Lecture Notes to Accompany

Scientific Computing

An Introductory Survey

Second Edition

by Michael T. Heath

Chapter 6

Optimization

Optimization

Given function $f: \mathbb{R}^n \to \mathbb{R}$, and set $S \subseteq \mathbb{R}^n$, find $\boldsymbol{x}^* \in S$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$

$\boldsymbol{x}^*$ called *minimizer* or *minimum* of f

Suffices to consider only minimization, since maximum of f is minimum of $-f$

Objective function f usually differentiable, may be linear or nonlinear

Constraint set S defined by system of equations and inequalities that may be linear or nonlinear

Points $\boldsymbol{x} \in S$ called *feasible* points

If $S = \mathbb{R}^n$, problem is *unconstrained*

Optimization Problems

General continuous optimization problem:

$$\min f(\boldsymbol{x}) \text{ subject to } \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{o} \text{ and } \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{o},$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g}\colon \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{h}\colon \mathbb{R}^n \to \mathbb{R}^p$

Linear programming: f, $\boldsymbol{g}$, and $\boldsymbol{h}$ all linear

Nonlinear programming: nonlinear objective or nonlinear constraints, or both

Examples: Optimization Problems

Minimize weight of structure subject to constraint on its strength, or maximize its strength subject to constraint on its weight

Minimize cost of diet subject to nutritional constraints

Minimize surface area of cylinder subject to constraint on its volume:

$$\min_{x_1,x_2} f(x_1,x_2) = 2\pi x_1(x_1+x_2)$$

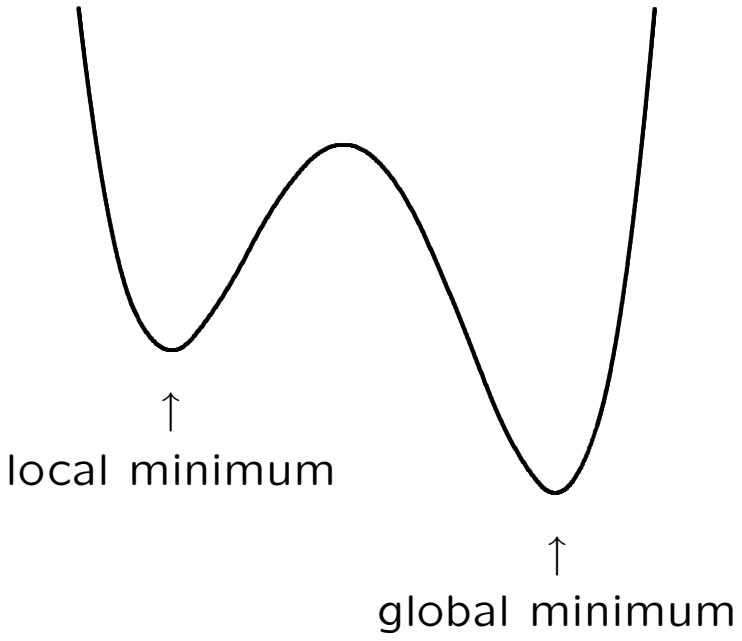
$$\text{subject to } g(x_1,x_2) = \pi x_1^2 x_2 - V = 0,$$

where x_1 and x_2 are radius and height of cylinder, and V is required volume

Local vs Global Optimization

$\boldsymbol{x}^* \in S$ is *global minimum* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$

$\boldsymbol{x}^* \in S$ is *local minimum* if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for all feasible $\boldsymbol{x}$ in some neighborhood of $\boldsymbol{x}^*$



Global Optimization

Finding, or even verifying, global minimum is difficult, in general

Most optimization methods designed to find local minimum, which may or may not be global minimum

If global minimum desired, can try several widely separated starting points and see if all produce same result

For some problems, such as linear programming, global optimization tractable

Existence of Minimum

If f is continuous on *closed* and *bounded* set $S \subseteq \mathbb{R}^n$, then f has global minimum on S

If S is not closed or is unbounded, then f may have no local or global minimum on S

Continuous function f on unbounded set $S \subseteq \mathbb{R}^n$ is *coercive* if

$$\lim_{\|\boldsymbol{x}\| \to \infty} f(\boldsymbol{x}) = +\infty,$$

i.e., $f(\boldsymbol{x})$ must be large whenever $\|\boldsymbol{x}\|$ is large

If f coercive on closed, unbounded set $S \subseteq \mathbb{R}^n$, then f has global minimum on S

Level Sets

Level set for function $f: S \subseteq \mathbb{R}^n \to \mathbb{R}$ is set of all points in S for which f has some given constant value (also called *contour line*)

For given $\gamma \in \mathbb{R}$, *sublevel set* is

$$L_\gamma = \{\boldsymbol{x} \in S : \, f(\boldsymbol{x}) \leq \gamma\}$$

If continuous function f on $S \subseteq \mathbb{R}^n$ has nonempty sublevel set that is closed and bounded, then f has global minimum on S

If S is unbounded, then f is coercive on S if, and only if, *all* its sublevel sets are bounded

Uniqueness of Minimum

Set $S \subseteq \mathbb{R}^n$ is *convex* if it contains line segment between any two of its points

Function $f\colon S \subseteq \mathbb{R}^n \to \mathbb{R}$ is *convex* on convex set S if its graph along any line segment in S lies *on or below* chord connecting function values at endpoints of segment

Any local minimum of convex function f on convex set $S \subseteq \mathbb{R}^n$ is global minimum of f on S

Any local minimum of *strictly* convex function f on convex set $S \subseteq \mathbb{R}^n$ is *unique* global minimum of f on S

First-Order Optimality Condition

For function of one variable, find extremum by differentiating function and setting derivative to zero

Generalization to function of n variables is to find *critical point*, i.e., solution of nonlinear system

$$\nabla f(\boldsymbol{x}) = \boldsymbol{o},$$

where $\nabla f(\boldsymbol{x})$ is *gradient* vector of f, whose ith component is $\partial f(\boldsymbol{x})/\partial x_i$

For continuously differentiable $f\colon S \subseteq \mathbb{R}^n \to \mathbb{R}$, any interior point $\boldsymbol{x}^*$ of S at which f has local minimum must be critical point of f

But not all critical points are minima: can also be maximum or saddle point

Second-Order Optimality Condition

For twice continuously differentiable $f: S \subseteq \mathbb{R}^n \to \mathbb{R}$, can distinguish among critical points by considering *Hessian* matrix $\boldsymbol{H}_f(x)$ defined by

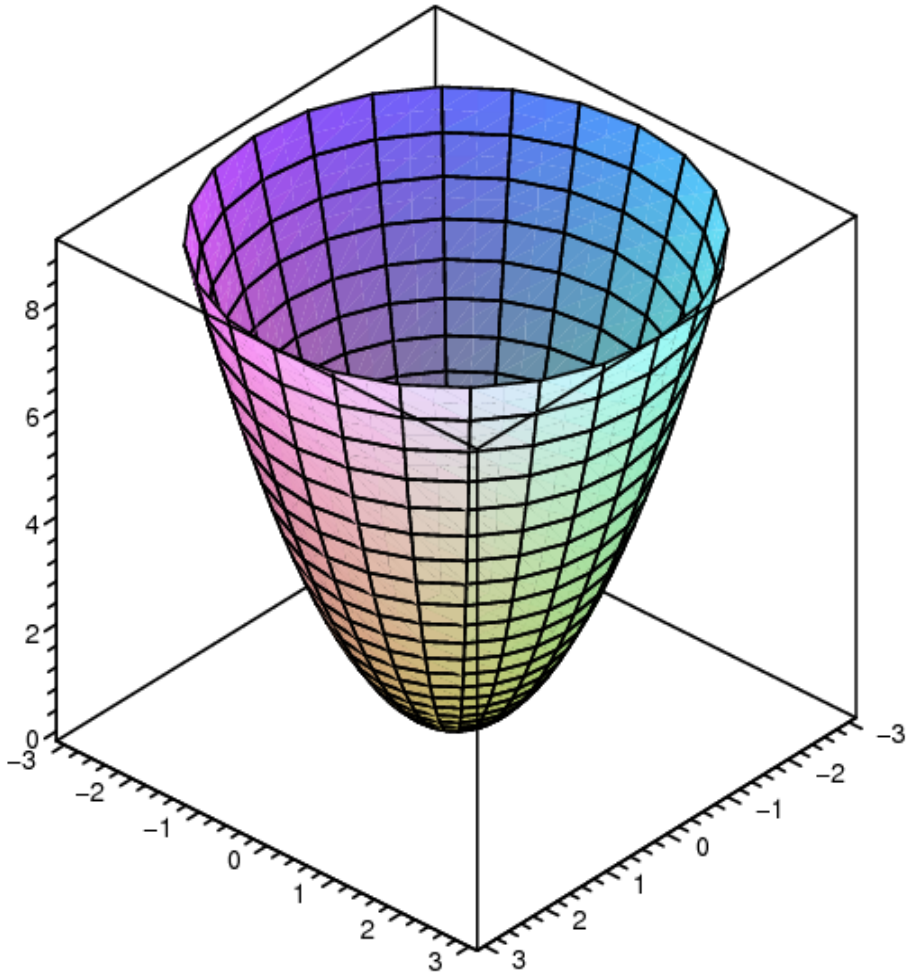
$$\{\boldsymbol{H}_f(x)\}_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j},$$

