Outline

1. Optimization Problems
2. One-Dimensional Optimization
3. Multi-Dimensional Optimization
Optimization Problems
One-Dimensional Optimization
Multi-Dimensional Optimization

Optimization

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

- \( x^* \) is called *minimizer* or *minimum* of \( f \)

- It suffices to consider only minimization, since maximum of \( f \) is minimum of \(-f\)

- *Objective* function \( f \) is usually differentiable, and may be linear or nonlinear

- *Constraint* set \( S \) is defined by system of equations and inequalities, which may be linear or nonlinear

- Points \( x \in S \) are called *feasible* points

- If \( S = \mathbb{R}^n \), problem is *unconstrained*
Optimization Problems

- General continuous optimization problem:

\[
\min f(x) \quad \text{subject to} \quad g(x) = 0 \quad \text{and} \quad h(x) \leq 0
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \), \( g : \mathbb{R}^n \rightarrow \mathbb{R}^m \), \( h : \mathbb{R}^n \rightarrow \mathbb{R}^p \)

- Linear programming: \( f, g, \) and \( h \) are all linear

- Nonlinear programming: at least one of \( f, g, \) and \( h \) is nonlinear
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:

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

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

- $x^* \in S$ is **global minimum** if $f(x^*) \leq f(x)$ for all $x \in S$
- $x^* \in S$ is **local minimum** if $f(x^*) \leq f(x)$ for all feasible $x$ in some neighborhood of $x^*$
Global Optimization

- Finding, or even verifying, global minimum is difficult, in general
- Most optimization methods are designed to find local minimum, which may or may not be global minimum
- If global minimum is desired, one can try several widely separated starting points and see if all produce same result
- For some problems, such as linear programming, global optimization is more 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 \textit{coercive} if
  \[
  \lim_{\|x\| \to \infty} f(x) = +\infty
  \]
  i.e., \( f(x) \) must be large whenever \( \|x\| \) is large.

- If \( f \) is coercive on closed, unbounded set \( S \subseteq \mathbb{R}^n \), then \( f \) has global minimum on \( S \).
Level Sets

- **Level set** for function $f : S \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ is set of all points in $S$ for which $f$ has some given constant value.

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

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

- 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* of 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 $f : S \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ 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, one can 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

\[
\nabla f(x) = 0
\]

where \( \nabla f(x) \) is gradient vector of \( f \), whose \( i \)th component is \( \frac{\partial f(x)}{\partial x_i} \).

- For continuously differentiable \( f : S \subseteq \mathbb{R}^n \rightarrow \mathbb{R} \), any interior point \( x^* \) of \( S \) at which \( f \) has local minimum must be critical point of \( f \).

- But not all critical points are minima: they can also be maxima or saddle points.
Second-Order Optimality Condition

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

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

which is symmetric.

- At critical point $x^*$, if $H_f(x^*)$ is
  - positive definite, then $x^*$ is minimum of $f$
  - negative definite, then $x^*$ is maximum of $f$
  - indefinite, then $x^*$ is saddle point of $f$
  - singular, then various pathological situations are possible
Constrained Optimality

- If problem is constrained, only *feasible* directions are relevant.

- For equality-constrained problem

  \[
  \min f(x) \quad \text{subject to} \quad g(x) = 0
  \]

  where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( g : \mathbb{R}^n \rightarrow \mathbb{R}^m \), with \( m \leq n \), necessary condition for feasible point \( x^* \) to be solution is that negative gradient of \( f \) lie in space spanned by constraint normals,

  \[
  -\nabla f(x^*) = J_{g}^T(x^*)\lambda
  \]

  where \( J_{g} \) is Jacobian matrix of \( g \), and \( \lambda \) is vector of *Lagrange multipliers*.

- This condition says we cannot reduce objective function without violating constraints.
Constrained Optimality, continued

- **Lagrangian function** \( \mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R} \), is defined by
  \[
  \mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)
  \]

- Its gradient is given by
  \[
  \nabla \mathcal{L}(x, \lambda) = \begin{bmatrix}
  \nabla f(x) + J_g^T(x) \lambda \\
  g(x)
  \end{bmatrix}
  \]

- Its Hessian is given by
  \[
  H_{\mathcal{L}}(x, \lambda) = \begin{bmatrix}
  B(x, \lambda) & J_g^T(x) \\
  J_g(x) & O
  \end{bmatrix}
  \]

  where
  \[
  B(x, \lambda) = H_f(x) + \sum_{i=1}^{m} \lambda_i H_{g_i}(x)
  \]
Constrained Optimality, continued

