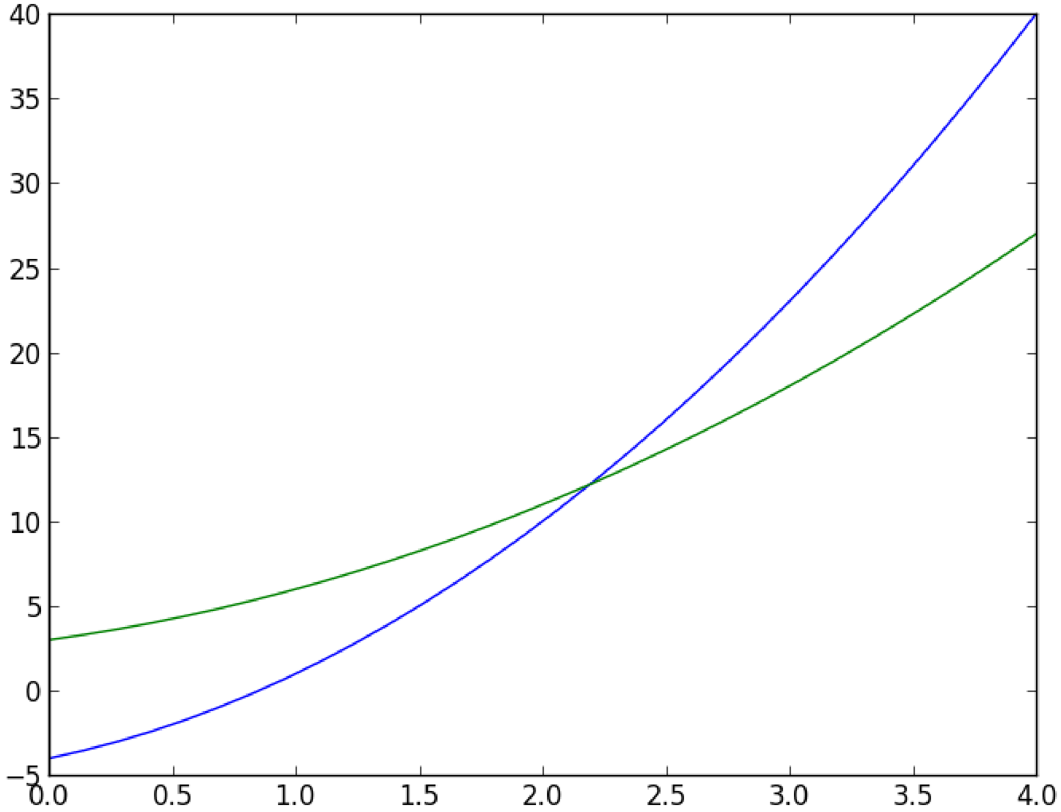
$$F'(x) = \begin{bmatrix} 4x + 3, & -1 \\ 2x + 2, & -1 \end{bmatrix}$$

Numerical Example

- $x_0 = 1, y_0 = 1$

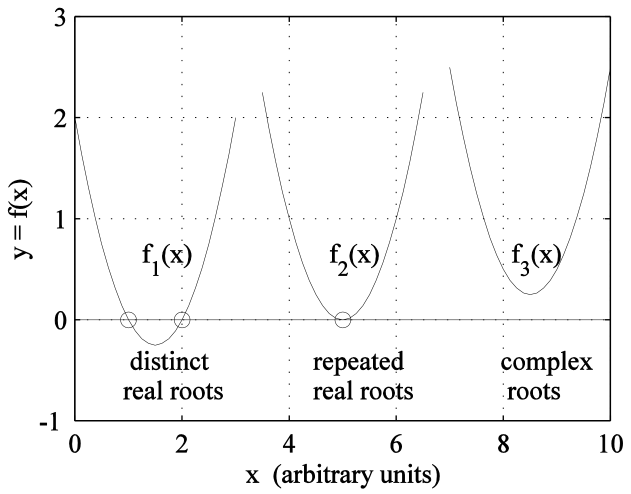
k	x	y	f_1	f_2	Δx	Δy
0	1.0	1.0	-0.0	-5.0	1.6667	11.6667
1	2.6667	12.6667	-5.5556	-2.7778	-0.4386	-0.4386
2	2.2281	12.2281	-0.3847	-0.1924	-0.3526	-0.3526
3	2.1928	12.1928	-0.0025	-0.0013	-0.0002	-0.0002
4	2.1926	12.1926	-1.06e-07	-5.33e-08	-9.89e-09	-9.89e-09

