Scientific Computing: An Introductory Survey

Chapter 5 – Nonlinear Equations

Prof. Michael T. Heath

Department of Computer Science
University of Illinois at Urbana-Champaign

Selected subset of slides for CS 357 - page numbers will not necessarily be correct

Outline

Nonlinear Equations

Given function f, we seek value x for which

$$f(x) = 0$$

Solution x is *root* of equation, or *zero* of function f

So problem is known as *root finding* or *zero finding*

Nonlinear Equations

Two important cases

Single nonlinear equation in one unknown, where

$$f\colon \mathbb{R} \to \mathbb{R}$$

Solution is scalar x for which $f(x) = 0$

System of n *coupled* nonlinear equations in n unknowns, where

$$\boldsymbol{f}\colon \mathbb{R}^n \to \mathbb{R}^n$$

Solution is vector $\boldsymbol{x}$ for which all components of $\boldsymbol{f}$ are zero *simultaneously*, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$

Examples: Nonlinear Equations

Example of nonlinear equation in one dimension

$$x^2 - 4\sin(x) = 0$$

for which $x = 1.9$ is one approximate solution

Example of system of nonlinear equations in two dimensions

$$\begin{aligned} x_1^2 - x_2 + 0.25 &= 0 \\ -x_1 + x_2^2 + 0.25 &= 0 \end{aligned}$$

for which $\boldsymbol{x} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T$ is solution vector

Existence and Uniqueness

Existence and uniqueness of solutions are more complicated for nonlinear equations than for linear equations

For function $f\colon \mathbb{R} \to \mathbb{R}$, *bracket* is interval $[a, b]$ for which sign of f differs at endpoints

If f is continuous and $\mathrm{sign}(f(a)) \neq \mathrm{sign}(f(b))$, then Intermediate Value Theorem implies there is $x^* \in [a, b]$ such that $f(x^*) = 0$

There is no simple analog for n dimensions

Examples: One Dimension

Nonlinear equations can have any number of solutions

$\exp(x) + 1 = 0$ has no solution

$\exp(-x) - x = 0$ has one solution

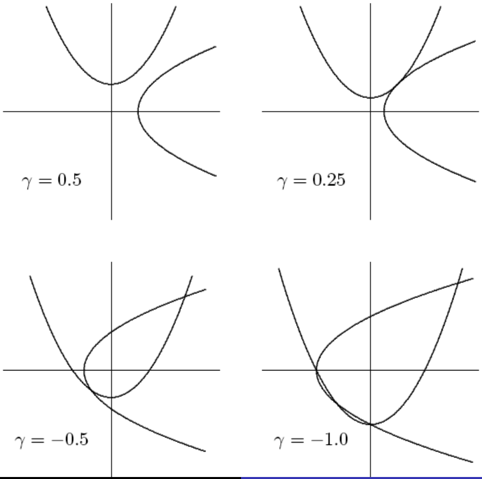
$x^2 - 4\sin(x) = 0$ has two solutions

$x^3 + 6x^2 + 11x - 6 = 0$ has three solutions

$\sin(x) = 0$ has infinitely many solutions

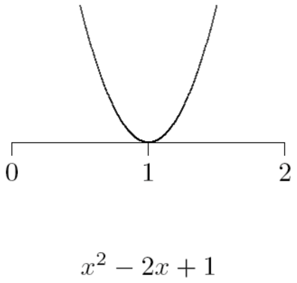
Example: Systems in Two Dimensions

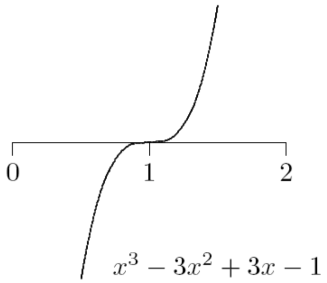
$$\begin{aligned} x_1^2 - x_2 + \gamma &= 0 \\ -x_1 + x_2^2 + \gamma &= 0 \end{aligned}$$



Multiplicity

If $f(x^*) = f'(x^*) = f''(x^*) = \cdots = f^{(m-1)}(x^*) = 0$ but $f^{(m)}(x^*) \neq 0$ (i.e., mth derivative is lowest derivative of f that does not vanish at x^*), then root x^* has *multiplicity* m