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

\[ \nabla \mathcal{L}(\mathbf{x}, \lambda) = \begin{bmatrix} \nabla f(\mathbf{x}) + \mathbf{J}_g^T(\mathbf{x}) \lambda \\ g(\mathbf{x}) \end{bmatrix} = 0 \]

- 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 \((\mathbf{x}^*, \lambda^*)\) of \( \mathcal{L} \) is constrained minimum of \( f \) if \( B(\mathbf{x}^*, \lambda^*) \) is positive definite on null space of \( \mathbf{J}_g(\mathbf{x}^*) \).

- If columns of \( \mathbf{Z} \) form basis for null space, then test projected Hessian \( \mathbf{Z}^T \mathbf{B} \mathbf{Z} \) for positive definiteness.
If inequalities are present, then KKT optimality conditions also require nonnegativity of Lagrange multipliers corresponding to inequalities, and complementarity condition.
Sensitivity and Conditioning

- Function minimization and equation solving are closely related problems, but their sensitivities differ.

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

- For minimizing $f$, Taylor series expansion

\[
\begin{align*}
f(\hat{x}) &= f(x^* + h) \\
&= f(x^*) + f'(x^*)h + \frac{1}{2} f''(x^*)h^2 + O(h^3)
\end{align*}
\]

shows that, since $f'(x^*) = 0$, if $|f(\hat{x}) - f(x^*)| \leq \epsilon$, then $|\hat{x} - x^*|$ 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
For minimizing function of one variable, we 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 $x^* \leq x$.

Unimodality 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 \([a, b]\), with \( x_1 < x_2 \).

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

- To repeat process, we need 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, we choose relative positions of two points as $\tau$ 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 $\tau$ relative to previous interval, and interior point retained will be at position either $\tau$ or $1 - \tau$ relative to new interval

- To continue iteration, we need to compute only one new function value, at complementary point

- This choice of sample points is called *golden section search*

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

\[ \tau = \frac{(\sqrt{5} - 1)}{2} \]
\[ x_1 = a + (1 - \tau)(b - a); \quad f_1 = f(x_1) \]
\[ x_2 = a + \tau(b - a); \quad f_2 = f(x_2) \]

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
    \[ b = x_2 \]
    \[ x_2 = x_1 \]
    \[ f_2 = f_1 \]
    \[ x_1 = a + (1 - \tau)(b - a) \]
    \[ f_1 = f(x_1) \]
  end
end
Example: Golden Section Search

Use golden section search to minimize

\[ f(x) = 0.5 - x \exp(-x^2) \]
Example, continued

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$f_1$</th>
<th>$x_2$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.764</td>
<td>0.074</td>
<td>1.236</td>
<td>0.232</td>
</tr>
<tr>
<td>0.472</td>
<td>0.122</td>
<td>0.764</td>
<td>0.074</td>
</tr>
<tr>
<td>0.764</td>
<td>0.074</td>
<td>0.944</td>
<td>0.113</td>
</tr>
<tr>
<td>0.652</td>
<td>0.074</td>
<td>0.764</td>
<td>0.074</td>
</tr>
<tr>
<td>0.584</td>
<td>0.085</td>
<td>0.652</td>
<td>0.074</td>
</tr>
<tr>
<td>0.652</td>
<td>0.074</td>
<td>0.695</td>
<td>0.071</td>
</tr>
<tr>
<td>0.695</td>
<td>0.071</td>
<td>0.721</td>
<td>0.071</td>
</tr>
<tr>
<td>0.679</td>
<td>0.072</td>
<td>0.695</td>
<td>0.071</td>
</tr>
<tr>
<td>0.695</td>
<td>0.071</td>
<td>0.705</td>
<td>0.071</td>
</tr>
<tr>
<td>0.705</td>
<td>0.071</td>
<td>0.711</td>
<td>0.071</td>
</tr>
</tbody>
</table>

< interactive example >
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 is 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

\[ f(x) = 0.5 - x \exp(-x^2) \]
## Example, continued

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$f(x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.500</td>
</tr>
<tr>
<td>0.600</td>
<td>0.081</td>
</tr>
<tr>
<td>1.200</td>
<td>0.216</td>
</tr>
<tr>
<td>0.754</td>
<td>0.073</td>
</tr>
<tr>
<td>0.721</td>
<td>0.071</td>
</tr>
<tr>
<td>0.692</td>
<td>0.071</td>
</tr>
<tr>
<td>0.707</td>
<td>0.071</td>
</tr>
</tbody>
</table>

