12. Nonlinear equations and optimization

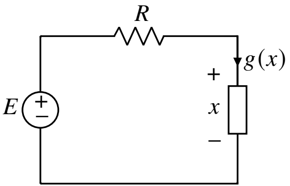
- nonlinear equation in one variable
- Newton method for nonlinear equations
- unconstrained optimization
- gradient and Newton methods for optimization

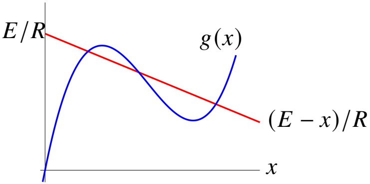
Nonlinear equation in one variable

$$f(x) = 0$$

- the *root* or *zero* is any solution of the above equation
- we assume f is a continuous function

Example: nonlinear resistive circuit

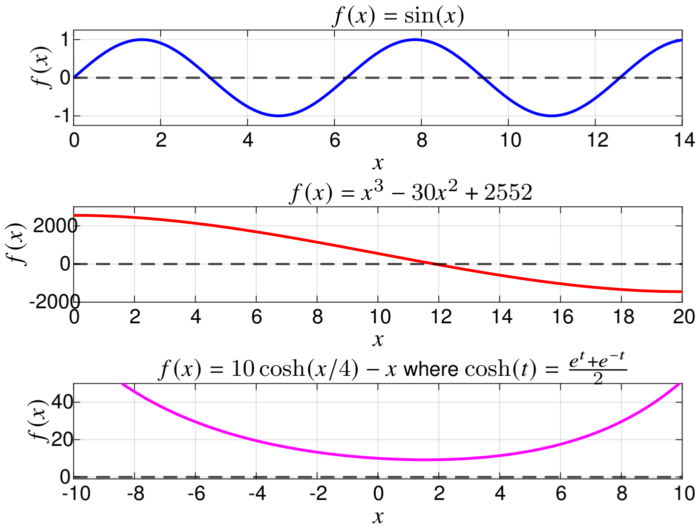




$$g(x) - \frac{E-x}{R} = 0$$

a nonlinear equation in the variable x, with three solutions

Examples


$f(x) = \sin(x)$
$f(x)$
1
0
-1
0
2
4
6
8
10
12
14
x
$f(x) = x^3 - 30x^2 + 2552$
$f(x)$
2000
0
-2000
0
2
4
6
8
10
12
14
16
18
20
x
$f(x) = 10\cosh(x/4) - x$ where $\cosh(t) = \frac{e^t+e^{-t}}{2}$
$f(x)$
40
20
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
x

Iterative methods

- nonlinear equations are much difficult to solve compared to linear equations
- obtaining a solution by finite-step algorithm is not feasible
- iterative algorithms start with *initial* or *starting point*, $x^{(1)}$ and compute estimates

$$x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots$$

- moving from $x^{(k)}$ to $x^{(k+1)}$ is called an *iteration* of the algorithm
- ideally converge to a root of the target function

$$x^{(k)} \to x^\star \quad \text{as } k \to \infty$$

  where $f(x^\star) = 0$

Convergence rates

assume the sequence $x^{(k)}$ converges to a limit $x^\star$

Linear convergence: if there exists a constant $c \in (0,1)$ such that

$$|x^{(k)} - x^\star| \leq c|x^{(k-1)} - x^\star| \quad \text{for sufficiently large } k$$

example: $x^{(k)} = 1 + (1/2)^k$ linearly converges to $x^\star = 1$,

$$|x^{(k+1)} - x^\star| = (1/2)^{k+1} = \frac{1}{2}|x^{(k)} - x^\star|$$

satisfies the definition with $c = 1/2$

R-linear convergence: if a positive constant M and a value $c \in (0,1)$ exist such that

$$|x^{(k)} - x^\star| \leq Mc^k \quad \text{for sufficiently large } k$$

linear convergence implies R-linear convergence (reverse is not necessarily true)

Superlinear convergence: if a sequence $c_k > 0$ with $c_k \to 0$ exists such that

$$|x^{(k)} - x^\star| \leq c_k |x^{(k-1)} - x^\star| \quad \text{for large } k$$

example: $x^{(k)} = 1 + (1/(k+1))^k$ has superlinear convergence to $x^\star = 1$, as

$$|x^{(k)} - x^\star| = \frac{1}{(k+1)^k} = \frac{k^{k-1}}{(k+1)^k} \frac{1}{k^{k-1}} = \frac{k^{k-1}}{(k+1)^k} |x^{(k-1)} - x^\star|$$