Numerical Example



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

The built-in roots function in numpy 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:

$$c_0 x^n + c_1 x^{n-1} + \cdots + c_{n-1} x + c_n = 0$$

Then for a 3rd order polynomial:

```
>>> coeff = [3.2, 2, 1]
>>> np.roots(coeff)
array([-0.3125+0.46351241j, -0.3125-0.46351241j])
```

What is a Companion Matrix

Eigenvalues of:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -c_0 & -c_1 & -c_2 & -c_3 \end{bmatrix}$$

Are the same as the roots of:

$$\lambda^4 + c_3\lambda^3 + c_2\lambda^2 + c_1\lambda + c_0 = 0$$

Companion Matrix

To see this, recall $Ax = \lambda x$ is the equation satisfied by eigenvalues of A.

Write this as $(A - \lambda I)x = 0$.

Then the matrix

$$(A - \lambda I)x = \begin{bmatrix} -\lambda & 1 & 0 & 0 \\ 0 & -\lambda & 1 & 0 \\ 0 & 0 & -\lambda & 1 \\ -c_0 & -c_1 & -c_2 & -\lambda - c_3 \end{bmatrix}$$

is singular and the determinant is zero. Because most of the elements are zero the determinant can be computed.

Example

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

Companion Matrix:

$$\begin{bmatrix} 0 & 1 \\ -25 & 10 \end{bmatrix}$$

```
>>> coef = [1.0,-10.0,25.0]
>>> np.roots(coef)
array([ 5.,  5.])
>>> coef = [-25.0,10.0];
>>> A = scipy.array([[0.,1.],coef]);
>>> scipy.linalg.eig(A,right=False);
array([ 5.+0.j,  5.+0.j])
```

Fractals

- Fractal: A mathematical pattern (geometric object) that is reproducible at any level of magnification or reduction.
- Fractal: A term used by Benoit Mandelbrot to refer to geometric objects with fractional dimensions rather than integer dimensions. Also used ”fractal” to refer to shapes that are self-similar: they look the same at any zoom level.

Fractals

Scientifically used to describe highly irregular objects

- fractal image compression
- Seismology
- Cosmology
- life sciences:
  - clouds and fluid turbulence
  - trees
  - coastlines
- More interesting observations:
  - New music/New art
  - Video games/graphics
  - Chaos theory
  - the Butterfly effect: small changes produces large effects

Fractal Generated Terrain

Fractals

Recall Complex Numbers: $z \in C$ means

$$z = x + iy$$

where $i = \sqrt{-1}$

- Things to notice:
  - still think of the x-y plane, but now it's in C^1 instead of R^2
  - $f(z) = z^2 + 1$ has 2 roots $z_{1,2} = \pm i$
  - $f(z) = z^3 + 1$ has 3 roots $z_1 = 1, z_{2,3} = \frac{-1 \pm i\sqrt{3}}{2}$
  - $f(z) = z^4 + 1$ has 4 roots $z_{1,2} = -1, z_{3,4} = \pm i$

Fractals

- Take a complex function like $f(z) = z^3 + 1$
- Pick a bunch of initial guesses z_0 as the roots
- Run Newton's Method
- The initial guesses z_0 will each converge to one of $n = 3$ roots
- Color each guess in the plane depending on the root to which it converged.

Fractals