which is symmetric

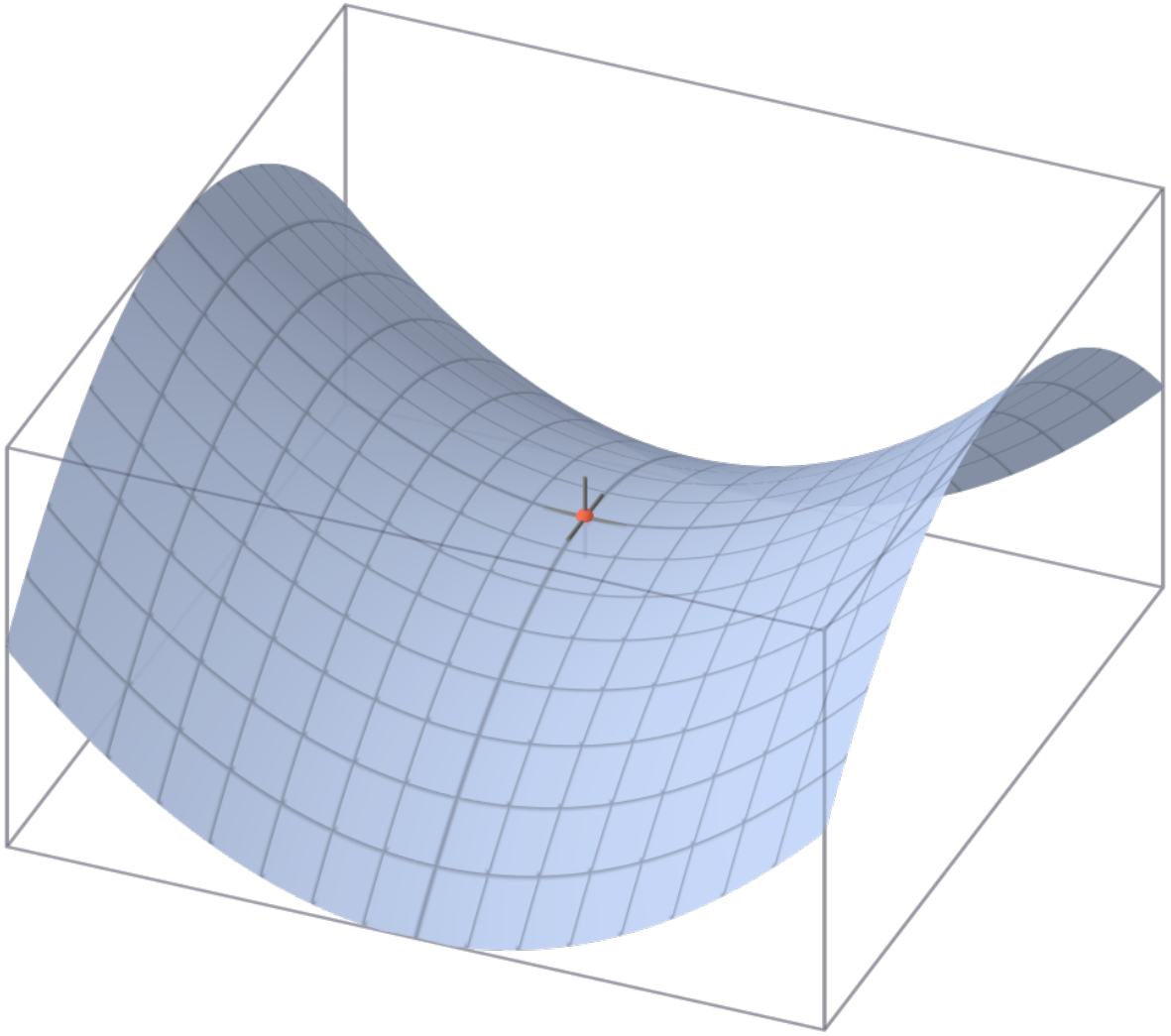
At critical point $\boldsymbol{x}^*$, if $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is

- positive definite, then $\boldsymbol{x}^*$ is minimum of f
- negative definite, then $\boldsymbol{x}^*$ is maximum of f
- indefinite, then $\boldsymbol{x}^*$ is saddle point of f
- singular, then various pathological situations possible

$z = x^2 + y^2$ has a minimum



$z = x^2 - y^2$ has a saddle point



Constrained Optimality

If problem is constrained, need be concerned only with *feasible* directions

For equality-constrained problem

$$\min f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{o},$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{g}\colon \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, necessary condition for feasible point $\boldsymbol{x}^*$ to be solution is that negative gradient of f lie in space spanned by constraint normals, i.e.,

$$-\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_g^T(\boldsymbol{x}^*)\boldsymbol{\lambda},$$

where $\boldsymbol{J}_g$ is Jacobian matrix of $\boldsymbol{g}$, and $\boldsymbol{\lambda}$ is vector of *Lagrange multipliers*

This condition says we cannot reduce objective function without violating constraints

Constrained Optimality, continued

Lagrangian function $\mathcal{L}\colon \mathbb{R}^{n+m} \to \mathbb{R}$, defined by

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}),$$

and its gradient and Hessian given by

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix}$$

and

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) & \boldsymbol{J}_g^T(\boldsymbol{x}) \\ \boldsymbol{J}_g(\boldsymbol{x}) & \boldsymbol{O} \end{bmatrix},$$

where

$$\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x})$$

Together, necessary condition and feasibility imply critical point of Lagrangian function,

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} = \boldsymbol{o}$$

Constrained Optimality, continued

Hessian of Lagrangian is symmetric, but not positive definite, so critical point of $\mathcal{L}$ is saddle point rather than minimum or maximum

Critical point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ of $\mathcal{L}$ is constrained minimum of f if $\boldsymbol{B}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is positive definite on *null space* of $\boldsymbol{J}_g(\boldsymbol{x}^*)$

If columns of $\boldsymbol{Z}$ form basis for null space, then test *projected* Hessian $\boldsymbol{Z}^T\boldsymbol{B}\boldsymbol{Z}$ for positive definiteness

If inequalities present, then KKT optimality conditions also require nonnegativity of Lagrange multipliers corresponding to inequalities, and complementarity condition

Sensitivity and Conditioning

Although problems of minimizing function and solving equation are closely related, their sensitivities differ

In one dimension, absolute condition number of root x^* of equation $f(x) = 0$ is $1/|f'(x^*)|$, so if $|f(\widehat{x})| \leq \epsilon$, then $|\widehat{x} - x^*|$ may be as large as $\epsilon/|f'(x^*)|$

For minimizing f, Taylor series expansion

$$\begin{aligned} f(\widehat{x}) &= f(x^* + h) \\ &= f(x^*) + f'(x^*)h + \tfrac{1}{2} f''(x^*)h^2 + \mathcal{O}(h^3) \end{aligned}$$

shows that, since $f'(x^*) = 0$, if $|f(\widehat{x}) - f(x^*)| \leq \epsilon$, then $|\widehat{x} - x^*|$ may be as large as $\sqrt{2\epsilon/|f''(x^*)|}$

Thus, based on function values alone, minima can be computed to only about half precision

Unimodality

For minimizing function of one variable, need "bracket" for solution analogous to sign change for nonlinear equation

Real-valued function f is *unimodal* on interval $[a, b]$ if there is unique $x^* \in [a, b]$ such that $f(x^*)$ is minimum of f on $[a, b]$, and f is strictly decreasing for $x \leq x^*$, strictly increasing for $x^* \leq x$

This property enables discarding portions of interval based on sample function values, analogous to interval bisection

Golden Section Search

Suppose f is unimodal on $[a, b]$, and let x_1 and x_2 be two points within interval, with $x_1 < x_2$

Evaluating and comparing $f(x_1)$ and $f(x_2)$, can discard either $(x_2, b]$ or $[a, x_1)$, with minimum known to lie in remaining subinterval

To repeat process, need to compute only one new function evaluation

To reduce length of interval by fixed fraction at each iteration, each new pair of points must have same relationship with respect to new interval that previous pair had with respect to previous interval

Golden Section Search, continued

To accomplish this, choose relative positions of two points as τ and $1 - \tau$, where $\tau^2 = 1 - \tau$, so $\tau = (\sqrt{5} - 1)/2 \approx 0.618$ and $1 - \tau \approx 0.382$

Whichever subinterval is retained, its length will be τ relative to previous interval, and interior point retained will be at position either τ or $1 - \tau$ relative to new interval

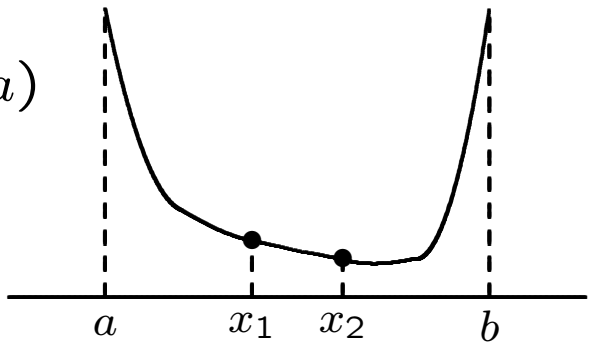
Need to compute only one new function value, at complementary point, to continue iteration