satisfies the definition with $c_k = k^{k-1}/(k+1)^k$, which approaches zero

Quadratic convergence: if a constant $c > 0$ exists such that

$$|x^{(k)} - x^\star| \leq c |x^{(k-1)} - x^\star|^2 \quad \text{for large } k$$

example: $x^{(k)} = 1 + (1/2)^{2^k}$ has quadratic convergence to $x^\star = 1$, as

$$|x^{(k+1)} - x^\star| = (1/2)^{2^{k+1}} = \left((1/2)^{2^k}\right)^2 = |x^{(k)} - x^\star|^2$$

satisfies the definition with $c = 1$

The bisection method

---

given: a, b with $a < b$, $f(a)f(b) < 0$, and tolerance ϵ

repeat

1. $x = (a + b)/2$
2. compute $f(x)$; **if** $f(x) = 0$, **return** x
3. **if** $f(x)f(a) < 0$, $b = x$, **else**, $a = x$
4. **stop** if $b - a \leq \epsilon$

- condition $f(a)f(b) < 0$ ensures a root exists between a, b
- a, b can be chosen from graphing the function

Convergence

let $[a^{(k)}, b^{(k)}]$ be the interval after iteration k, then

$$b^{(k)} - a^{(k)} = (b - a)/2^k$$

- after k iterations, the midpoint $x^{(k)} = (b^{(k)} + a^{(k)})/2$ satisfies

$$|x^{(k)} - x^\star| \leq b^{(k)} - a^{(k)} \leq (1/2)^k (b - a)$$

  thus, it is R-linearly convergent with $c = 1/2$ and $M = b - a$

- the exit condition $b^{(k)} - a^{(k)} \leq \epsilon$ will be satisfied if

$$\log_2\left(\frac{b-a}{2^k}\right) = \log_2(b - a) - k \leq \log_2 \epsilon$$

  the algorithm therefore terminates after

$$k \approx \left\lceil \log_2\left(\frac{b-a}{\epsilon}\right) \right\rceil$$

  iterations where $\lceil \alpha \rceil$ is the smallest integer greater than or equal to α

MATLAB implementation

```
function [x,k] = bisect(f,a,b,tol)
% assuming func is a defined input  function
% this function returns in p a value such that | x - root | < atol
% and in k the number of iterations required.
fa=f(a);fb=f(b);
if (a >= b) | (fa*fb >= 0) | (tol <= 0)
disp('something wrong with the input: quitting');
x = NaN; k=NaN;
return
end
k = ceil(log2(b-a) - log2(tol));
for i=1:k
x = (a+b)/2;
fx = f(x);
if abs(fx) < eps, k = i; return, end
if fa * fx < 0
b = x; fb = fx;
else
a = x; end
end
end
```

Examples

- for $f(x) = x^3 - 30x^2 + 2552$, starting from interval $[0, 20]$ with a tolerance of 1×10^{-8}, the method converges to $x^\star \approx 11.86150151$ after 31 iterations

  the associated MATLAB script is:

```
f = @(x) x^3 - 30*x^2 + 2552;
[x,k] = bisect(f,0,20,1.e-8)
```

- for $f(x) = 2.5 \sinh(x/4) - 1$, beginning with interval $[-10, 10]$ and using a tolerance of 1×10^{-10}, the method converges to $x^\star \approx 1.5601412791$ after 38 iterations

  the associated MATLAB script is:

```
f = @(x) 2.5 * sinh (x/4) - 1;
[x,k] = bisect(f,-10,10,1.e-10)
```

Outline

- nonlinear equation in one variable
- **Newton method for nonlinear equations**
- unconstrained optimization
- gradient and Newton methods for optimization

Set of nonlinear equations

consider n nonlinear equations in n variables $x_1, x_2, \ldots, x_n$:

$$f_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, n$$

in vector notation: $f(x) = 0$ with

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad f(x) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{bmatrix}$$

- $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$
- $f_i(x)$ is ith *residual*
- $f(x)$ is residual vector

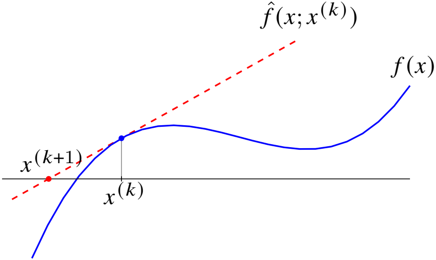
Deriving Newton method

- linearize f (*i.e.*, make affine approximation) around current iterate $x^{(k)}$

$$\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})$$

- take solution x of linearized equation $\hat{f}(x; x^{(k)}) = 0$ as the next iterate $x^{(k+1)}$:

$$x = x^{(k)} - Df(x^{(k)})^{-1} f(x^{(k)})$$



Newton method for nonlinear equations

assume $f : \mathbb{R}^n \to \mathbb{R}^n$ is differentiable

---

given a starting point $x^{(1)}$ and solution tolerance ϵ

repeat for $k \geq 0$

1. evaluate $Df(x^{(k)})$
2. set
$$x^{(k+1)} = x^{(k)} - Df(x^{(k)})^{-1} f(x^{(k)})$$

   if $\|f(x^{(k+1)})\| < \epsilon$ (or $\|x^{(k+1)} - x^{(k)}\| < \epsilon$), stop and output $x^{(k+1)}$