< interactive example >
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 $h = -\frac{f'(x)}{f''(x)}$

Suggests iteration scheme

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

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 to converge < interactive example >
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
  \[
  f''(x) = 2x(3 - 2x^2) \exp(-x^2)
  \]
- Newton iteration for zero of \( f' \) is given by
  \[
  x_{k+1} = x_k - \frac{(2x_k^2 - 1)}{(2x_k(3 - 2x_k^2))} \]
- Using starting guess \( x_0 = 1 \), we obtain
  \[
  \begin{array}{c|c}
  x_k & f(x_k) \\
  \hline
  1.000 & 0.132 \\
  0.500 & 0.111 \\
  0.700 & 0.071 \\
  0.707 & 0.071 \\
  \end{array}
  \]
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 are based on this hybrid approach.

Popular combination is golden section search and successive parabolic interpolation, for which no derivatives are 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, the Nelder-Mead method begins with \( n + 1 \) starting points, forming a simplex in \( \mathbb{R}^n \).

- Then move to a new point along a straight line from the current point having the highest function value through the centroid of the other points.

- New point replaces the worst point, and the process is repeated.

- Direct search methods are useful for nonsmooth functions or for small \( n \), but expensive for larger \( n \).
Steepest Descent Method

- Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a real-valued function of $n$ real variables.
- At any point $x$ where the gradient vector is nonzero, the negative gradient, $-\nabla f(x)$, points downhill toward lower values of $f$.
- In fact, $-\nabla f(x)$ is locally the direction of steepest descent: $f$ decreases more rapidly along the direction of the negative gradient than along any other.
- **Steepest descent** method: starting from initial guess $x_0$, successive approximate solutions are given by
  \[ x_{k+1} = x_k - \alpha_k \nabla f(x_k) \]
  where $\alpha_k$ is the line search parameter that determines how far to go in the given direction.
Steepest Descent, continued

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

$$\min_{\alpha_k} f(x_k - \alpha_k \nabla f(x_k))$$

that can be solved by methods already discussed

- Steepest descent method is very reliable: it 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
  \[ f(x) = 0.5x_1^2 + 2.5x_2^2 \]

- Gradient is given by \( \nabla f(x) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \)

- Taking \( x_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} \), we have \( \nabla f(x_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \)

- Performing line search along negative gradient direction,
  \[ \min_{\alpha_0} f(x_0 - \alpha_0 \nabla f(x_0)) \]
  exact minimum along line is given by \( \alpha_0 = 1/3 \), so next approximation is \( x_1 = \begin{bmatrix} 3.333 \\ -0.667 \end{bmatrix} \)
### Example, continued

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$f(x_k)$</th>
<th>$\nabla f(x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.000</td>
<td>1.000</td>
<td>15.000</td>
</tr>
<tr>
<td>3.333</td>
<td>−0.667</td>
<td>6.667</td>
</tr>
<tr>
<td>2.222</td>
<td>0.444</td>
<td>2.963</td>
</tr>
<tr>
<td>1.481</td>
<td>−0.296</td>
<td>1.317</td>
</tr>
<tr>
<td>0.988</td>
<td>0.198</td>
<td>0.585</td>
</tr>
<tr>
<td>0.658</td>
<td>−0.132</td>
<td>0.260</td>
</tr>
<tr>
<td>0.439</td>
<td>0.088</td>
<td>0.116</td>
</tr>
<tr>
<td>0.293</td>
<td>−0.059</td>
<td>0.051</td>
</tr>
<tr>
<td>0.195</td>
<td>0.039</td>
<td>0.023</td>
</tr>
<tr>
<td>0.130</td>
<td>−0.026</td>
<td>0.010</td>
</tr>
</tbody>
</table>
Example, continued

< interactive example >
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

\[ \mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_f^{-1}(\mathbf{x}_k) \nabla f(\mathbf{x}_k) \]

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

\[ \{ \mathbf{H}_f(\mathbf{x}) \}_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \]
Newton’s Method, continued

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

\[
H_f(x_k) s_k = -\nabla f(x_k)
\]

for Newton step \( s_k \), then take as next iterate

\[
x_{k+1} = x_k + s_k
\]