This choice of sample points called *golden section search*

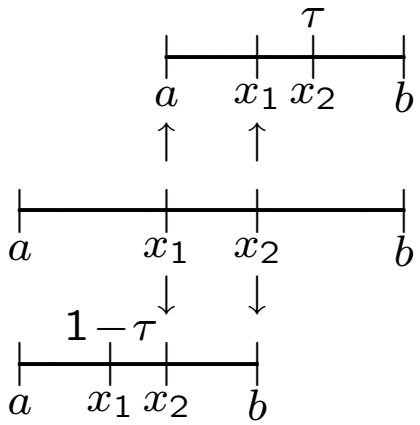
Golden section search is safe but convergence rate only linear, with constant $C \approx 0.618$

Golden Section Search, continued

$\tau = (\sqrt{5} - 1)/2$
$x_1 = a + (1 - \tau)(b - a)$
$f_1 = f(x_1)$
$x_2 = a + \tau(b - a)$
$f_2 = f(x_2)$



while $((b - a) > tol)$ **do**
  if $(f_1 > f_2)$ **then**
    $a = x_1$
    $x_1 = x_2$
    $f_1 = f_2$
    $x_2 = a + \tau(b - a)$
    $f_2 = f(x_2)$
  else
    $b = x_2$
    $x_2 = x_1$
    $f_2 = f_1$
    $x_1 = a + (1 - \tau)(b - a)$
    $f_1 = f(x_1)$

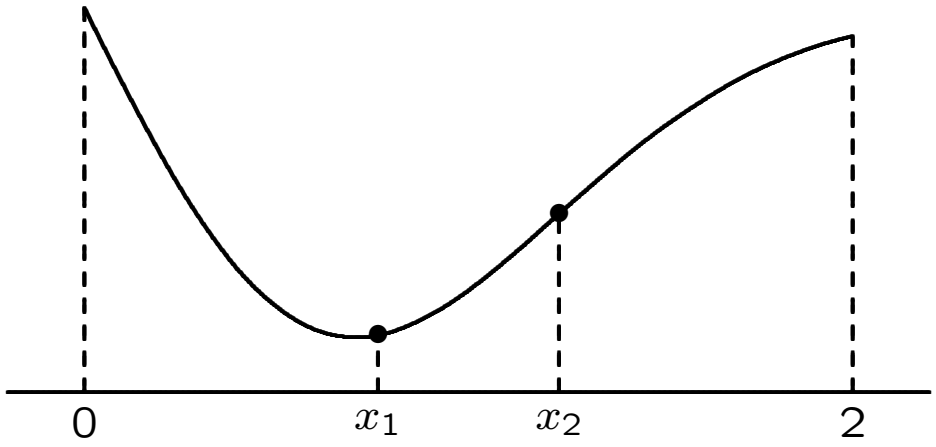


Example: Golden Section Search

Use golden section search to minimize

$$f(x) = 0.5 - x\exp(-x^2)$$

x_1	f_1	x_2	f_2
0.764	0.074	1.236	0.232
0.472	0.122	0.764	0.074
0.764	0.074	0.944	0.113
0.652	0.074	0.764	0.074
0.584	0.085	0.652	0.074
0.652	0.074	0.695	0.071
0.695	0.071	0.721	0.071
0.679	0.072	0.695	0.071
0.695	0.071	0.705	0.071
0.705	0.071	0.711	0.071



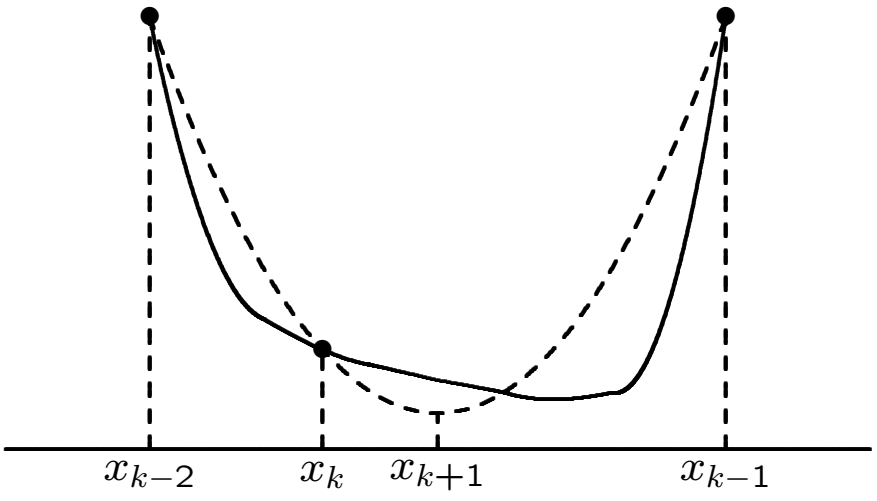
Successive Parabolic Interpolation

Fit quadratic polynomial to three function values

Take minimum of quadratic to be new approximation to minimum of function

New point replaces oldest of three previous points and process repeated until convergence

Convergence rate of successive parabolic interpolation is superlinear, with $r \approx 1.324$

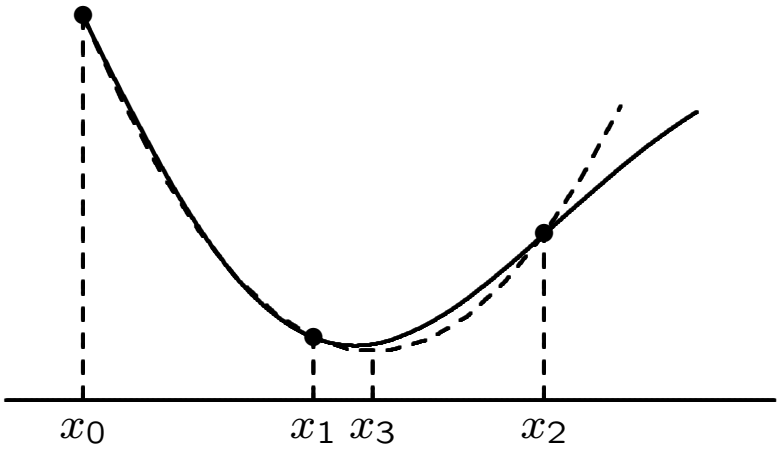


Example: Successive Parabolic Interpolation

Use successive parabolic interpolation to minimize

$$f(x) = 0.5 - x\exp(-x^2)$$

x_k	$f(x_k)$
0.000	0.500
0.600	0.081
1.200	0.216
0.754	0.073
0.721	0.071
0.692	0.071
0.707	0.071



Newton's Method

Another local quadratic approximation is truncated Taylor series

$$f(x+h) \approx f(x) + f'(x)h + \frac{f''(x)}{2}h^2$$

By differentiation, minimum of this quadratic function of h is given by $h = -f'(x)/f''(x)$

Suggests iteration scheme

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

which is Newton's method for solving nonlinear equation $f'(x) = 0$

Newton's method for finding minimum normally has quadratic convergence rate, but must be started close enough to solution

Example: Newton's Method

Use Newton's method to minimize

$$f(x) = 0.5 - x\exp(-x^2)$$

First and second derivatives of f are given by

$$f'(x) = (2x^2 - 1)\exp(-x^2)$$

and

$$f''(x) = 2x(3 - 2x^2)\exp(-x^2),$$

so Newton iteration for zero of f' given by

$$x_{k+1} = x_k - (2x_k^2 - 1)/(2x_k(3 - 2x_k^2))$$

Using starting guess $x_0 = 1$, obtain

x_k	$f(x_k)$
1.000	0.132
0.500	0.111
0.700	0.071
0.707	0.071

Safeguarded Methods

As with nonlinear equations in one dimension, slow-but-sure and fast-but-risky optimization methods can be combined to provide both safety and efficiency

Most library routines for one-dimensional optimization based on hybrid approach

Popular combination is golden section search and successive parabolic interpolation, for which no derivatives required

Direct Search Methods

Direct search methods for multidimensional optimization make no use of function values other than comparing them