- $Df(x^{(k)})$ is derivative (Jacobian) matrix of f at $x^{(k)}$ assumed to be nonsingular
- each iteration requires one evaluation of $f(x)$ and $Df(x)$
- each iteration requires factorization of the $n \times n$ matrix $Df(x)$
- also called Newton-Raphson algorithm

Example

applying Newton's method on $f(x) = 2\cosh(\frac{x}{4}) - x$ gives

$$x^{(k+1)} = x^{(k)} - \frac{2\cosh(x^{(k)}/4) - x^{(k)}}{0.5\sinh(x^{(k)}/4) - 1}$$

where $\cosh(u) = (e^u + e^{-u})/2$ and $\sinh(u) = (e^u - e^{-u})/2$

with tolerance of 1×10^{-8}, we have

- starting from $x_0 = 2$, 4 iterations are needed to get $x_1^\star = 2.35755106$
- from $x_0 = 8$, 5 iterations are enough to reach $x_2^\star = 8.50719958$

for $x_0 = 8$, the values of $f(x^{(k)})$ evolve as:

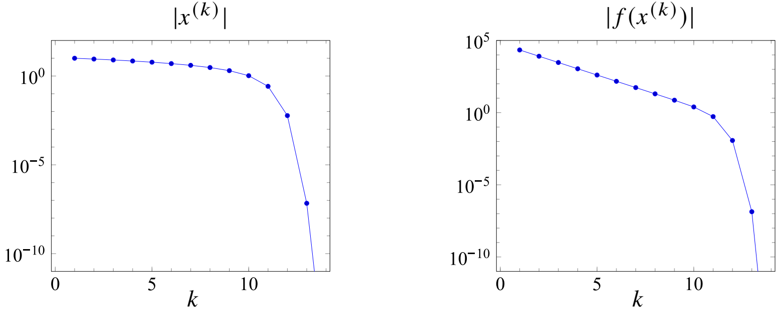
k	0	1	2	3	4	5
$f(x^{(k)})$	-4.76e-1	8.43e-2	1.56e-3	5.65e-7	7.28e-14	1.78e-15

Example

Newton method applied to $f(x) = e^x - e^{-x}$

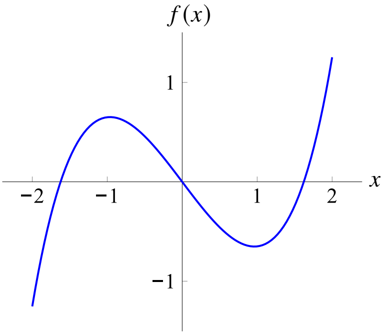
$$x^{(k+1)} = x^{(k)} - \frac{e^{x^{(k)}} - e^{-x^{(k)}}}{e^{x^{(k)}} + e^{-x^{(k)}}}$$

results with $x^{(1)} = 10$



Example

$$f(x) = e^{x} - e^{-x} - 3x$$

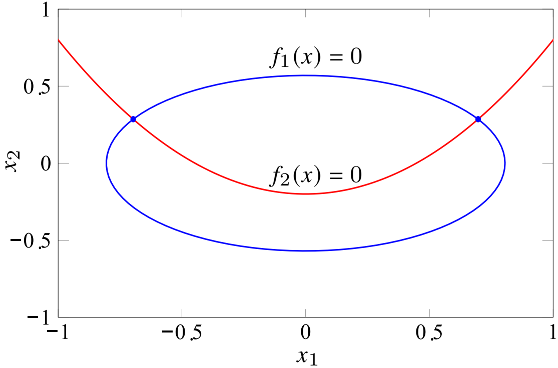


- starting point $x^{(1)} = -1$: converges to $x^{\star} = -1.62$
- starting point $x^{(1)} = -0.8$: converges to $x^{\star} = 1.62$
- starting point $x^{(1)} = -0.7$: converges to $x^{\star} = 0$