- Convergence rate of Newton’s method for minimization is normally quadratic

- As usual, Newton’s method is unreliable unless started close enough to solution to converge

< interactive example >
Example: Newton’s Method

- Use Newton’s method to minimize
  \[ f(x) = 0.5x_1^2 + 2.5x_2^2 \]

- Gradient and Hessian are given by
  \[ \nabla f(x) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \quad \text{and} \quad H_f(x) = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \]

- Taking \( x_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} \), we have \( \nabla f(x_0) = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \)

- Linear system for Newton step is
  \[ \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} s_0 = \begin{bmatrix} -5 \\ -5 \end{bmatrix} , \text{ so} \]
  \[ x_1 = x_0 + s_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} , \text{ which is exact solution} \]
  for this problem, as expected for quadratic function.
Newton’s Method, continued

- In principle, line search parameter is 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 $s_k$ to make method more robust (damped Newton).

- Once iterates are 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 $H_f$ 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, $H_f(x_k)$ may not be positive definite, so Newton step $s_k$ may not be descent direction for function, i.e., we may not have $\nabla f(x_k)^T s_k < 0$.

- In this case, alternative descent direction can be computed, such as negative gradient or direction of negative curvature, and then perform line search.
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:

  $$x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k)$$

  where $\alpha_k$ is line search parameter and $B_k$ is approximation to Hessian matrix.
- Many quasi-Newton methods are more robust than Newton’s method, are superlinearly convergent, and have lower overhead per iteration, which often more than offsets their slower convergence rate.
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

- BFGS Method is one of most effective secant updating methods for minimization
- Named after Broyden, Fletcher, Goldfarb, and Shanno
- Unlike Broyden’s method, BFGS preserves the symmetry of approximate Hessian matrix
- In addition, BFGS preserves the positive definiteness of the approximate Hessian matrix
Algorithm

\[ x_0 = \text{initial guess} \]
\[ B_0 = \text{initial Hessian approximation} \]
\textbf{for} \ k = 0, 1, 2, \ldots \textbf{do} \]
\hspace{1cm} \text{Solve} \ B_k s_k = -\nabla f(x_k) \text{ for } s_k \\
\hspace{1cm} x_{k+1} = x_k + s_k \\
\hspace{1cm} y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \\
\hspace{1cm} B_{k+1} = B_k + (y_k y_k^T) / (y_k^T s_k) - (B_k s_k s_k^T B_k) / (s_k^T B_k s_k) \]
Motivation of BFGS

- Let $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
- Matrix $B_{k+1}$ should satisfy secant equation

\[ B_{k+1} s_k = y_k \]

- In addition, $B_{k+1}$ is positive definition, which requires $s_k^T y_k > 0$
- There are infinite number of $B_{k+1}$ that satisfies secant equation
- Davidon (1950s) proposed to choose $B_{k+1}$ to be closest to $B_k$, i.e.,

\[
\min_B \|B - B_k\|
\]

Subject to $B = B^T, Bs_k = y_k$.

- BFGS proposed to choose $B_{k+1}$ so that $B_{k+1}^{-1}$ is closest to $B_k^{-1}$, i.e.,

\[
\min_B \|B^{-1} - B_k^{-1}\|
\]

Subject to $B = B^T, Bs_k = y_k$. 
Properties of BFGS

- BFGS normally has superlinear convergence rate, even though approximate Hessian does not necessarily converge to true Hessian.
- Approximate Hessian preserves positive definiteness.
  - Key idea of proof: Let $H_k$ denote $B_k^{-1}$. For any vector $z \neq 0$, and let $w = z - \rho_k y_k (s_k^T z)$, where $\rho_k > 0$. Then it can be shown that
    $$z^T H_{k+1} z = w^T H_k w + \rho_k (s_k^T z)^2 \geq 0.$$
    If $s_k^T z = 0$, then $w = z \neq 0$. So $z^T H_{k+1} z > 0$.
- Line search can be used to enhance effectiveness of BFGS. If exact line search is performed at each iteration, BFGS terminates at exact solution in at most $n$ iterations for a quadratic objective function.
BFGS Method, continued

- In practice, factorization of $B_k$ is updated rather than $B_k$ itself, so linear system for $s_k$ can be solved at cost of $O(n^2)$ rather than $O(n^3)$ work.
- Unlike Newton’s method for minimization, no second derivatives are required.
- Can start with $B_0 = I$, 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.
Example: BFGS Method