If $m = 1$ ($f(x^*) = 0$ and $f'(x^*) \neq 0$), then x^* is *simple* root

Sensitivity and Conditioning

Conditioning of root finding problem is opposite to that for evaluating function

Absolute condition number of root finding problem for root x^* of $f\colon \mathbb{R} \to \mathbb{R}$ is $1/|f'(x^*)|$

Root is ill-conditioned if tangent line is nearly horizontal

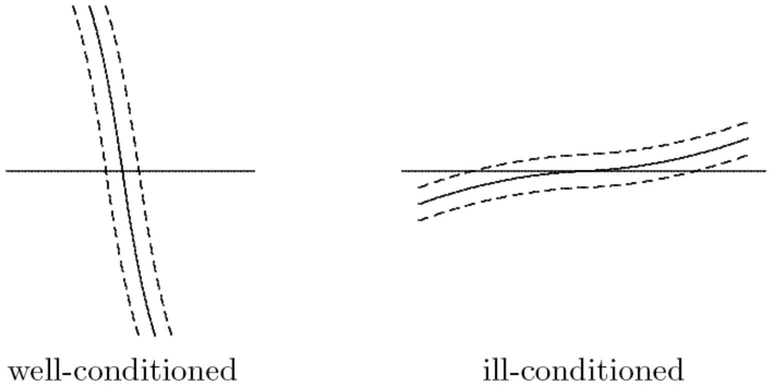
In particular, multiple root ($m > 1$) is ill-conditioned

Absolute condition number of root finding problem for root $\boldsymbol{x}^*$ of $\boldsymbol{f}\colon \mathbb{R}^n \to \mathbb{R}^n$ is $\|\boldsymbol{J}_f^{-1}(\boldsymbol{x}^*)\|$, where $\boldsymbol{J}_f$ is *Jacobian* matrix of $\boldsymbol{f}$,

$$\{\boldsymbol{J}_f(\boldsymbol{x})\}_{ij} = \partial f_i(\boldsymbol{x})/\partial x_j$$

Root is ill-conditioned if Jacobian matrix is nearly singular

Sensitivity and Conditioning



Sensitivity and Conditioning

What do we mean by approximate solution $\hat{\boldsymbol{x}}$ to nonlinear system,

$$\|\boldsymbol{f}(\hat{\boldsymbol{x}})\| \approx 0 \qquad \text{or} \qquad \|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\| \approx 0\ ?$$

First corresponds to "small residual," second measures closeness to (usually unknown) true solution $\boldsymbol{x}^*$

Solution criteria are not necessarily "small" simultaneously

Small residual implies accurate solution only if problem is well-conditioned

Convergence Rate

For general iterative methods, define error at iteration k by

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$$

where $\boldsymbol{x}_k$ is approximate solution and $\boldsymbol{x}^*$ is true solution

For methods that maintain interval known to contain solution, rather than specific approximate value for solution, take error to be length of interval containing solution

Sequence converges with rate r if

$$\lim_{k\to\infty} \frac{\|\boldsymbol{e}_{k+1}\|}{\|\boldsymbol{e}_k\|^r} = C$$

for some finite nonzero constant C

Convergence Rate, continued

Some particular cases of interest

$r = 1$: *linear* $\quad (C < 1)$

$r > 1$: *superlinear*

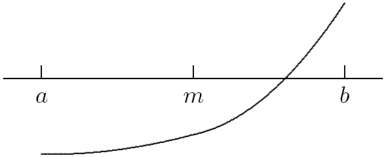
$r = 2$: *quadratic*

Convergence rate	Digits gained per iteration
linear	constant
superlinear	increasing
quadratic	double

Interval Bisection Method

Bisection method begins with initial bracket and repeatedly halves its length until solution has been isolated as accurately as desired

```
while $((b - a) > tol)$ do
    $m = a + (b - a)/2$
    if $\text{sign}(f(a)) = \text{sign}(f(m))$ then
        $a = m$
    else
        $b = m$
    end
end
```