Example

$$f_1(x_1, x_2) = \log\left(x_1^2 + 2x_2^2 + 1\right) - 0.5 = 0$$

$$f_2(x_1, x_2) = x_2 - x_1^2 + 0.2 = 0$$



two equations in two variables; two solutions $(0.70, 0.29)$, $(-0.70, 0.29)$

Newton iteration

- evaluate $g = f(x)$ and

$$H = Df(x) = \left[\begin{array}{cc} 2x_1/\left(x_1^2 + 2x_2^2 + 1\right) & 4x_2/\left(x_1^2 + 2x_2^2 + 1\right) \\ -2x_1 & 1 \end{array}\right]$$

- solve $Hv = -g$ (two linear equations in two variables)
- update $x := x + v$

Results

- $x^{(1)} = (1, 1)$: converges to $x^\star = (0.70, 0.29)$ in about 4 iterations
- $x^{(1)} = (-1, 1)$: converges to $x^\star = (-0.70, 0.29)$ in about 4 iterations
- $x^{(1)} = (1, -1)$ or $x^{(1)} = (-1, -1)$: does not converge

Observations

- Newton's method works well if started near a solution; may not work otherwise
- can converge to different solutions depending on the starting point
- does not necessarily find the solution closest to the starting point

Convergence of Newton's method

if $f(x^\star) = 0$ and $Df(x^\star)$ is nonsingular, and $x^{(1)}$ is sufficiently close to $x^\star$, then

$$x^{(k)} \to x^\star, \quad \|x^{(k+1)} - x^\star\| \leq c\|x^{(k)} - x^\star\|^2$$

for some $c > 0$

- has quadratic convergence when started near a solution
- explains fast convergence when started near solution

Outline

- nonlinear equation in one variable
- Newton method for nonlinear equations
- **unconstrained optimization**
- gradient and Newton methods for optimization

Hessian

Hessian of g at z: a symmetric $n \times n$ matrix

$$\nabla^2 g(x) = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_n} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} & \cdots & \frac{\partial^2 g}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 g}{\partial x_n \partial x_1} & \frac{\partial^2 g}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_n^2} \end{bmatrix}$$

- $\nabla^2 g(z)_{ij} = \frac{\partial^2 g}{\partial x_i \partial x_j}(z)$
- this is also the derivative matrix of gradient $\nabla g(x)$ at z

Quadratic (second order) approximation of g around z

$$g_{\mathrm{q}}(x) = g(z) + \nabla g(z)^T (x - z) + \frac{1}{2}(x - z)^T \nabla^2 g(z)(x - z)$$

when $n = 1$ this reduces to

$$g_{\mathrm{q}}(x) = g(z) + g'(z)(x - z) + \frac{1}{2} g''(z)(x - z)^2$$

Examples

Affine function: $g(x) = a^T x + b$

$$\nabla g(x) = a, \quad \nabla^2 g(x) = 0$$

Quadratic function: $g(x) = x^T P x + q^T x + r$ with P symmetric

$$\nabla g(x) = 2Px + q, \quad \nabla^2 g(x) = 2P$$

Least squares cost: $g(x) = \|Ax - b\|^2 = x^T A^T A x - 2b^T A x + b^T b$

$$\nabla g(x) = 2A^T A x - 2A^T b, \quad \nabla^2 g(x) = 2A^T A$$

Composition with affine mapping: if $g(x) = h(Cx + d)$, then

$$\begin{aligned} \nabla g(x) &= C^T \nabla h(Cx + d) \\ \nabla^2 g(x) &= C^T \nabla^2 h(Cx + d) C \end{aligned}$$

Example

$$g(x_1, x_2) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

gradient is

$$\nabla g(x) = \left[\begin{array}{c} e^{x_1+x_2-1} + e^{x_1-x_2-1} - e^{-x_1-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} \end{array} \right]$$

Hessian is

$$\nabla^2 g(x) = \left[\begin{array}{cc} e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1} & e^{x_1+x_2-1} - e^{x_1-x_2-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} & e^{x_1+x_2-1} + e^{x_1-x_2-1} \end{array} \right]$$

using composition property we can express g as $g(x) = h(Cx + d)$ with

$$h(y_1, y_2, y_3) = e^{y_1} + e^{y_2} + e^{y_3}, \quad C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}, \quad d = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Gradient: $\nabla g(x) = C^T \nabla h(Cx + d)$

$$\nabla g(x) = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} \\ e^{x_1-x_2-1} \\ e^{-x_1-1} \end{bmatrix}$$

Hessian: $\nabla^2 g(x) = C^T \nabla h^2(Cx + d) C$

$$\nabla^2 g(x) = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} & 0 & 0 \\ 0 & e^{x_1-x_2-1} & 0 \\ 0 & 0 & e^{-x_1-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}$$