- Use BFGS to minimize \( f(x) = 0.5x_1^2 + 2.5x_2^2 \)
- Gradient is given by \( \nabla f(x) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \)
- Taking \( x_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}^T \) and \( B_0 = I \), initial step is negative gradient, so

\[
x_1 = x_0 + s_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} -5 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}
\]

- Updating approximate Hessian using BFGS formula, we obtain

\[
B_1 = \begin{bmatrix} 0.667 & 0.333 \\ 0.333 & 0.667 \end{bmatrix}
\]

- Then new step is computed and process is repeated
### Example: BFGS Method

<table>
<thead>
<tr>
<th>( \mathbf{x}_k )</th>
<th>( f(\mathbf{x}_k) )</th>
<th>( \nabla f(\mathbf{x}_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.000</td>
<td>15.000</td>
<td>5.000, 5.000</td>
</tr>
<tr>
<td>0.000</td>
<td>40.000</td>
<td>0.000, -20.000</td>
</tr>
<tr>
<td>-2.222</td>
<td>2.963</td>
<td>-2.222, 2.222</td>
</tr>
<tr>
<td>0.816</td>
<td>0.350</td>
<td>0.816, 0.408</td>
</tr>
<tr>
<td>-0.009</td>
<td>0.001</td>
<td>-0.009, -0.077</td>
</tr>
<tr>
<td>-0.001</td>
<td>0.000</td>
<td>-0.001, 0.005</td>
</tr>
</tbody>
</table>

- 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.

< interactive example >
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.
Motivation of Conjugate Gradients

- Conjugate gradient can be used to solve a linear system $Ax = b$, where $A$ is symmetric positive definite (SPD).
- If $A$ is $m \times m$ SPD, then quadratic function $\varphi(x) = \frac{1}{2}x^T Ax - x^T b$ has unique minimum.
- Negative gradient of this function is residual vector $-\nabla \varphi(x) = b - Ax = r$ so minimum is obtained precisely when $Ax = b$. 
Optimization methods have form

$$x_{n+1} = x_n + \alpha_n p_n$$

where \( p_n \) is search direction and \( \alpha \) is step length chosen to minimize \( \varphi(x_n + \alpha_n p_n) \).

Line search parameter can be determined analytically as

$$\alpha_n = \frac{r_n^T p_n}{p_n^T A p_n}$$

In CG, \( p_n \) is chosen to be A-conjugate (or A-orthogonal) to previous search directions, i.e., \( p_n^T A p_j = 0 \) for \( j < n \).
Optimality of Step Length

- Select step length $\alpha_n$ over vector $p_{n-1}$ to minimize
  $\varphi(x) = \frac{1}{2} x^T A x - x^T b$
- Let $x_n = x_{n-1} + \alpha_n p_{n-1}$,

  $\varphi(x_n) = \frac{1}{2} (x_{n-1} + \alpha_n p_{n-1})^T A (x_{n-1} + \alpha_n p_{n-1}) - (x_{n-1} + \alpha_n p_{n-1})^T b$

  $= \frac{1}{2} \alpha_n^2 p_{n-1}^T A p_{n-1} + \alpha_n p_{n-1}^T A x_{n-1} - \alpha_n p_{n-1}^T b + \text{constant}$

  $= \frac{1}{2} \alpha_n^2 p_{n-1}^T A p_{n-1} - \alpha_n p_{n-1}^T r_{n-1} + \text{constant}$

- Therefore,

  \[ \frac{d\varphi}{d\alpha_n} = 0 \Rightarrow \alpha_n p_{n-1}^T A p_{n-1} - p_{n-1}^T r_{n-1} = 0 \Rightarrow \alpha_n = \frac{p_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}. \]

- In addition, $p_{n-1}^T r_{n-1} = r_{n-1}^T r_{n-1}$ because $p_{n-1} = r_{n-1} + \beta_n p_{n-2}$ and $r_{n-1}^T p_{n-2} = 0$. 
Algorithm: Conjugate Gradient Method

\[ x_0 = 0, \quad r_0 = b, \quad p_0 = r_0 \]

for \( n = 1 \) to \( 1, 2, 3, \ldots \)

\[ \alpha_n = (r_{n-1}^T r_{n-1})/(p_{n-1}^T Ap_{n-1}) \]

step length

\[ x_n = x_{n-1} + \alpha_n p_{n-1} \]

approximate solution

\[ r_n = r_{n-1} - \alpha_n Ap_{n-1} \]

residual

\[ \beta_n = (r_n^T r_n)/(r_{n-1}^T r_{n-1}) \]

improvement this step

\[ p_n = r_n + \beta_n p_{n-1} \]

search direction

- Only one matrix-vector product \( Ap_{n-1} \) per iteration
- Apart from matrix-vector product, \#operations per iteration is \( O(m) \)
- CG can be viewed as minimization of quadratic function \( \varphi(x) = \frac{1}{2} x^T Ax - x^T b \) by modifying steepest descent
- First proposed by Hestens and Stiefel in 1950s
An Alternative Interpretation of CG