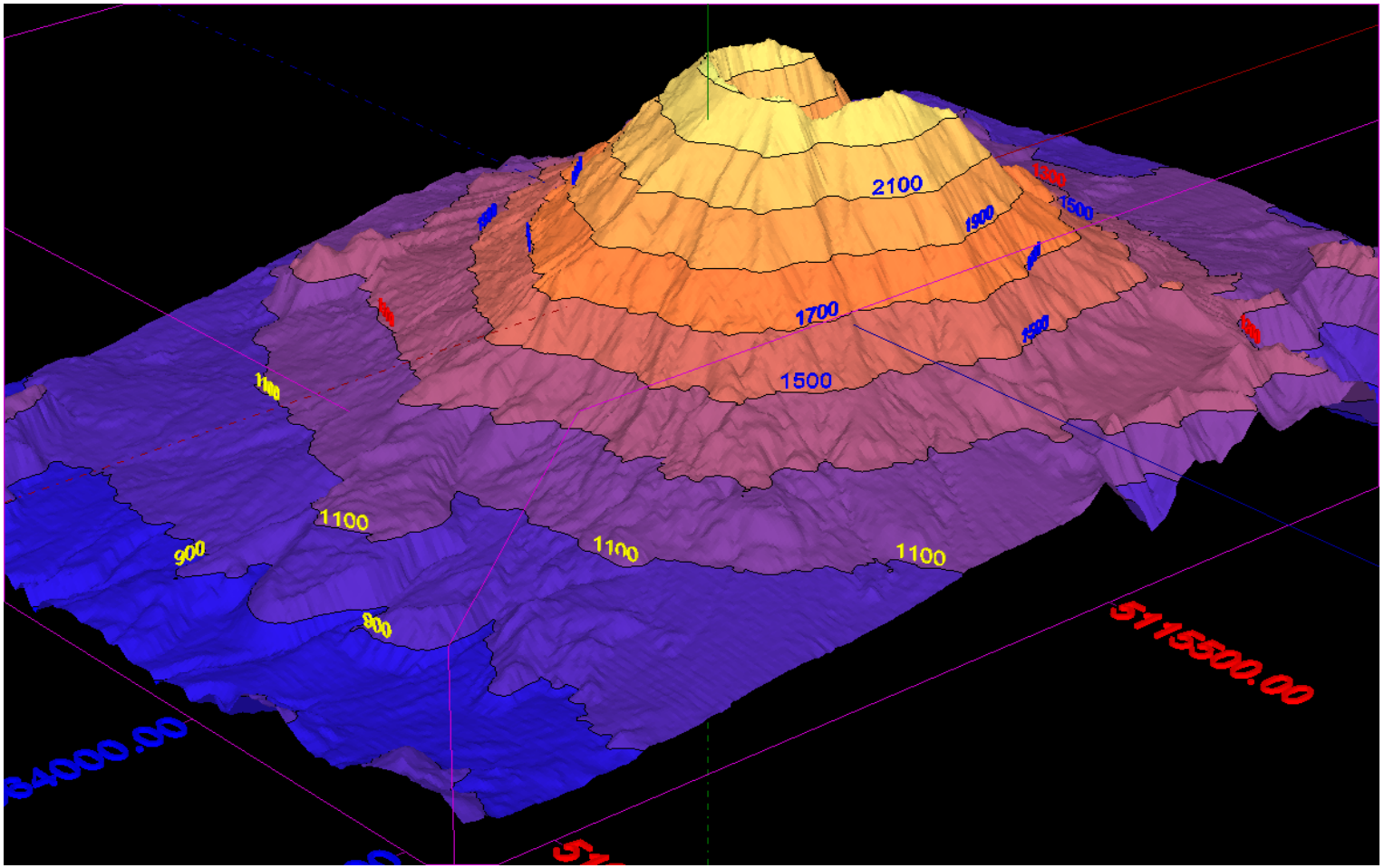
For minimizing function f of n variables, Nelder-Mead method begins with $n+1$ starting points, forming *simplex* in $\mathbb{R}^n$

Then move to new point along straight line from current point having highest function value through centroid of points

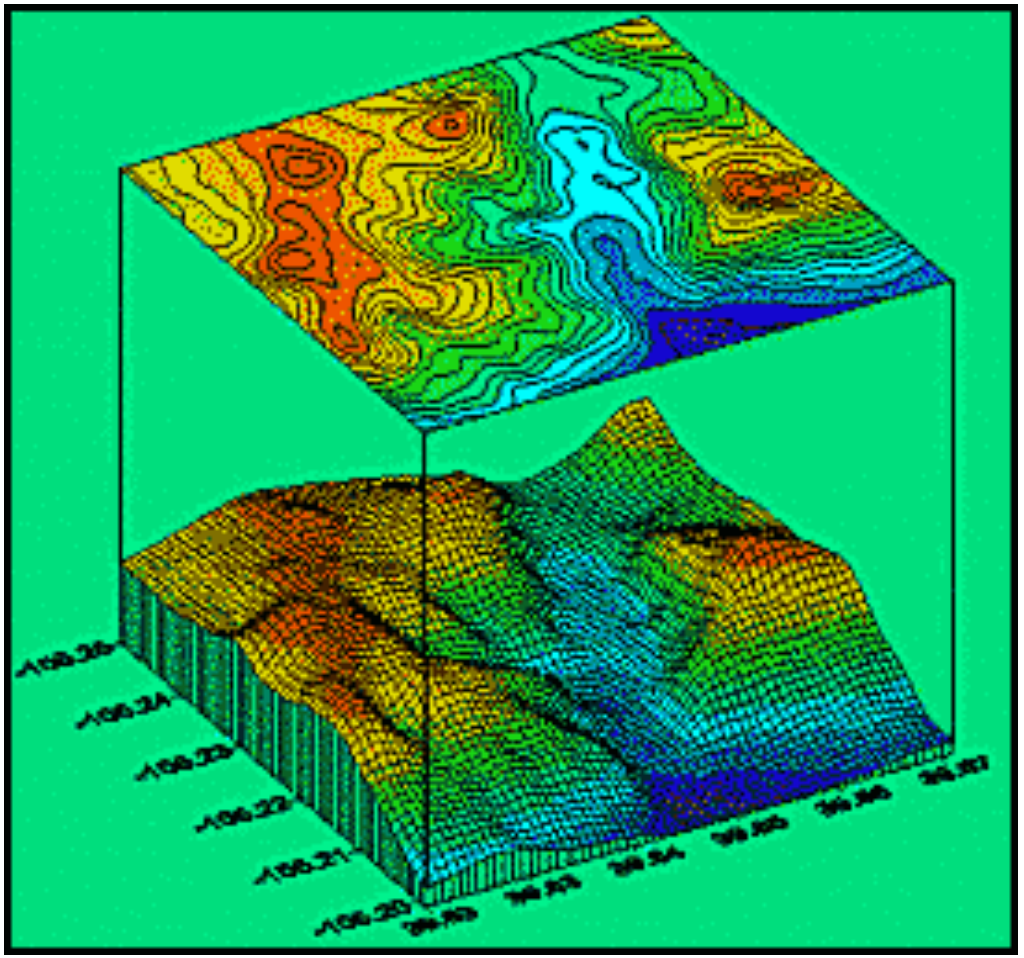
New point replaces worst point, and process repeated

Direct search methods useful for nonsmooth functions or for small n, but expensive for larger n