Unconstrained minimization problem

$$\text{minimize} \quad g(x_1, x_2, \ldots, x_n)$$

- $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the *cost* or *objective* function
- $x = (x_1, x_2, \ldots, x_n)$ is n-vector of optimization *variables*
- to solve a maximization problem (*i.e.*, maximize $g(x)$), we can minimize $-g(x)$
- we will assume that g is twice differentiable

Local and global optimum

- $x^\star$ is an **optimal point** (or a **minimum**) if

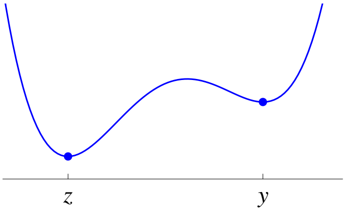
$$g(x^\star) \leq g(x) \text{ for all } x$$

  also called globally optimal

- $x^\star$ is a **locally optimal point** (**local minimum**) if for some $R > 0$

$$g(x^\star) \leq g(x) \quad \text{for all } x \text{ with } \|x - x^\star\| \leq R$$

Example



z is (globally) optimal $\qquad$ y is locally optimal

Conditions for a minimum

a necessary condition for a minimum is

$$\nabla g(x^\star) = 0$$

a point where the gradient vanishes $\nabla g(x) = 0$ is called a *critical* or *stationary* point

- if $x^\star$ is a local minimum, then for any direction v we have

$$g(x^\star + v) = g(x^\star) + \nabla g(x^\star)^T v + (1/2) v^T \nabla^2 g(x^\star) v \geq g(x^\star)$$

- for a very small $\|v\|$, if $\nabla g\,(x^\star) \neq 0$, then we can find v such that $\nabla g(x^\star)^T v < 0$
- so we must have $\nabla g(x^\star) = 0$ at a minimum (or maximum)
- gradient also vanish at saddle points, which is neither a minimum or maximum
- at a strict minimum we must also have for all v satisfying $0 < \|v\| \ll 1$

$$g(x^\star + v) = g(x^\star) + (1/2) v^T \nabla^2 g(x^\star) v > g(x^\star)$$

this will happen if the Hessian matrix $\nabla^2 g(x^\star)$ is positive definite

Optimality conditions

Necessary condition: if $x^\star$ is locally optimal, then

$$\nabla g(x^\star) = 0 \quad \text{and} \quad \nabla^2 g(x^\star) \text{ is positive semidefinite}$$

Sufficient condition: if $x^\star$ satisfies

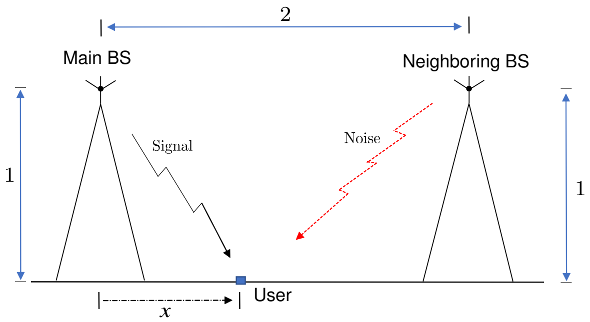
$$\nabla g(x^\star) = 0 \quad \text{and} \quad \nabla^2 g(x^\star) \text{ is positive definite}$$

then $x^\star$ is locally optimal

Necessary and sufficient condition for convex functions

- g is called *convex* if $\nabla^2 g(x)$ is positive semidefinite everywhere
- if g is convex then $x^\star$ is optimal if and only if $\nabla g(x^\star) = 0$

Example



- power of the received signal measured by the user from each antenna is the reciprocal of the squared distance from the corresponding antenna
- find position x of user (relative to main station) that maximizes signal-to-noise ratio

to solve this problem, we need to maximize the signal-to-noise ratio:

$$g(x) = \frac{1 + (2 - x)^2}{1 + x^2}$$

setting the derivative to zero:

$$g'(x) = \frac{-2(2 - x)(1 + x^2) - 2x(1 + (2 - x)^2)}{(1 + x^2)^2} = \frac{4(x^2 - 2x - 1)}{(1 + x^2)^2} = 0$$

- $g'(x) = 0$ at $x = 1 \pm \sqrt{2}$
- checking the objective values, we see that $x = 1 - \sqrt{2}$ gives larger objective
- derivative changes sign + to $-$ when passing through $x = 1 - \sqrt{2}$, so $f''(x) < 0$
- hence, $x^\circ = 1 - \sqrt{2}$ is a local maximizer
- it is a global maximizer since $g(x) \to 1 < g(x^\circ)$ as $|x| \to \infty$

Examples $(n = 1)$

- $g(x) = \log(e^x + e^{-x})$

$$g'(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad g''(x) = \frac{4}{(e^x + e^{-x})^2}$$

