

Numerical Methods - Finding Solutions of Nonlinear Equations

Y. K. Goh

Universiti Tunku Abdul Rahman

2013

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 Iterative/Open Methods
 - Fixed-point iteration
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Problem setting

For a given function $f(x)$ find a value of $x = x_0$ such that $f(x_0) = 0$. x_0 is called the root of $f(x)$

Example

- Some roots can be found **explicitly**: the roots of a quadratic polynomial $f(x) = ax^2 + bx + c$ are given by the formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Example

- However for most engineering problems, roots can be only be expressed **implicitly**. For example, there is no simple formula to solve $f(x) = 0$, where $f(x) = 2^{x^2} - x + 7$ or $f(x) = x^2 - 3 \sin(x) + 2$.

Numerical root finding algorithms are for solving nonlinear equations.

Example

An electric power cable is suspended from two equal height towers that are 100 meters apart. The cable is allowed to dip 10 meters in the middle. How long is the cable?

Answer: This is a **catenary** problem and the length of the cable ℓ is given by the solution to the following equation:

$$\ell \cosh\left(\frac{50}{\ell}\right) = \ell + 10.$$

Example

The **internal rate of return** (*IRR*) of an investment is the annualised effective compound return rate that sets the net present value (*NPV*) of all cash flows from the investment to be zero.

Find the *IRR* for an investment with just three cash flows -1,600, 10,000, and -10,000 at three different years.

Answer: The *IRR* is given by solving

$$-1600 + 10000(1 + IRR)^{-1} - 10000(1 + IRR)^{-2} = 0.$$

Example

Solve the following system of nonlinear equations:

$$\begin{aligned}x + y + z &= 3 \\x^2 + y^2 + z^2 &= 5 \\e^x + xy - xz &= 1\end{aligned}$$

Various Root Finding methods

Two families: bracketing or open methods.

- Bracketing methods: find $x \in [a, b]$ such that $f(x) = 0$.
 - Graphing
 - Bisection
 - False-position
- Open methods: given initial point(s) $x = x_0$, find x such that $f(x) = 0$.
 - Fixed-point iteration
 - Newton-Raphson method
 - Secant method
- **Bisection**, **Newton-Raphson**, and **Secant** methods are most popular methods.
- We will discuss the *algorithms*, *error analysis*, *convergence* for iterative methods and *acceleration of convergence*.

Outline

1 Motivation

2 Bracketing Methods

- Graphing
- Bisection
- False-position

3 Iterative/Open Methods

- Fixed-point iteration
- Newton-Raphson
- Secant method

4 Convergence Acceleration: Aitken's Δ^2 and Steffensen

5 Muller's Methods for Polynomials

6 System of Nonlinear Equations

1 Motivation

2 Bracketing Methods

- Graphing
- Bisection
- False-position

3 Iterative/Open Methods

- Fixed-point iteration
- Newton-Raphson
- Secant method

4 Convergence Acceleration: Aitken's Δ^2 and Steffensen

5 Muller's Methods for Polynomials

6 System of Nonlinear Equations

Graphing: The idea

- This is a primitive but useful method to give rough estimate on where the roots are.
- The main idea is to plot the graph of $f(x)$ for a sufficiently wide range of x and estimate the roots as the x -intersects.
- With the advance in graphing software, the hassle of plotting a graph becomes an easy task.
- When higher precision of the root is required, this method could serves as a starting point to suggest the initial points or bracket for the other methods.
- The estimated error ϵ of this method is $\frac{1}{2}(x_{n+1} - x_n)$, where $x_0, x_1, \dots$ are the points used for plotting the graph of $f(x)$ and $[x_n, x_{n+1}]$ is the bracket where the root is.

Example: Graphing

Example

Find ℓ in $\ell \cosh\left(\frac{50}{\ell}\right) = \ell + 10$.

Answer:

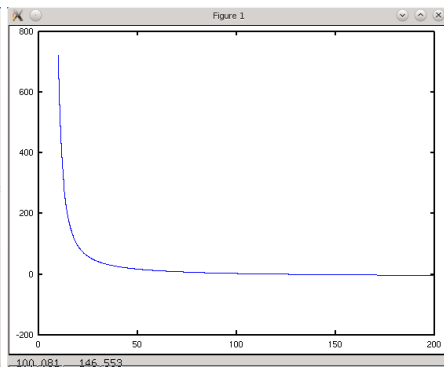
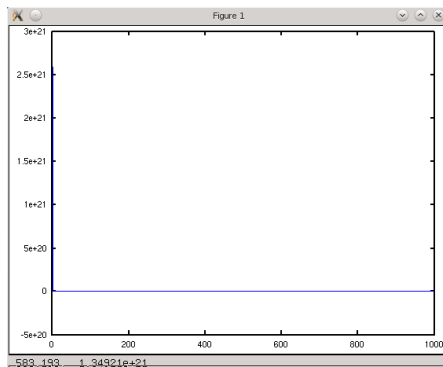
- Rewrite the equation into $f(x) = 0$, ie $f(x) = x \cosh(50/x) - x - 10$.
- Decide a sufficient wide range of the graph for $f(x)$, say $0 \leq x \leq 1000$.
- Decide how to partition the range, say partitioned into 1001 points, ie $x_0 = 0, x_1 = 1, \dots, x_{1000} = 1000$.
- Calculate $f(x_i)$ for $0 \leq i \leq 1000$. Plot x_i versus $f(x_i)$.
- Revise the range if necessary.
- Locate the root(s), ie, the intersect at x -axis.
- The estimated error(s) is $\frac{1}{2}(x_{n+1} - x_n)$.

Example: Graphing (Cont.)

Matlab

```
1 >> x = linspace(0, 1000, 1001);
2 >> f = x .* cosh(50./x) - x - 10;
3 >> plot(x,f);
4 >> % starting at x = 0 might not be a good idea
5 >> x = linspace(10, 200, 1001);
6 >> f = x .* cosh(50./x) - x - 10;
7 >> plot(x,f);
8 >> % scan through some values of f ...
9 >> f(612: 616)
0 >> % the graph crosses y-axis at i = 614 and i = 615
1 >> x(614:615)
2 >> est_error = (x(615) - x(614))/2.0
```

Example: Graphing (Cont.)



$f(612:616) = [0.0441419, 0.0286354, 0.0131774, -0.0022326, -0.0175946]$

$x(614:615) = [126.47, 126.66]$

$est_error = 0.09500$

1 Motivation

2 Bracketing Methods

- Graphing
- **Bisection**
- False-position

3 Iterative/Open Methods

- Fixed-point iteration
- Newton-Raphson
- Secant method

4 Convergence Acceleration: Aitken's Δ^2 and Steffensen

5 Muller's Methods for Polynomials

6 System of Nonlinear Equations

Bisection: The idea

Given f is continuous on $[a, b]$ and $f(a)$ and $f(b)$ have opposite signs, find x such that $f(x) = 0$.

- Also known as **Binary search** method.
- Instead of plotting out every points in graphing methods, the main idea of bisection method is to divide the interval into two equal size sub-intervals and choose the sub-interval that containing the root in every iteration.

The algorithm

- 1 check if the given bracket $[a, b]$ contains at least one root.
- 2 calculate the estimate for the root, $p_n = (a + b)/2$.
- 3 p_n divides $[a, b]$ into two new brackets $[a, p_n]$ and $[p_n, b]$.
- 4 check which of the bracket contains root, and reset the values for a and b .
- 5 repeat the process to an acceptable error tolerance.

Calculus theorem in action!

Theorem (Intermediate Value Theorem)

If $f \in C[a, b]$ and K is any number between $f(a)$ and $f(b)$, then there exists a number c in (a, b) such that $f(c) = K$. In particular, if $f(a)$ and $f(b)$ are opposite signs, then there exists a number p in (a, b) such that $f(p) = 0$.

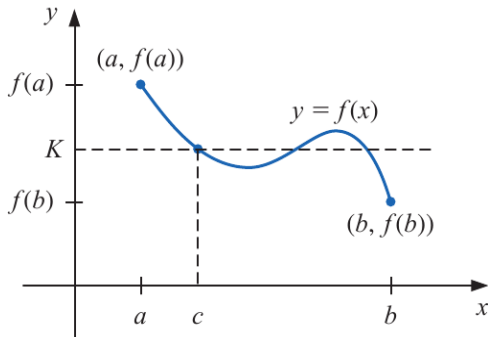
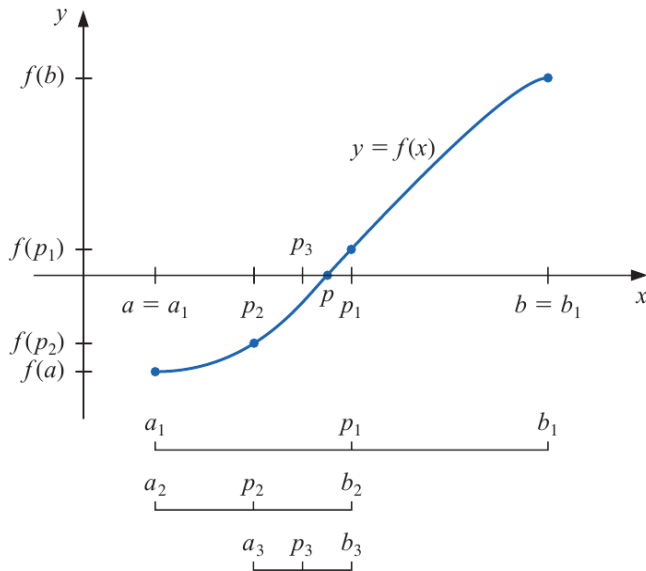