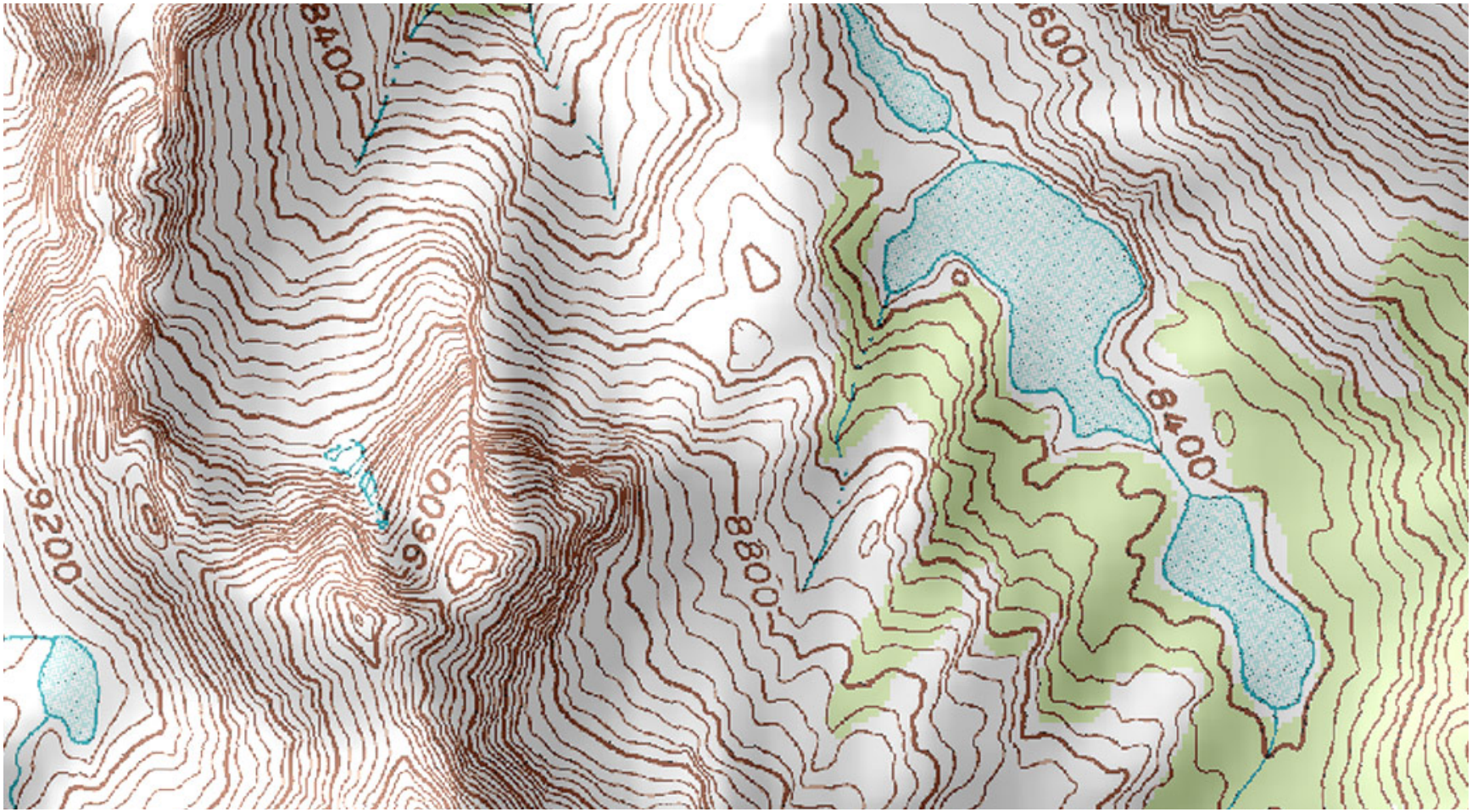
Contour Maps



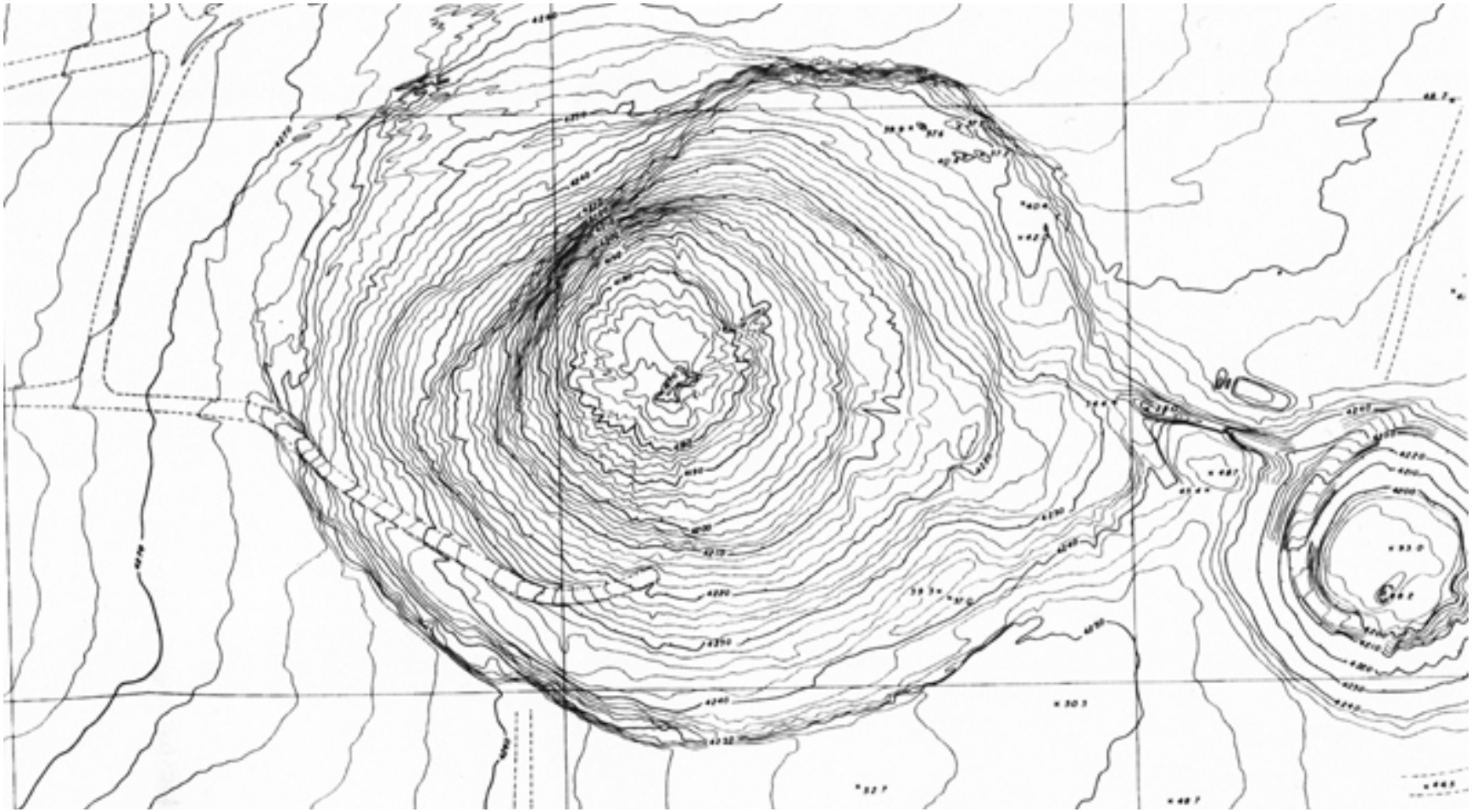
Contour Maps



Contour Map



Contour Map



Steepest Descent Method

Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be real-valued function of n real variables

At any point $\boldsymbol{x}$ where gradient vector is nonzero, negative gradient, $-\nabla f(\boldsymbol{x})$, points downhill toward lower values of function f

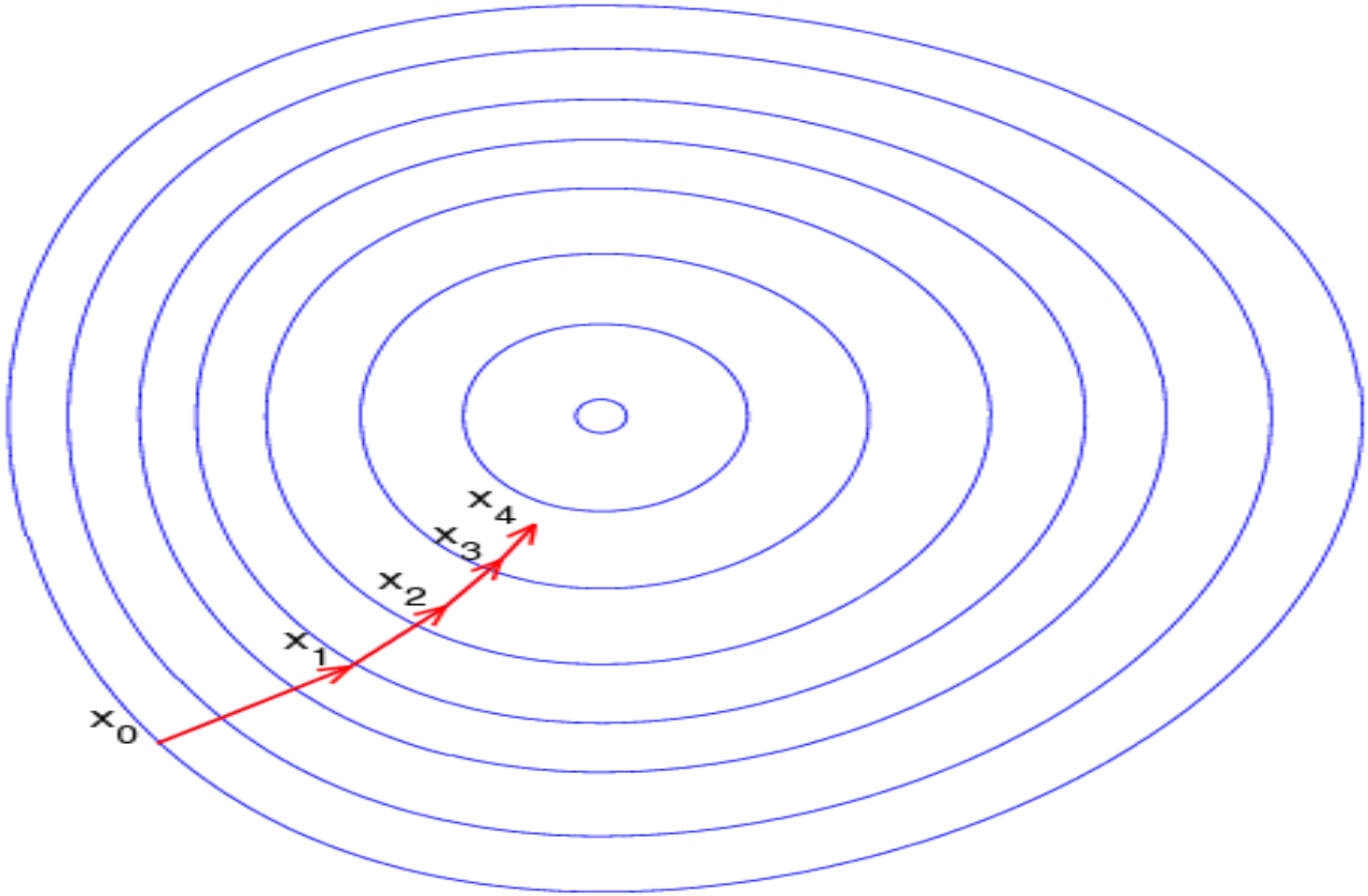
In fact, $-\nabla f(\boldsymbol{x})$ is locally direction of steepest descent: function decreases more rapidly along direction of negative gradient than along any other

Steepest descent method: starting from initial guess $\boldsymbol{x}_0$, successive approximate solutions given by

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

where α_k is *line search* parameter that determines how far to go in given direction

Steepest Descent



Steepest Descent, continued

Given descent direction, such as negative gradient, determining appropriate value for α_k at each iteration is one-dimensional minimization problem

$$\min_{\alpha_k} f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k))$$

that can be solved by methods already discussed

Steepest descent method is very reliable: can always make progress provided gradient is nonzero

But method is myopic in its view of function's behavior, and resulting iterates can zigzag back and forth, making very slow progress toward solution

In general, convergence rate of steepest descent is only linear, with constant factor that can be arbitrarily close to 1

Example: Steepest Descent

Use steepest descent method to minimize

$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

Gradient is given by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$$

Taking $\boldsymbol{x}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, we have $\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