  $g''(x) \geq 0$ everywhere; $x^\star = 0$ is the unique optimal point

- $g(x) = x^4$

$$g'(x) = 4x^3, \quad g''(x) = 12x^2$$

  $g''(x) \geq 0$ everywhere; $x^\star = 0$ is the unique optimal point

- $g(x) = x^3$

$$g'(x) = 3x^2, \quad g''(x) = 6x$$

  $g'(0) = 0$, $g''(0) = 0$ but $x = 0$ is not locally optimal

Examples

- $g(x) = x^T P x + q^T x + r$ (P is symmetric positive definite)

$$\nabla g(x) = 2Px + q, \quad \nabla^2 g(x) = 2P$$

$\nabla^2 g(x)$ is positive definite everywhere, hence the unique optimal point is

$$x^\star = -(1/2)P^{-1}q$$

- $g(x) = \|Ax - b\|^2$ (A is a matrix with linearly independent columns)

$$\nabla g(x) = 2A^T Ax - 2A^T b, \quad \nabla^2 g(x) = 2A^T A$$

$\nabla^2 g(x)$ is positive definite everywhere, hence the unique optimal point is

$$x^\star = (A^T A)^{-1} A^T b$$

Examples

$$g\left(x_1, x_2\right) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

- we can express $\nabla^2 g(x)$ as

$$\nabla^2 g(x) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} & 0 & 0 \\ 0 & e^{x_1-x_2-1} & 0 \\ 0 & 0 & e^{-x_1-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix}$$

  this shows that $\nabla^2 g(x)$ is positive definite for all x

- therefore $x^\star$ is optimal if and only if

$$\nabla g\left(x^\star\right) = \begin{bmatrix} e^{x_1^\star+x_2^\star-1} + e^{x_1^\star-x_2^\star-1} - e^{-x_1^\star-1} \\ e^{x_1^\star+x_2^\star-1} - e^{x_1^\star-x_2^\star-1} \end{bmatrix} = 0$$

  two nonlinear equations in two variables

Outline

- nonlinear equation in one variable
- Newton method for nonlinear equations
- unconstrained optimization
- **gradient and Newton methods for optimization**

Descent direction

Descent direction: a vector $v \in \mathbb{R}^n$ is called a *descent direction* for g if

$$g(x + tv) < g(x)$$

for sufficiently small $t > 0$

Directional derivative

- for given z and nonzero v, define $h(t) = g(z + tv)$
- derivative of h at $t = 0$

$$\begin{aligned} h'(0) &= \frac{\partial g}{\partial x_1}(z)v_1 + \frac{\partial g}{\partial x_2}(z)v_2 + \cdots + \frac{\partial g}{\partial x_n}(z)v_n \\ &= \nabla g(z)^T v \end{aligned}$$

  is called the *directional derivative* of g (at z, in the direction v)

- $\nabla g(x)^T v$ gives an approximate rate of increase of f in the direction v at x
- a vector $v \in \mathbb{R}^n$ is a descent direction if $\nabla g(x)^T v < 0$

Gradient descent method

the directional derivative of f at x in the direction $v = -\nabla g(x)$ is

$$v^T \nabla g(x) = -\|\nabla g(x)\|^2 < 0$$

for any x with $\nabla g(x) \neq 0$; thus, $-\nabla g(x)$ is a descent direction

Gradient descent algorithm

---

given a starting point $x^{(0)}$ and a solution tolerance $\epsilon > 0$

repeat for $k \geq 1$

1. choose a stepsize t_k
2. update

$$x^{(k+1)} = x^{(k)} - t_k \nabla g(x^{(k)})$$

if $\|\nabla f(x^{(k+1)})\| \leq \epsilon$ stop and output $x^{(k+1)}$

- t_k is called the *stepsize* or *learning rate*
- for t_k small enough, the algorithm is a descent method
- for large t_k is large enough, algorithm may not be a descent method and may fail

Determining the stepsize

suppose $v^{(k)}$ is any descent direction

Constant stepsize: set $t_k = t$ for all k

Backtracking line search

- choose $\beta \in (0, 1)$, and $\gamma \in (0, 1)$ and start with an initial guess t_1 (*e.g.*, $t_1 = 1$)
- set $t_k := \beta t_k$ until

$$g(x^{(k)} + t_k v^{(k)}) - g(x^{(k)}) < \gamma t_k \nabla g(x^{(k)})^T v^{(k)}$$