Figure: Illustration of Intermediate Value Theorem.

Illustration of bisection method



Example: Bisection

Example

Find the root for $\ell \cosh\left(\frac{50}{\ell}\right) - \ell - 10 = 0$ in the interval $[100, 200]$.

Answer:

- Check there is at least one root in $[100, 200]$, i.e. $f(100)f(200) < 0$.
- Decide an acceptable tolerance ϵ
- Set $a = 100$ and $b = 200$.
- Repeat the following until desirable precision:
 - 1 Estimate the root, $p_n = (a + b)/2$
 - 2 Calculate $f(p_n)$.
 - 3 If $f(p_n) = 0$, or $|f(p_n)| < \epsilon$ finished.
 - 4 If $f(p_n)f(a) < 0$, then reset $b = p_n$, else reset $a = p_n$.

Example: Bisection (Cont.)

Matlab

```
1 function ans = nm02_fun(x) % save as nm02_fun.m
2     ans = x*cosh(50/x) - x - 10;
```

Matlab

```
1 % FILE : nm02_bisec.m
2 a = 100; b = 200;
3 epsilon = 10^(-4)
4 while (1)
5     pn = (a+b)/2;
6     if abs( nm02_fun(pn) ) < epsilon
7         break
8     elseif nm02_fun(a) * nm02_fun(pn) < 0
9         b = pn;
10    else
11        a = pn;
12    end
13 end
```

Bisection Theorem

Theorem (Bisection Theorem)

Suppose $f \in C[a, b]$ and $f(a)f(b) < 0$. The bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of f with error $|p_n - p| \leq \frac{b-a}{2^n}$.

Definition (Linear Convergence)

A sequence p_n exhibits *linear convergence* to a limit p if there is a constant $k \in [0, 1)$ such that $|p_{n+1} - p| \leq k|p_n - p|$, for all $n \geq 1$.

- For linear convergent sequence: $|p_{n+1} - p| \leq k|p_n - p| \leq k^2|p_{n-1} - p| \leq \dots \leq k^n|p_1 - p| \implies |p_{n+1} - p| \leq k^n A$.
- Compare with the Bisection Theorem, $|p_n - p| \leq (\frac{1}{2})^n(b-a)$ indicates the approximations generated by Bisection methods converge linearly.

Example

Given the tolerance ϵ , we need at most $n > \log_2((b-a)/\epsilon)$ steps to achieve the desirable accuracy. If $\epsilon = 10^{-4}$ as in our example, we need at most $n \approx 20$ steps, but in actual case we only use $n = 15$.

1 Motivation

2 Bracketing Methods

- Graphing
- Bisection
- False-position

3 Iterative/Open Methods

- Fixed-point iteration
- Newton-Raphson
- Secant method

4 Convergence Acceleration: Aitken's Δ^2 and Steffensen

5 Muller's Methods for Polynomials

6 System of Nonlinear Equations

False-position: The idea

- This is a hybrid methods that combining the bracketing idea and secant method that you will learn later.

Outline

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 **Interative/Open Methods**
 - Fixed-point iteration
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 **Interative/Open Methods**
 - **Fixed-point iteration**
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Fixed-point iteration: The idea

The idea

- If we have an equation $f(x) = 0$, sometime we could re-write the equation into a self-consistent form $x = g(x)$; here we call x as a **fixed-point**.
- Then, we could setup an iterative scheme $x_{n+1} = g(x_n)$.
- The advantage of this scheme is easy to program, but the down side is the iteration is not always converge.
- The algorithm will not converge if $|g'(x)| \geq 1$, for all x in the interval that g is defined.