Performing line search along negative gradient direction, i.e.,

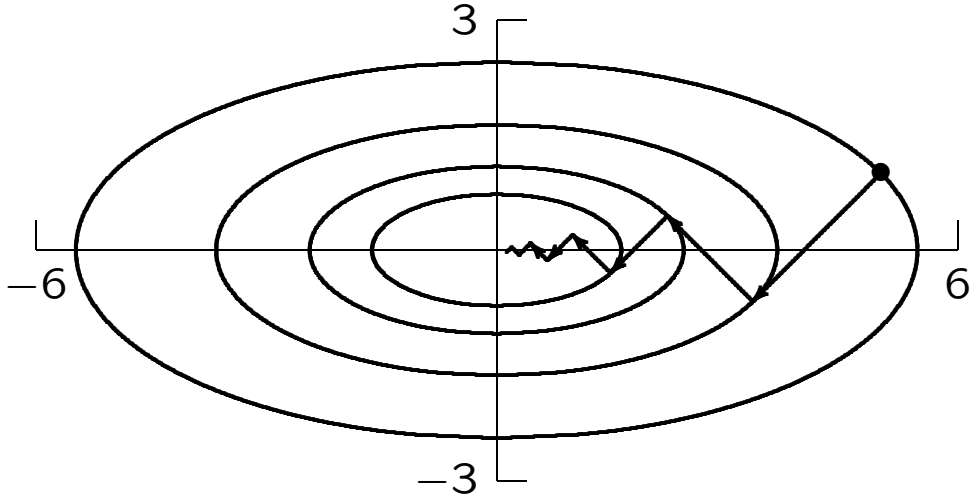
$$\min_{\alpha_0} f(\boldsymbol{x}_0 - \alpha_0 \nabla f(\boldsymbol{x}_0)),$$

exact minimum along line is given by $\alpha_0 = 1/3$, so next approximation is

$$\boldsymbol{x}_1 = \begin{bmatrix} 3.333 \\ -0.667 \end{bmatrix}$$

Example Continued

$\boldsymbol{x}_k$		$f(\boldsymbol{x}_k)$	$\nabla f(\boldsymbol{x}_k)$	
5.000	1.000	15.000	5.000	5.000
3.333	-0.667	6.667	3.333	-3.333
2.222	0.444	2.963	2.222	2.222
1.481	-0.296	1.317	1.481	-1.481
0.988	0.198	0.585	0.988	0.988
0.658	-0.132	0.260	0.658	-0.658
0.439	0.088	0.116	0.439	0.439
0.293	-0.059	0.051	0.293	-0.293
0.195	0.039	0.023	0.195	0.195
0.130	-0.026	0.010	0.130	-0.130



Newton's Method

Broader view can be obtained by local quadratic approximation, which is equivalent to Newton's method

In multidimensional optimization, we seek zero of gradient, so Newton iteration has form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{H}_f^{-1}(\boldsymbol{x}_k)\nabla f(\boldsymbol{x}_k),$$

where $\boldsymbol{H}_f(\boldsymbol{x})$ is *Hessian* matrix of second partial derivatives of f,

$$\{\boldsymbol{H}_f(\boldsymbol{x})\}_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}$$

Newton's Method, continued

Do not explicitly invert Hessian matrix, but instead solve linear system

$$\boldsymbol{H}_f(\boldsymbol{x}_k)\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$$

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

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

Convergence rate of Newton's method for minimization is normally quadratic

As usual, Newton's method is unreliable unless started close enough to solution

Example: Newton's Method

Use Newton's method to minimize

$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

Gradient and Hessian are given by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{H}_f(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$$

Taking $\boldsymbol{x}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, we have $\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

Linear system for Newton step is

$$\begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \boldsymbol{s}_0 = \begin{bmatrix} -5 \\ -5 \end{bmatrix},$$

so next approximate solution is

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

which is exact solution for this problem, as expected for quadratic function

Newton's Method, continued

Line search parameter unnecessary with Newton's method, since quadratic model determines length, as well as direction, of step to next approximate solution

When started far from solution, however, it may still be advisable to perform line search along direction of Newton step $\boldsymbol{s}_k$ to make method more robust (*damped* Newton)

Once iterations near solution, then $\alpha_k = 1$ should suffice for subsequent iterations

Newton's Method, continued

If objective function f has continuous second partial derivatives, then Hessian matrix $\boldsymbol{H}_f$ is symmetric, and near minimum it is positive definite

Thus, linear system for step to next iterate can be solved in only about half of work required for LU factorization

Far from minimum, $\boldsymbol{H}_f(\boldsymbol{x}_k)$ may not be positive definite, so Newton step $\boldsymbol{s}_k$ may not even be descent direction for function, i.e., we may not have

$$\nabla f(\boldsymbol{x}_k)^T \boldsymbol{s}_k < 0$$

In this case, alternative descent direction can be computed, such as negative gradient or direction of negative curvature, and line search performed

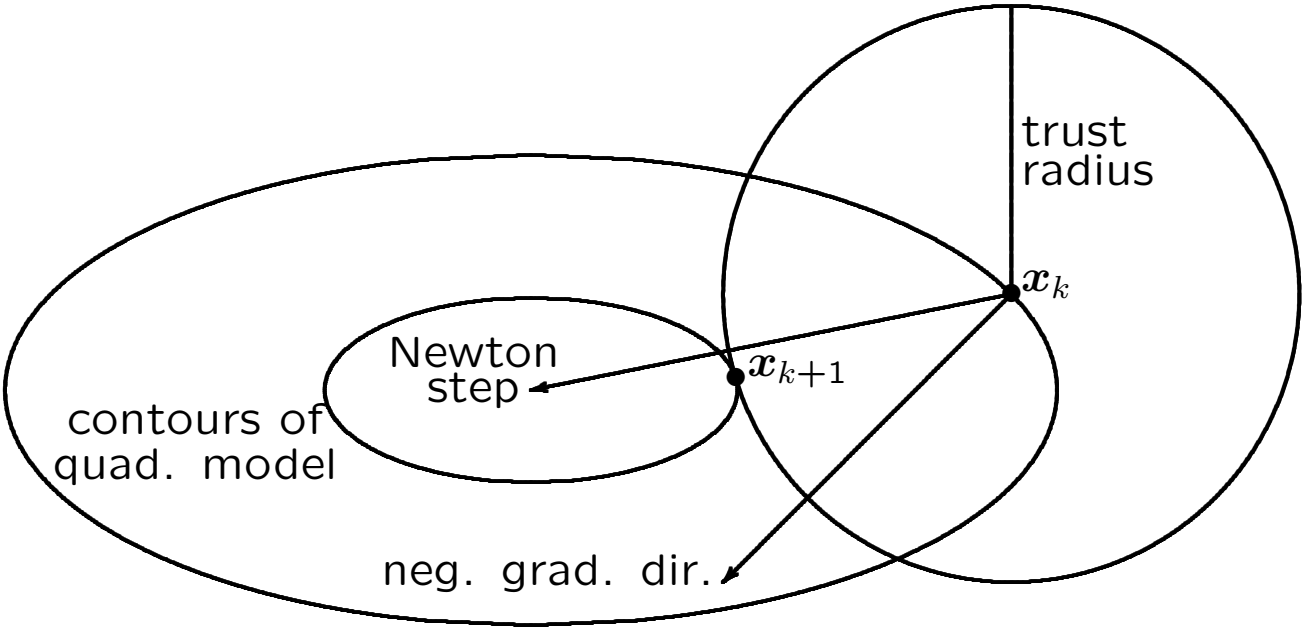
Trust Region Methods

Alternative to line search is *trust region method*, in which approximate solution is constrained to lie within region where quadratic model is sufficiently accurate

If current trust radius is binding, minimizing quadratic model function subject to this constraint may modify direction as well as length of Newton step

Accuracy of quadratic model is assessed by comparing actual decrease in objective function with that predicted by quadratic model, and trust radius is increased or decreased accordingly

Trust Region Methods, continued



Quasi-Newton Methods

Newton's method costs $\mathcal{O}(n^3)$ arithmetic and $\mathcal{O}(n^2)$ scalar function evaluations per iteration for dense problem

Many variants of Newton's method improve reliability and reduce overhead

Quasi-Newton methods have form

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

where α_k is line search parameter and $\boldsymbol{B}_k$ is approximation to Hessian matrix

Many quasi-Newton methods are more robust than Newton's method, superlinearly convergent, and have lower overhead per iteration, which often more than offsets their slower convergence rate

Secant Updating Methods

Could use Broyden's method to seek zero of gradient, but this would not preserve symmetry of Hessian matrix

Several secant updating formulas have been developed for minimization that not only preserve symmetry in approximate Hessian matrix, but also preserve positive definiteness

Symmetry reduces amount of work required by about half, while positive definiteness guarantees that quasi-Newton step will be descent direction

BFGS Method

One of most effective secant updating methods for minimization is BFGS:

$x_0 =$ initial guess
$B_0 =$ initial Hessian approximation
for $k = 0, 1, 2, \ldots$
  Solve $\boldsymbol{B}_k \boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k)$ for $\boldsymbol{s}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$
  $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$
  $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + (\boldsymbol{y}_k \boldsymbol{y}_k^T)/(\boldsymbol{y}_k^T \boldsymbol{s}_k)$
    $- (\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{B}_k)/(\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k)$
end

BFGS Method, continued

In practice, factorization of $\boldsymbol{B}_k$ is updated rather than $\boldsymbol{B}_k$ itself, so linear system for $\boldsymbol{s}_k$ can be solved at cost of $\mathcal{O}(n^2)$ rather than $\mathcal{O}(n^3)$ work