- simple backtracking algorithm is to set

$$t = 1, 0.5, 0.5^2, 0.5^3, \ldots$$

  until the above is satisfied or until $g(x^{(k)} + t_k v^{(k)}) < g(x^{(k)})$

Example

$$g(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

the gradient of this function is

$$\nabla g(x) = \begin{bmatrix} 4(x_1 - 4)^3 \\ 2(x_2 - 3) \\ 16(x_3 + 5)^3 \end{bmatrix}$$

applying one iteration of gradient descent with $x^{(1)} = (4, 2, -1)$ and $t = 0.002$ gives

$$x^{(2)} = \begin{bmatrix} 4 \\ 2 \\ -1 \end{bmatrix} - 0.002 \begin{bmatrix} 4(4-4)^3 \\ 2(2-3) \\ 16(-1+5)^3 \end{bmatrix} = \begin{bmatrix} 4.000 \\ 2.004 \\ -3.048 \end{bmatrix}$$

notice that

$$59.06 = g(4, 2.004, -3.048) < g(4, 2, -1) = 1025$$

this shows that $t = 0.002$ is a good choice

Newton's method for minimizing a convex function

if $\nabla^2 g(x)$ is positive definite everywhere, we can minimize $g(x)$ by solving

$$\nabla g(x) = 0$$

Algorithm: choose $x^{(1)}$ and repeat for $k = 1, 2, \ldots$

$$x^{(k+1)} = x^{(k)} - \nabla^2 g(x^{(k)})^{-1} \nabla g(x^{(k)})$$

- $v = -\nabla^2 g(x)^{-1} \nabla g(x)$ is called the *Newton step* at x, which is a descent direction
- converges if started sufficiently close to the solution
- Newton step is computed by a Cholesky factorization of the Hessian
- for $n = 1$, the iteration can be written as

$$x^{(k+1)} = x^{(k)} - \frac{g'(x^{(k)})}{g''(x^{(k)})}$$

Interpretations of Newton step

Affine approximation of gradient

- affine approximation of $f(x) = \nabla g(x)$ around $x^{(k)}$ is

$$\hat{f}(x; x^{(k)}) = \nabla g(x^{(k)}) + \nabla^2 g(x^{(k)})(x - x^{(k)})$$

- Newton update $x^{(k+1)}$ is solution of linear equation $\hat{f}(x; x^{(k)}) = 0$

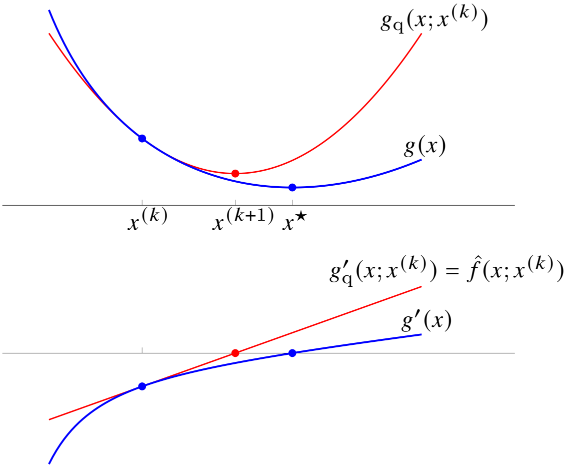
Quadratic approximation of function

- quadratic approximation of $g(x)$ around $x^{(k)}$ is

$$g_{\mathrm{q}}(x; x^{(k)}) = g(x^{(k)}) + \nabla g(x^{(k)})^T(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T \nabla^2 g(x^{(k)})(x - x^{(k)})$$

- Newton update $x^{(k+1)}$ satisfies $\nabla g_{\mathrm{q}}(x; x^{(k)}) = 0$

Example $(n = 1)$



$$g_q(x; x^{(k)}) = g(x^{(k)}) + g'(x^{(k)})(x - x^{(k)}) + \frac{g''(x^{(k)})}{2}(x - x^{(k)})^T(x - x^{(k)})$$

Example

$$\text{minimize} \quad g(x) = \frac{1}{2}x^2 - \sin x$$

- applying Newton's method with $x^{(1)} = 0.5$, we have

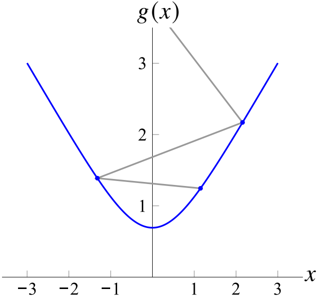
$$\begin{aligned} x^{(2)} = x^{(1)} - \frac{g'(x^{(1)})}{g''(x^{(1)})} &= 0.5 - \frac{0.5 - \cos(0.5)}{1 + \sin(0.5)} \\ &= 0.5 - \frac{-0.3775}{1.479} = 0.7552 \end{aligned}$$