The algorithm

- 1 Write the equation into the form of $x_{n+1} = g(x_n)$.
- 2 Choose an initial value for x_0 .
- 3 Use x_0 to estimate x_1 , and then repeat the process to generate “better” approximations $x_0, x_1, x_2, \dots, x_n$ so that $|g(x_n) - x_n|$ is less than the pre-set tolerance.

Example: Fixed-point Iteration

Example

Rewrite $\ell \cosh\left(\frac{50}{\ell}\right) - \ell - 10 = 0$ into $\ell = g(\ell)$. Setup the iterative scheme and let $\ell_0 = 200$, estimate $\ell_i, i = 1, \dots, 100$.

Answer: [MATLAB code: nm02_fixed_point.m]

- define $x_{n+1} = x_n \cosh(50/x_n) - 10$.
- let $x_0 = 200$, then $x_1 = x_0 \cosh(50/x_0) - 10$.
- repeat until getting x_{100}

Note that of course the algorithm could be refined so that the loop stopped when a desirable accuracy is reached.

Example

Repeat the fixed point iteration for $g(x) = (10 + x)/\cosh(50/x)$. Is the iterations converge? What's going on? What is the value of $g'(200)$? [ANS: $g'(200) = 1.03$]

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 **Interative/Open Methods**
 - Fixed-point iteration
 - **Newton-Raphson**
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Newton-Raphson: The idea

The idea

- One of the drawback for fixed-point iteration is that the choices of $g(x)$ is arbitrary, and often not convergent.
- One way to choose a good is to choose $g(x) = x - f(x)/f'(x)$.

The algorithm

- 1 Choose an initial value for x_0 .
- 2 Setup the Newton-Raphson iterative scheme $x_{n+1} = x_n - f(x_n)/f'(x_n)$.
- 3 Use x_0 to estimate x_1 , and then repeat the process to generate “better” approximations $x_0, x_1, x_2, \dots, x_n$ until $|f(x_n)| < \epsilon$.

Newton-Raphson Method

Suppose $f \in C^2[a, b]$ and let $p_0 \in [a, b]$ be an approximation to the root p such that $f'(p_0) \neq 0$ and $|p_0 - p|$ is small. From Taylor's theorem,

$$f(p) = f(p_0) + f'(p_0)(p - p_0) + f''(\xi(p)) \frac{(p - p_0)^2}{2!},$$

assuming $(p - p_0)^2$ is negligible and we know $f(p) = 0$, we get

$$p \approx p_0 - \frac{f(p_0)}{f'(p_0)}.$$

or the iterative Newton-Raphson formula

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}. \quad (1)$$

Illustration of Newton-Raphson method

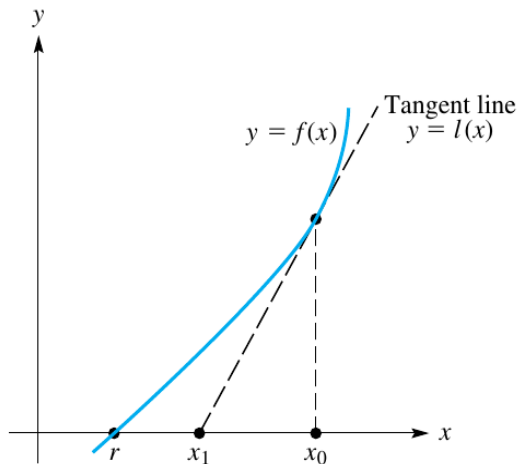


Figure: Illustration of Newton-Raphson method.

Example: Newton-Raphson

Example

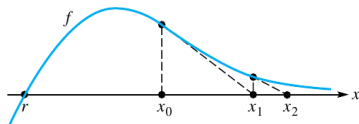
Solve $\ell \cosh\left(\frac{50}{\ell}\right) - \ell - 10 = 0$ with Newton-Raphson method and let $\ell_0 = 200$, estimate $\ell_i, i = 1, \dots, 10$.

Answer: [MATLAB code: nm02_newton.m]

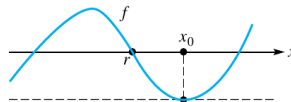
- define $f(x) = x \cosh(50/x) - x - 10$.
- get $f'(x) = \cosh(50/x) - \sinh(50/x) \frac{50}{x} - 1$.
- define $x_{n+1} = x_n - f(x_n)/f'(x_n)$.
- let $x_0 = 200$, then $x_1 = x_0 - f(x_0)/f'(x_0)$.
- repeat until getting x_{10}

Note that of course the algorithm could be refined so that the loop stopped when a desirable accuracy is reached.