Unlike Newton's method for minimization, no second derivatives are required

Can start with $\boldsymbol{B}_0 = \boldsymbol{I}$, so initial step is along negative gradient, and then second derivative information is gradually built up in approximate Hessian matrix over successive iterations

BFGS normally has superlinear convergence rate, even though approximate Hessian does not necessarily converge to true Hessian

Line search can be used to enhance effectiveness of method

Example: BFGS Method

Use BFGS to minimize

$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

Gradient is given by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$$

Taking $\boldsymbol{x}_0 = [5 \quad 1]^T$ and $\boldsymbol{B}_0 = \boldsymbol{I}$, initial step is negative gradient, so

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}$$

Updating approximate Hessian using BFGS formula, obtain

$$\boldsymbol{B}_1 = \begin{bmatrix} 0.667 & 0.333 \\ 0.333 & 0.667 \end{bmatrix}$$

Then new step is computed and process is repeated

Example: BFGS Method

$\boldsymbol{x}_k$		$f(\boldsymbol{x}_k)$	$\nabla f(\boldsymbol{x}_k)$	
5.000	1.000	15.000	5.000	5.000
0.000	-4.000	40.000	0.000	-20.000
-2.222	0.444	2.963	-2.222	2.222
0.816	0.082	0.350	0.816	0.408
-0.009	-0.015	0.001	-0.009	-0.077
-0.001	0.001	0.000	-0.001	0.005

Increase in function value can be avoided by using line search, which generally enhances convergence

For quadratic objective function, BFGS with exact line search finds exact solution in at most n iterations, where n is dimension of problem

Conjugate Gradient Method

Another method that does not require explicit second derivatives, and does not even store approximation to Hessian matrix, is *conjugate gradient* (CG) method

CG generates sequence of conjugate search directions, implicitly accumulating information about Hessian matrix

For quadratic objective function, CG is theoretically exact after at most n iterations, where n is dimension of problem

CG is effective for general unconstrained minimization as well

Conjugate Gradient Method, continued

$\boldsymbol{x}_0 =$ initial guess
$\boldsymbol{g}_0 = \nabla f(\boldsymbol{x}_0)$
$\boldsymbol{s}_0 = -\boldsymbol{g}_0$
for $k = 0, 1, 2, \ldots$
    Choose α_k to minimize $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k)$
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k$
    $\boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{x}_{k+1})$
    $\beta_{k+1} = (\boldsymbol{g}_{k+1}^T \boldsymbol{g}_{k+1})/(\boldsymbol{g}_k^T \boldsymbol{g}_k)$
    $\boldsymbol{s}_{k+1} = -\boldsymbol{g}_{k+1} + \beta_{k+1} \boldsymbol{s}_k$
end

Alternative formula for β_{k+1} is

$$\beta_{k+1} = ((\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)^T \boldsymbol{g}_{k+1})/(\boldsymbol{g}_k^T \boldsymbol{g}_k)$$

Example: Conjugate Gradient Method

Use CG method to minimize

$$f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2$$

Gradient is given by

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix}$$

Taking $\boldsymbol{x}_0 = [\,5 \quad 1\,]^T$, initial search direction is negative gradient,

$$\boldsymbol{s}_0 = -\boldsymbol{g}_0 = -\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} -5 \\ -5 \end{bmatrix}$$

Exact minimum along line is given by $\alpha_0 = 1/3$, so next approximation is $\boldsymbol{x}_1 = [\,3.333 \quad -0.667\,]^T$, and we compute new gradient,

$$\boldsymbol{g}_1 = \nabla f(\boldsymbol{x}_1) = \begin{bmatrix} 3.333 \\ -3.333 \end{bmatrix}$$

Example Continued

So far there is no difference from steepest descent method

At this point, however, rather than search along new negative gradient, we compute instead

$$\beta_1 = (\boldsymbol{g}_1^T \boldsymbol{g}_1)/(\boldsymbol{g}_0^T \boldsymbol{g}_0) = 0.444,$$

which gives as next search direction

$$\boldsymbol{s}_1 = -\boldsymbol{g}_1 + \beta_1 \boldsymbol{s}_0 = \begin{bmatrix} -3.333 \\ 3.333 \end{bmatrix} + 0.444 \begin{bmatrix} -5 \\ -5 \end{bmatrix}$$

$$= \begin{bmatrix} -5.556 \\ 1.111 \end{bmatrix}$$

Minimum along this direction is given by $\alpha_1 = 0.6$, which gives exact solution at origin, as expected for quadratic function

Truncated Newton Methods

Another way to reduce work in Newton-like methods is to solve linear system for Newton step by iterative method

Small number of iterations may suffice to produce step as useful as true Newton step, especially far from overall solution, where true Newton step may be unreliable anyway

Good choice for linear iterative solver is CG method, which gives step intermediate between steepest descent and Newton-like step

Since only matrix-vector products are required, explicit formation of Hessian matrix can be avoided by using finite difference of gradient along given vector

Nonlinear Least Squares

Given data (t_i, y_i), find vector $\boldsymbol{x}$ of parameters that gives "best fit" in least squares sense to model function $f(t, \boldsymbol{x})$

If we define components of residual function

$$r_i(\boldsymbol{x}) = y_i - f(t_i, \boldsymbol{x}), \quad i = 1, \ldots, m,$$

then we want to minimize $\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})$

Gradient vector of ϕ is

$$\nabla\phi(\boldsymbol{x}) = \boldsymbol{J}^T(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}),$$

and Hessian matrix is

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}^T(\boldsymbol{x})\boldsymbol{J}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_i(\boldsymbol{x}),$$

where $\boldsymbol{J}(\boldsymbol{x})$ is Jacobian of $\boldsymbol{r}(\boldsymbol{x})$, and $\boldsymbol{H}_i(\boldsymbol{x})$ is Hessian of $r_i(\boldsymbol{x})$

Nonlinear Least Squares, continued

Linear system for Newton step is

$$\left(\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k) + \sum_{i=1}^{m} r_i(\boldsymbol{x}_k)\boldsymbol{H}_i(\boldsymbol{x}_k) \right) \boldsymbol{s}_k =$$

$$-\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k)$$

m Hessian matrices $\boldsymbol{H}_i$ are usually inconvenient and expensive to compute

Moreover, in $\boldsymbol{H}_\phi$ each $\boldsymbol{H}_i$ is multiplied by residual component r_i, which is small at solution if fit of model function to data is good

Gauss-Newton Method

This motivates Gauss-Newton method for nonlinear least squares, in which second-order term is dropped and linear system

$$\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k)$$

is solved for approximate Newton step $\boldsymbol{s}_k$ at each iteration

This is system of normal equations for linear least squares problem

$$\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{s}_k \cong -\boldsymbol{r}(\boldsymbol{x}_k),$$

which can be solved more reliably by orthogonal factorization

Next approximate solution is then given by

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

and process is repeated until convergence

Example: Gauss-Newton Method

Use Gauss-Newton method to fit nonlinear model function

$$f(t, \boldsymbol{x}) = x_1 \exp(x_2 t)$$

to data

t	0.0	1.0	2.0	3.0
y	2.0	0.7	0.3	0.1

For this model function, entries of Jacobian matrix of residual function $\boldsymbol{r}$ are given by

$$\{\boldsymbol{J}(\boldsymbol{x})\}_{i,1} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_1} = -\exp(x_2 t_i),$$

$$\{\boldsymbol{J}(\boldsymbol{x})\}_{i,2} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_2} = -x_1 t_i \exp(x_2 t_i)$$

Example Continued

If we take $\boldsymbol{x}_0 = [\,1 \quad 0\,]^T$, then Gauss-Newton step $\boldsymbol{s}_0$ is given by linear least squares problem

$$\begin{bmatrix} -1 & 0 \\ -1 & -1 \\ -1 & -2 \\ -1 & -3 \end{bmatrix} \boldsymbol{s}_0 \cong \begin{bmatrix} -1 \\ 0.3 \\ 0.7 \\ 0.9 \end{bmatrix},$$

whose solution is $\boldsymbol{s}_0 = \begin{bmatrix} 0.69 \\ -0.61 \end{bmatrix}$

Then next approximate solution is given by $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{s}_0$, and process is repeated until convergence

$\boldsymbol{x}_k$		$\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2^2$
1.000	0.000	2.390
1.690	-0.610	0.212
1.975	-0.930	0.007
1.994	-1.004	0.002
1.995	-1.009	0.002
1.995	-1.010	0.002

Gauss-Newton Method, continued

Gauss-Newton method replaces nonlinear least squares problem by sequence of linear least squares problems whose solutions converge to solution of original nonlinear problem

If residual at solution is large, then second-order term omitted from Hessian is not negligible, and Gauss-Newton method may converge slowly or fail to converge

In such "large-residual" cases, it may be best to use general nonlinear minimization method that takes into account true full Hessian matrix

Levenberg-Marquardt Method

Levenberg-Marquardt method is another useful alternative when Gauss-Newton approximation is inadequate or yields rank deficient linear least squares subproblem

In this method, linear system at each iteration is of form

$$(\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{J}(\boldsymbol{x}_k) + \mu_k \boldsymbol{I})\boldsymbol{s}_k = -\boldsymbol{J}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k),$$

where μ_k is scalar parameter chosen by some strategy

Corresponding linear least squares problem is

$$\begin{bmatrix} \boldsymbol{J}(\boldsymbol{x}_k) \\ \sqrt{\mu_k}\boldsymbol{I} \end{bmatrix} \boldsymbol{s}_k \cong \begin{bmatrix} -\boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{o} \end{bmatrix}$$