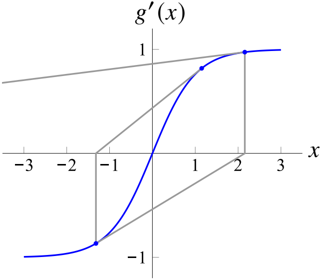
  repeating, we get $x^{(3)} = 0.7391$, $x^{(4)} = 0.7390$, and $x^{(5)} \approx 0.7390$

- note that $g'(x^{(5)}) \approx 0$, and $g''(x^{(5)}) = 1.672 > 0$
- so, $x^{(5)}$ is an approximate local minimizer (it is an approximate global minimizer)

Example

$$g(x) = \log\left(e^x + e^{-x}\right), \quad g'(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad g''(x) = \frac{4}{(e^x + e^{-x})^2}$$





does not converge when started at $x^{(1)} = 1.15$

Damped Newton method

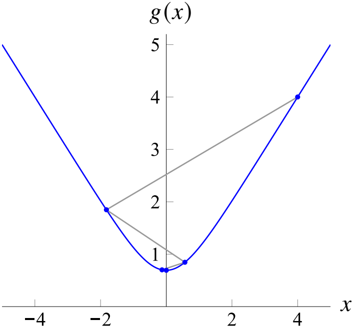
given a starting point $x^{(1)}$

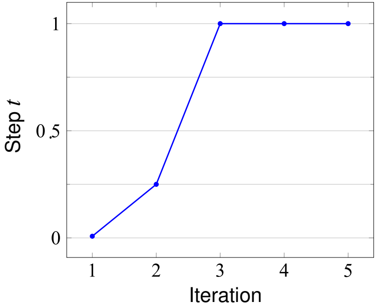
repeat for $k = 1, 2, \ldots$

1. compute Newton step $v = -\nabla^2 g(x^{(k)})^{-1} \nabla g(x^{(k)})$
2. select a stepsize t (*e.g.*, using backtracking line search)
3. update $x^{(k+1)} = x^{(k)} + tv$

Example

$$g(x) = \log\left(e^{x} + e^{-x}\right), \quad x^{(1)} = 4$$



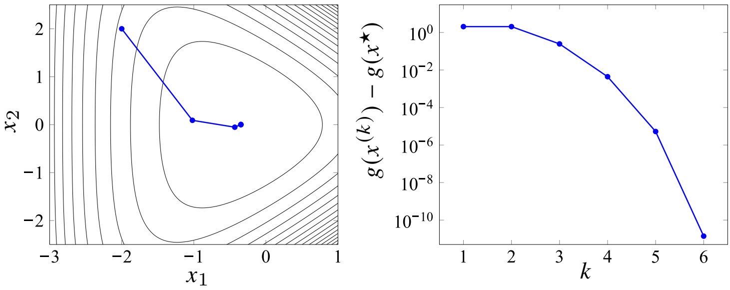


close to the solution: very fast convergence, no backtracking steps

Example

$$g(x_1, x_2) = e^{x_1 + x_2 - 1} + e^{x_1 - x_2 - 1} + e^{-x_1 - 1}$$

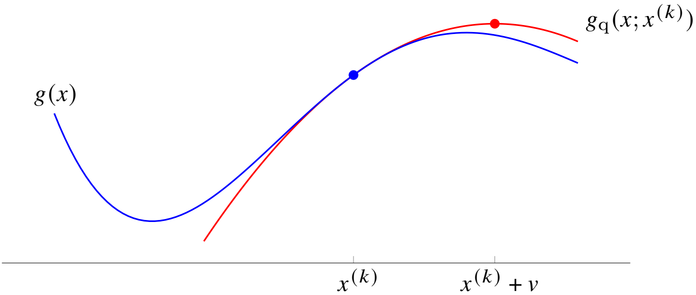
damped Newton method started at $x = (-2, 2)$



Newton method for nonconvex functions

if $\nabla^2 g(x^{(k)})$ is not positive definite, it is possible that Newton step v satisfies

$$\nabla g(x^{(k)})^T v = -\nabla g(x^{(k)})^T \nabla^2 g(x^{(k)})^{-1} \nabla g(x^{(k)}) > 0$$



- if Newton step is not descent direction, replace it with descent direction
- simplest choice is $v = -\nabla g(x^{(k)})$ or $v = -(\nabla^2 g(x_k) + \mu_k I)^{-1} \nabla g(x^{(k)})$

References and further readings

- S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares,* Cambridge University Press, 2018.

- L. Vandenberghe. *EE133A lecture notes,* Univ. of California, Los Angeles. (`http://www.seas.ucla.edu/~vandenbe/ee133a.html`)