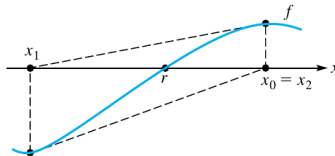
Newton-Raphson method will fail in the following



(a) Runaway



(b) Flat spot



(c) Cycle

Figure: Scenario when Newton-Raphson method fails.

Outline

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 **Interative/Open Methods**
 - Fixed-point iteration
 - Newton-Raphson
 - **Secant method**
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Secant method: The idea

The idea

- One of the drawback for Newton-Raphson method is it requires to calculate the derivative $f'(x_n)$, which sometimes is not straight forward.
- One way is to approximate $f'(x_n)$ as the slope of a secant connecting two close points. i.e. $f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$.

The algorithm

- 1 Choose an initial value for x_0 and x_1 .
- 2 Setup the Newton-Raphson iterative scheme
$$x_{n+1} = x_n - f(x_n)(x_n - x_{n-1})/(f(x_n) - f(x_{n-1})).$$
- 3 Use x_0 and x_1 to estimate x_2 , and then repeat the process to generate “better” approximations $x_0, x_1, x_2, \dots, x_n$ until $|f(x_n)| < \epsilon$.

Example: Secant

Example

Solve $\ell \cosh\left(\frac{50}{\ell}\right) - \ell - 10 = 0$ with secant method and let $\ell_0 = 200$ and $\ell_1 = 201$, estimate $\ell_i, i = 1, \dots, 10$.

Answer: [MATLAB code: nm02_secant.m]

- define $f(x) = x \cosh(50/x) - x - 10$.
- define $x_{n+1} = x_n - f(x_n) * (x_n - x_{n-1}) / (f(x_n) - f(x_{n-1}))$.
- let $x_0 = 200$ and $x_1 = 201$, and calculate x_2 from above formula.
- repeat until getting x_{10}

Note that of course the algorithm could be refined so that the loop stopped when a desirable accuracy is reached.

Outline

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 Iterative/Open Methods
 - Fixed-point iteration
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Order of Convergence

Definition (Order of Convergence)

Suppose $\{p_n\}_{n=0}^{\infty}$ is a sequence that converges to p , with $p_n \neq p$ for all n . If there exist $\lambda > 0$ and $\alpha > 0$ such that

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda,$$

then $\{p_n\}_{n=0}^{\infty}$ is said to converge to p of **order** α , with asymptotic error constant λ .

- An iterative method $p_n = g(p_{n-1})$ is said to be of order α if $\{p_n\}_{n=0}^{\infty}$ converges to p of order α .
- if $\alpha = 1$, we said $\{p_n\}$ is linearly convergent.
- if $1 < \alpha < 2$, we said $\{p_n\}$ is super-linearly convergent.
- if $\alpha = 2$, we said $\{p_n\}$ is quadratically convergent.

Aitken's Δ^2 Method

- Suppose $\{p_n\}_{n=0}^{\infty}$ is linearly convergent, we can accelerate the convergence by construct $\{\tilde{p}_n\}_{n=0}^{\infty}$ by Aitken's Δ^2 Method.
- Since $\{p_n\}_{n=0}^{\infty}$ is linearly convergent, for large n

$$\frac{|p_{n+1} - p|}{|p_n - p|} \approx \lambda \approx \frac{|p_{n+2} - p|}{|p_{n+1} - p|}.$$

- Solving $(p_{n+1} - p)^2 \approx (p_{n+2} - p)(p_n - p)$, we get

$$p \approx \frac{p_n p_{n+2} - p_{n+1}^2}{p_{n+2} - 2p_{n+1} + p_n} = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n} = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}$$

where $\Delta p_n = p_{n+1} - p_n$ and $\Delta^2 p_n = \Delta(\Delta p_n) = p_{n+2} - 2p_{n+1} + p_n$.

- Hence,

$$\tilde{p}_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}$$

are accelerated approximations for p .