< interactive example >

Example: Bisection Method

$$f(x) = x^2 - 4\sin(x) = 0$$

a	$f(a)$	b	$f(b)$
1.000000	−2.365884	3.000000	8.435520
1.000000	−2.365884	2.000000	0.362810
1.500000	−1.739980	2.000000	0.362810
1.750000	−0.873444	2.000000	0.362810
1.875000	−0.300718	2.000000	0.362810
1.875000	−0.300718	1.937500	0.019849
1.906250	−0.143255	1.937500	0.019849
1.921875	−0.062406	1.937500	0.019849
1.929688	−0.021454	1.937500	0.019849
1.933594	−0.000846	1.937500	0.019849
1.933594	−0.000846	1.935547	0.009491
1.933594	−0.000846	1.934570	0.004320
1.933594	−0.000846	1.934082	0.001736

Bisection Method, continued

Bisection method makes no use of magnitudes of function values, only their signs

Bisection is certain to converge, but does so slowly

At each iteration, length of interval containing solution reduced by half, convergence rate is *linear*, with $r = 1$ and $C = 0.5$

One bit of accuracy is gained in approximate solution for each iteration of bisection

Given starting interval $[a, b]$, length of interval after k iterations is $(b - a)/2^k$, so achieving error tolerance of tol requires

$$\left\lceil \log_2 \left(\frac{b - a}{tol} \right) \right\rceil$$

iterations, regardless of function f involved

Newton's Method

Truncated Taylor series

$$f(x+h) \approx f(x) + f'(x)h$$

is linear function of h approximating f near x

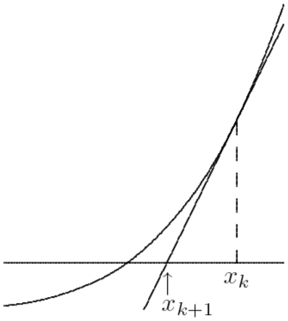
Replace nonlinear function f by this linear function, whose zero is $h = -f(x)/f'(x)$

Zeros of original function and linear approximation are not identical, so repeat process, giving *Newton's method*

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

Newton's Method, continued

Newton's method approximates nonlinear function f near x_k by *tangent line* at $f(x_k)$



Example: Newton's Method

Use Newton's method to find root of

$$f(x) = x^2 - 4\sin(x) = 0$$

Derivative is

$$f'(x) = 2x - 4\cos(x)$$

so iteration scheme is

$$x_{k+1} = x_k - \frac{x_k^2 - 4\sin(x_k)}{2x_k - 4\cos(x_k)}$$

Taking $x_0 = 3$ as starting value, we obtain

x	$f(x)$	$f'(x)$	h
3.000000	8.435520	9.959970	−0.846942
2.153058	1.294772	6.505771	−0.199019
1.954039	0.108438	5.403795	−0.020067
1.933972	0.001152	5.288919	−0.000218
1.933754	0.000000	5.287670	0.000000

Convergence of Newton's Method

Newton's method transforms nonlinear equation $f(x) = 0$ into fixed-point problem $x = g(x)$, where

$$g(x) = x - f(x)/f'(x)$$

and hence

$$g'(x) = f(x)f''(x)/(f'(x))^2$$

If x^* is simple root (i.e., $f(x^*) = 0$ and $f'(x^*) \neq 0$), then $g'(x^*) = 0$

Convergence rate of Newton's method for simple root is therefore *quadratic* ($r = 2$)

But iterations must start close enough to root to converge

< interactive example >

Newton's Method, continued

For multiple root, convergence rate of Newton's method is only linear, with constant $C = 1 - (1/m)$, where m is multiplicity

k	$f(x) = x^2 - 1$	$f(x) = x^2 - 2x + 1$
0	2.0	2.0
1	1.25	1.5
2	1.025	1.25
3	1.0003	1.125
4	1.00000005	1.0625
5	1.0	1.03125

Secant Method

For each iteration, Newton's method requires evaluation of both function and its derivative, which may be inconvenient or expensive

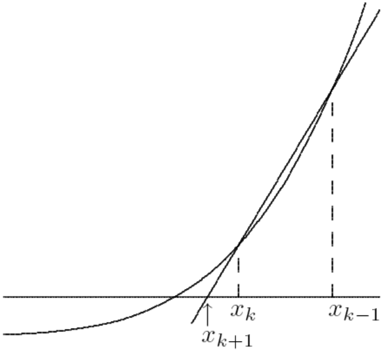
In *secant method*, derivative is approximated by finite difference using two successive iterates, so iteration becomes