With suitable strategy for choosing μ_k, this method can be very robust in practice, and it forms basis for several effective software packages

Equality-Constrained Optimization

For equality-constrained minimization problem

$$\min f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{o},$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{g}\colon \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, we seek critical point of Lagrangian

Applying Newton's method to nonlinear system

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} = \boldsymbol{o},$$

obtain linear system

$$\begin{bmatrix} \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) & \boldsymbol{J}_g^T(\boldsymbol{x}) \\ \boldsymbol{J}_g(\boldsymbol{x}) & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{\delta} \end{bmatrix} = - \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix}$$

for Newton step $(\boldsymbol{s}, \boldsymbol{\delta})$ in $(\boldsymbol{x}, \boldsymbol{\lambda})$ at each iteration

Sequential Quadratic Programming

Foregoing block 2×2 linear system equivalent to quadratic programming problem, so this approach known as *sequential quadratic programming*

Types of solution methods include:

- *Direct solution* methods, in which entire block 2×2 system is solved directly
- *Range space* methods, based on block elimination in block 2×2 linear system
- *Null space* methods, based on orthogonal factorization of matrix of constraint normals, $\boldsymbol{J}_g^T(\boldsymbol{x})$

Merit Function

Once Newton step $(\boldsymbol{s}, \boldsymbol{\delta})$ determined, need *merit function* to measure progress toward overall solution for use in line search or trust region

Popular choices include *penalty function*

$$\phi_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \tfrac{1}{2}\,\rho\,\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{g}(\boldsymbol{x})$$

and *augmented Lagrangian function*

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T\boldsymbol{g}(\boldsymbol{x}) + \tfrac{1}{2}\,\rho\,\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{g}(\boldsymbol{x}),$$

where parameter $\rho > 0$ determines weighting of optimality vs feasibility

Given starting guess $\boldsymbol{x}_0$, good starting guess for $\boldsymbol{\lambda}_0$ can be obtained from least squares problem

$$\boldsymbol{J}_g^T(\boldsymbol{x}_0)\,\boldsymbol{\lambda}_0 \cong -\nabla f(\boldsymbol{x}_0)$$

Inequality-Constrained Optimization

Methods just outlined for equality constraints can be extended to handle inequality constraints by using *active set* strategy

Inequality constraints are provisionally divided into those that are satisfied already (and can therefore be temporarily disregarded) and those that are violated (and are therefore temporarily treated as equality constraints)

This division of constraints is revised as iterations proceed until eventually correct constraints are identified that are binding at solution

Penalty Methods

Merit function can also be used to convert equality-constrained problem into sequence of unconstrained problems

If $\boldsymbol{x}_\rho^*$ is solution to

$$\min_{\boldsymbol{x}} \phi_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \tfrac{1}{2}\,\rho\,\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{g}(\boldsymbol{x}),$$

then under appropriate conditions

$$\lim_{\rho\to\infty} \boldsymbol{x}_\rho^* = \boldsymbol{x}^*$$

This enables use of unconstrained optimization methods, but problem becomes increasingly ill-conditioned for large ρ, so solve sequence of problems with increasing values of ρ

Barrier Methods

For inequality-constrained problems, another alternative is *barrier function*, such as

$$\phi_\mu(\boldsymbol{x}) = f(\boldsymbol{x}) - \mu \sum_{i=1}^{p} \frac{1}{h_i(\boldsymbol{x})}$$

or

$$\phi_\mu(\boldsymbol{x}) = f(\boldsymbol{x}) - \mu \sum_{i=1}^{p} \log(-h_i(\boldsymbol{x})),$$

which increasingly penalize feasible points as they approach boundary of feasible region

Again, solutions of unconstrained problem approach $\boldsymbol{x}^*$ as $\mu \to 0$, but problems increasingly ill-conditioned, so solve sequence of problems with decreasing values of μ

Barrier functions are basis for *interior point* methods for linear programming

Example: Constrained Optimization

Consider quadratic programming problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2,$$

subject to

$$g(\boldsymbol{x}) = x_1 - x_2 - 1 = 0,$$

with Lagrangian function given by

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda\, g(\boldsymbol{x})$$

$$= 0.5x_1^2 + 2.5x_2^2 + \lambda(x_1 - x_2 - 1)$$

Since

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{J}_g(\boldsymbol{x}) = [\,1 \quad -1\,],$$

we have

$$\nabla_x \mathcal{L}(\boldsymbol{x}, \lambda) = \nabla f(\boldsymbol{x}) + \boldsymbol{J}_g^T(\boldsymbol{x})\lambda = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} + \lambda \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Example Continued

So system to be solved for critical point of Lagrangian is

$$\begin{aligned} x_1 + \lambda &= 0, \\ 5x_2 - \lambda &= 0, \\ x_1 - x_2 &= 1, \end{aligned}$$

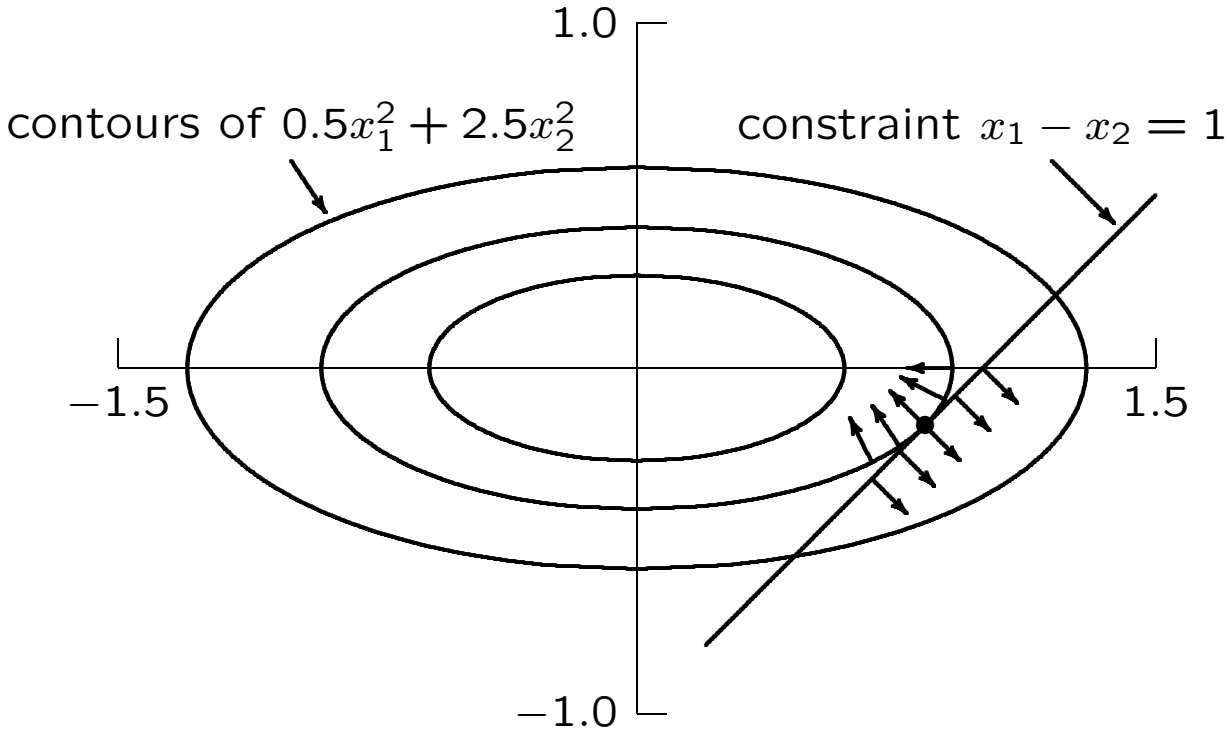
which in this case is linear system

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 5 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Solving this system, we obtain solution

$$x_1 = 0.833, \quad x_2 = -0.167, \quad \lambda = -0.833$$

Example Continued



Linear Programming

One of most important and common constrained optimization problems is *linear programming*

One standard form for such problems is

$$\min f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} \text{ subject to } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \text{ and } \boldsymbol{x} \geq \boldsymbol{o},$$

where $m < n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{c}, \boldsymbol{x} \in \mathbb{R}^n$

Feasible region is convex polyhedron in $\mathbb{R}^n$, and minimum must occur at one of its vertices

Simplex method moves from vertex to vertex until minimum point is found

Linear Programming, continued

Simplex method is reliable and normally efficient, able to solve problems with thousands of variables, but can require time exponential in size of problem in worst case

Interior point methods for linear programming developed in recent years have polynomial worst case solution time

These methods move through interior of feasible region, not restricting themselves to investigating only its vertices

Although interior point methods have significant practical impact, simplex method is still predominant method in standard packages for linear programming, and its effectiveness in practice is excellent

Example: Linear Programming

To illustrate linear programming, consider

$$\min_{\boldsymbol{x}} = \boldsymbol{c}^T\boldsymbol{x} = -8x_1 - 11x_2$$

subject to linear inequality constraints

$$5x_1+4x_2 \leq 40,\ -x_1+3x_2 \leq 12,\ x_1 \geq 0,\ x_2 \geq 0$$

Minimum value must occur at vertex of feasible region, in this case at $x_1 = 3.79$, $x_2 = 5.26$, where objective function has value -88.2