Steffensen's Acceleration

- In Aitken's method, a better approximation is produced for every triplets of p_n :
 - from (p_0, p_1, p_2) we get $\tilde{p}_0$,
 - from (p_1, p_2, p_3) we get $\tilde{p}_1$,
 - from (p_2, p_3, p_4) we get $\tilde{p}_2$, and so on
- Steffensen' method make use of the value $\tilde{p}_n$ back into the fixed-point iteration immediately whenever it is available:
 - Given p_0 , use $p_n = g(p_{n-1})$ to find p_1 and p_2
 - from (p_0, p_1, p_2) we get $\tilde{p}_0$,
 - Assume $\tilde{p}_0$ is better approximation then p_2 and calculate $p_3 = g(\tilde{p}_0)$
 - from (p_1, p_2, p_3) we get $\tilde{p}_1$,
 - calculate $p_4 = g(\tilde{p}_1)$
 - from (p_2, p_3, p_4) we get $\tilde{p}_2$, and so on
- See MATLAB codes:
 - `nm02_aitken.m`
 - `nm02_steffesen.m`

Outline

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 Iterative/Open Methods
 - Fixed-point iteration
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

Up to now all our roots are **real**, what if we have complex root? A particular class of root finding problem that of interest is finding the zeros of polynomials.

Theorem (Fundamental Theorem of Algebra)

If $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$ is a polynomial of degree $n \geq 1$, then $P_n(x) = 0$ has at least one (possibly complex) root.

Müller's Method

The idea

- If secant method is using initial points x_0 and x_1 to construct a straight line, and x_2 is the x -intersect, then Müller's method is using a quadratic curve.
- Let $(x_0, f(x_0))$, $(x_1, f(x_1))$ and $(x_2, f(x_2))$ be initial points and let $P(x) = a(x - x_2)^2 + b(x - x_2) + c$ be a parabola passes through these points, then

$$f(x_0) = a(x_0 - x_2)^2 + b(x_0 - x_2) + c,$$

$$f(x_1) = a(x_1 - x_2)^2 + b(x_1 - x_2) + c,$$

$$f(x_2) = a \cdot 0 + b \cdot 0 + c.$$

- solve for a, b and c

$$c = f(x_2)$$

$$b = \frac{(x_0 - x_2)^2[f(x_1) - f(x_2)] - (x_1 - x_2)^2[f(x_0) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}$$

$$a = \frac{(x_1 - x_2)[f(x_0) - f(x_2)] - (x_0 - x_2)[f(x_1) - f(x_2)]}{(x_0 - x_2)(x_1 - x_2)(x_0 - x_1)}$$

Müller's Method (Cont.)

- To find the new approximation x_3 , set $P(x) = 0$, solving the quadratic equation

$$x_3 - x_2 = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}},$$

which gives two possible values for x_3 .

- Taking the largest denominator, we have

$$x_3 = x_2 - \frac{-2c}{b + \operatorname{sgn}(b)\sqrt{b^2 - 4ac}}$$

Example

Find the root of $x^4 - 3x^3 + x^2 + x + 1 = 0$ with Müller's method, given the initial guesses are $x_0 = 0.5$, $x_1 = 0.5$, and $x_2 = 0$.

ANSWER: [MATLAB code: nm02_muller.m]

Outline

- 1 Motivation
- 2 Bracketing Methods
 - Graphing
 - Bisection
 - False-position
- 3 Iterative/Open Methods
 - Fixed-point iteration
 - Newton-Raphson
 - Secant method
- 4 Convergence Acceleration: Aitken's Δ^2 and Steffensen
- 5 Muller's Methods for Polynomials
- 6 System of Nonlinear Equations

System of Nonlinear Equations

The idea

- We can extend Newton-Raphson method to n nonlinear equations.

$$\left. \begin{array}{l} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{array} \right\} \Rightarrow \mathbf{F}(\mathbf{X}) = \mathbf{0}$$

where $\mathbf{F} = [f_1, f_2, \dots, f_n]^T$ and $\mathbf{X} = [x_1, x_2, \dots, x_n]^T$.

- Newton-Raphson method:

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

where $\mathbf{F}'(\mathbf{X}^{(k)})$ is the Jacobian matrix, for example

$$\mathbf{F}'(\mathbf{X}^{(k)}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

System of Nonlinear Equations (Cont.)

- In practice we don't invert $[\mathbf{F}'(\mathbf{X}^{(k)})]$ but use numerical linear algebra methods to evaluate the inverse Jacobian matrix by solving the linear system

$$\mathbf{F}'(\mathbf{X}^{(k)})\mathbf{H}^{(k)} = -\mathbf{F}(\mathbf{X}^{(k)}).$$

Example

Solve the following system of nonlinear equations:

$$\begin{aligned}x + y + z &= 3 \\x^2 + y^2 + z^2 &= 5 \\e^x + xy - xz &= 1\end{aligned}$$

with the initial conditions: $x_1 = 0.1, x_2 = 1.2$ and $x_3 = 2.5$.

ANSWER: [MATLAB code: nm02_system.m]

THE END