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

Convergence rate of secant method is normally *superlinear*, with $r \approx 1.618$

Secant Method, continued

Secant method approximates nonlinear function f by secant line through previous two iterates



< interactive example >

Example: Secant Method

Use secant method to find root of

$$f(x) = x^2 - 4\sin(x) = 0$$

Taking $x_0 = 1$ and $x_1 = 3$ as starting guesses, we obtain

x	$f(x)$	h
1.000000	−2.365884	
3.000000	8.435520	−1.561930
1.438070	−1.896774	0.286735
1.724805	−0.977706	0.305029
2.029833	0.534305	−0.107789
1.922044	−0.061523	0.011130
1.933174	−0.003064	0.000583
1.933757	0.000019	−0.000004
1.933754	0.000000	0.000000

Higher-Degree Interpolation

Secant method uses linear interpolation to approximate function whose zero is sought

Higher convergence rate can be obtained by using higher-degree polynomial interpolation

For example, quadratic interpolation (Muller's method) has superlinear convergence rate with $r \approx 1.839$

Unfortunately, using higher degree polynomial also has disadvantages

- interpolating polynomial may not have real roots
- roots may not be easy to compute
- choice of root to use as next iterate may not be obvious

Safeguarded Methods

Rapidly convergent methods for solving nonlinear equations may not converge unless started close to solution, but safe methods are slow

Hybrid methods combine features of both types of methods to achieve both speed and reliability

Use rapidly convergent method, but maintain bracket around solution

If next approximate solution given by fast method falls outside bracketing interval, perform one iteration of safe method, such as bisection

Safeguarded Methods, continued

Fast method can then be tried again on smaller interval with greater chance of success

Ultimately, convergence rate of fast method should prevail

Hybrid approach seldom does worse than safe method, and usually does much better

Systems of Nonlinear Equations

Solving systems of nonlinear equations is much more difficult than scalar case because

- Wider variety of behavior is possible, so determining existence and number of solutions or good starting guess is much more complex
- There is no simple way, in general, to guarantee convergence to desired solution or to bracket solution to produce absolutely safe method
- Computational overhead increases rapidly with dimension of problem

Newton's Method

In n dimensions, *Newton's method* has form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}(\boldsymbol{x}_k)^{-1}\boldsymbol{f}(\boldsymbol{x}_k)$$

where $\boldsymbol{J}(\boldsymbol{x})$ is Jacobian matrix of $\boldsymbol{f}$,

$$\{\boldsymbol{J}(\boldsymbol{x})\}_{ij} = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j}$$

In practice, we do not explicitly invert $\boldsymbol{J}(\boldsymbol{x}_k)$, but instead solve linear system

$$\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{f}(\boldsymbol{x}_k)$$

for *Newton step* $\boldsymbol{s}_k$, then take as next iterate

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$$

Example: Newton's Method

Use Newton's method to solve nonlinear system

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 - 4 \end{bmatrix} = \boldsymbol{0}$$

Jacobian matrix is $\boldsymbol{J}_f(\boldsymbol{x}) = \begin{bmatrix} 1 & 2 \\ 2x_1 & 8x_2 \end{bmatrix}$

If we take $\boldsymbol{x}_0 = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$, then

$$\boldsymbol{f}(\boldsymbol{x}_0) = \begin{bmatrix} 3 \\ 13 \end{bmatrix}, \quad \boldsymbol{J}_f(\boldsymbol{x}_0) = \begin{bmatrix} 1 & 2 \\ 2 & 16 \end{bmatrix}$$

Solving system $\begin{bmatrix} 1 & 2 \\ 2 & 16 \end{bmatrix} \boldsymbol{s}_0 = \begin{bmatrix} -3 \\ -13 \end{bmatrix}$ gives $\boldsymbol{s}_0 = \begin{bmatrix} -1.83 \\ -0.58 \end{bmatrix}$,

so $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0 = \begin{bmatrix} -0.83 & 1.42 \end{bmatrix}^T$

Example, continued

Evaluating at new point,

$$\boldsymbol{f}(\boldsymbol{x}_1) = \begin{bmatrix} 0 \\ 4.72 \end{bmatrix}, \quad \boldsymbol{J}_f(\boldsymbol{x}_1) = \begin{bmatrix} 1 & 2 \\ -1.67 & 11.3 \end{bmatrix}$$

Solving system $\begin{bmatrix} 1 & 2 \\ -1.67 & 11.3 \end{bmatrix} \boldsymbol{s}_1 = \begin{bmatrix} 0 \\ -4.72 \end{bmatrix}$ gives $\boldsymbol{s}_1 = \begin{bmatrix} 0.64 & -0.32 \end{bmatrix}^T$, so $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{s}_1 = \begin{bmatrix} -0.19 & 1.10 \end{bmatrix}^T$

Evaluating at new point,

$$\boldsymbol{f}(\boldsymbol{x}_2) = \begin{bmatrix} 0 \\ 0.83 \end{bmatrix}, \quad \boldsymbol{J}_f(\boldsymbol{x}_2) = \begin{bmatrix} 1 & 2 \\ -0.38 & 8.76 \end{bmatrix}$$

Iterations eventually convergence to solution $x^* = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$

< interactive example >

Convergence of Newton's Method

Differentiating corresponding fixed-point operator

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{J}(\boldsymbol{x})^{-1}\boldsymbol{f}(\boldsymbol{x})$$

and evaluating at solution $\boldsymbol{x}^*$ gives

$$\boldsymbol{G}(\boldsymbol{x}^*) = \boldsymbol{I} - (\boldsymbol{J}(\boldsymbol{x}^*)^{-1}\boldsymbol{J}(\boldsymbol{x}^*) + \sum_{i=1}^{n} f_i(\boldsymbol{x}^*)\boldsymbol{H}_i(\boldsymbol{x}^*)) = \boldsymbol{O}$$

where $\boldsymbol{H}_i(\boldsymbol{x})$ is component matrix of derivative of $\boldsymbol{J}(\boldsymbol{x})^{-1}$

Convergence rate of Newton's method for nonlinear systems is normally *quadratic*, provided Jacobian matrix $\boldsymbol{J}(\boldsymbol{x}^*)$ is nonsingular

But it must be started close enough to solution to converge

Cost of Newton's Method

Cost per iteration of Newton's method for dense problem in n dimensions is substantial

- Computing Jacobian matrix costs n^2 scalar function evaluations
- Solving linear system costs $\mathcal{O}(n^3)$ operations

Secant Updating Methods

Secant updating methods reduce cost by

- Using function values at successive iterates to build approximate Jacobian and avoiding explicit evaluation of derivatives
- Updating factorization of approximate Jacobian rather than refactoring it each iteration

Most secant updating methods have superlinear but not quadratic convergence rate

Secant updating methods often cost less overall than Newton's method because of lower cost per iteration

Broyden's Method

Broyden's method is typical secant updating method

Beginning with initial guess $\boldsymbol{x}_0$ for solution and initial approximate Jacobian $\boldsymbol{B}_0$, following steps are repeated until convergence

```
$\boldsymbol{x}_0 =$ initial guess
$\boldsymbol{B}_0 =$ initial Jacobian approximation
**for** $k = 0, 1, 2, \ldots$
    Solve $\boldsymbol{B}_k\, \boldsymbol{s}_k = -\boldsymbol{f}(\boldsymbol{x}_k)$ for $\boldsymbol{s}_k$
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$
    $\boldsymbol{y}_k = \boldsymbol{f}(\boldsymbol{x}_{k+1}) - \boldsymbol{f}(\boldsymbol{x}_k)$
    $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + ((\boldsymbol{y}_k - \boldsymbol{B}_k \boldsymbol{s}_k)\boldsymbol{s}_k^T)/(\boldsymbol{s}_k^T \boldsymbol{s}_k)$
**end**
```

Broyden's Method, continued

Motivation for formula for $\boldsymbol{B}_{k+1}$ is to make least change to $\boldsymbol{B}_k$ subject to satisfying *secant equation*

$$\boldsymbol{B}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = \boldsymbol{f}(\boldsymbol{x}_{k+1}) - \boldsymbol{f}(\boldsymbol{x}_k)$$

In practice, factorization of $\boldsymbol{B}_k$ is updated instead of updating $\boldsymbol{B}_k$ directly, so total cost per iteration is only $\mathcal{O}(n^2)$

Example: Broyden's Method

Use Broyden's method to solve nonlinear system

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 - 4 \end{bmatrix} = \boldsymbol{0}$$

If $\boldsymbol{x}_0 = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$, then $\boldsymbol{f}(\boldsymbol{x}_0) = \begin{bmatrix} 3 & 13 \end{bmatrix}^T$, and we choose

$$\boldsymbol{B}_0 = \boldsymbol{J}_f(\boldsymbol{x}_0) = \begin{bmatrix} 1 & 2 \\ 2 & 16 \end{bmatrix}$$

Solving system

$$\begin{bmatrix} 1 & 2 \\ 2 & 16 \end{bmatrix} \boldsymbol{s}_0 = \begin{bmatrix} -3 \\ -13 \end{bmatrix}$$

gives $\boldsymbol{s}_0 = \begin{bmatrix} -1.83 \\ -0.58 \end{bmatrix}$, so $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0 = \begin{bmatrix} -0.83 \\ 1.42 \end{bmatrix}$

Example, continued

Evaluating at new point $\boldsymbol{x}_1$ gives $\boldsymbol{f}(\boldsymbol{x}_1) = \begin{bmatrix} 0 \\ 4.72 \end{bmatrix}$, so

$$\boldsymbol{y}_0 = \boldsymbol{f}(\boldsymbol{x}_1) - \boldsymbol{f}(\boldsymbol{x}_0) = \begin{bmatrix} -3 \\ -8.28 \end{bmatrix}$$

From updating formula, we obtain

$$\boldsymbol{B}_1 = \begin{bmatrix} 1 & 2 \\ 2 & 16 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -2.34 & -0.74 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ -0.34 & 15.3 \end{bmatrix}$$

Solving system

$$\begin{bmatrix} 1 & 2 \\ -0.34 & 15.3 \end{bmatrix} \boldsymbol{s}_1 = \begin{bmatrix} 0 \\ -4.72 \end{bmatrix}$$

gives $\boldsymbol{s}_1 = \begin{bmatrix} 0.59 \\ -0.30 \end{bmatrix}$, so $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{s}_1 = \begin{bmatrix} -0.24 \\ 1.120 \end{bmatrix}$

Example, continued

Evaluating at new point $\boldsymbol{x}_2$ gives $\boldsymbol{f}(\boldsymbol{x}_2) = \begin{bmatrix} 0 \\ 1.08 \end{bmatrix}$, so

$$\boldsymbol{y}_1 = \boldsymbol{f}(\boldsymbol{x}_2) - \boldsymbol{f}(\boldsymbol{x}_1) = \begin{bmatrix} 0 \\ -3.64 \end{bmatrix}$$

From updating formula, we obtain

$$\boldsymbol{B}_2 = \begin{bmatrix} 1 & 2 \\ -0.34 & 15.3 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1.46 & -0.73 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1.12 & 14.5 \end{bmatrix}$$

Iterations continue until convergence to solution $\boldsymbol{x}^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

< interactive example >

Robust Newton-Like Methods

Newton's method and its variants may fail to converge when started far from solution

Safeguards can enlarge region of convergence of Newton-like methods

Simplest precaution is *damped Newton method*, in which new iterate is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k$$

where $\boldsymbol{s}_k$ is Newton (or Newton-like) step and α_k is scalar parameter chosen to ensure progress toward solution

Parameter α_k reduces Newton step when it is too large, but $\alpha_k = 1$ suffices near solution and still yields fast asymptotic convergence rate

Trust-Region Methods

Another approach is to maintain estimate of *trust region* where Taylor series approximation, upon which Newton's method is based, is sufficiently accurate for resulting computed step to be reliable

Adjusting size of trust region to constrain step size when necessary usually enables progress toward solution even starting far away, yet still permits rapid converge once near solution

Unlike damped Newton method, trust region method may modify direction as well as length of Newton step