**Algorithm: CG**

\[
\begin{align*}
x_0 &= 0, \quad r_0 = b, \quad p_0 = r_0 \\
\text{for } n &= 1, 2, 3, \ldots \\
\alpha_n &= r_{n-1}^T r_{n-1} / (p_{n-1}^T A p_{n-1}) \\
x_n &= x_{n-1} + \alpha_n p_{n-1} \\
r_n &= r_{n-1} - \alpha_n A p_{n-1} \\
\beta_n &= r_n^T r_n / (r_{n-1}^T r_{n-1}) \\
p_n &= r_n + \beta_n p_{n-1}
\end{align*}
\]

**Algorithm: A non-standard CG**

\[
\begin{align*}
x_0 &= 0, \quad r_0 = b, \quad p_0 = r_0 \\
\text{for } n &= 1, 2, 3, \ldots \\
\alpha_n &= r_{n-1}^T p_{n-1} / (p_{n-1}^T A p_{n-1}) \\
x_n &= x_{n-1} + \alpha_n p_{n-1} \\
r_n &= b - A x_n \\
\beta_n &= -r_n^T A p_{n-1} / (p_{n-1}^T A p_{n-1}) \\
p_n &= r_n + \beta_n p_{n-1}
\end{align*}
\]

- The non-standard one is less efficient but easier to understand
- It is easy to see \( r_n = r_{n-1} - \alpha_n A p_{n-1} = b - A x_n \)
Comparison of Linear and Nonlinear CG

Algorithm: Linear CG

\[ x_0 = 0, \quad r_0 = b, \]
\[ p_0 = r_0 \]
\[
\text{for } n = 1, 2, 3, \ldots
\]
\[ \alpha_n = r_{n-1}^T r_{n-1} / (p_{n-1}^T A p_{n-1}) \]
\[ x_n = x_{n-1} + \alpha_n p_{n-1} \]
\[ r_n = r_{n-1} - \alpha_n A p_{n-1} \]
\[ \beta_n = r_n^T r_n / (r_{n-1}^T r_{n-1}) \]
\[ p_n = r_n + \beta_n p_{n-1} \]

Algorithm: Non-linear CG

\[ x_0 = \text{initial guess}, \quad g_0 = \nabla f(x_0), \]
\[ s_0 = -g_0 \]
\[
\text{for } k = 0, 1, 2, \ldots
\]
Choose \( \alpha_k \) to min \( f(x_k + \alpha_k s_k) \)
\[ x_{k+1} = x_k + \alpha_k s_k \]
\[ g_{k+1} = \nabla f(x_{k+1}) \]
\[ \beta_{k+1} = (g_{k+1}^T g_{k+1}) / (g_k^T g_k) \]
\[ s_{k+1} = -g_{k+1} + \beta_{k+1} s_k \]

- \( \beta_{k+1} = (g_{k+1}^T g_{k+1}) / (g_k^T g_k) \) was due to Fletcher and Reeves (1964)
- An alternative formula \( \beta_{k+1} = (g_{k+1} - g_k)^T g_{k+1} / (g_k^T g_k) \) was due to Polak and Ribeire (1969)
Properties of Conjugate Gradients

Krylov subspaces for $Ax = b$ is $\mathcal{K}_n = \{b, Ab, \ldots, A^{n-1}b\}$.

**Theorem**

If $r_{n-1} \neq 0$, spaces spanned by approximate solutions $x_n$, search directions $p_n$, and residuals $r_n$ are all equal to Krylov subspaces

$$\mathcal{K}_n = \langle x_1, x_2, \ldots, x_n \rangle = \langle p_0, p_1, \ldots, p_{n-1} \rangle = \langle r_0, r_1, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle$$

The residual are orthogonal (i.e., $r_n^T r_j = 0$ for $j < n$) and search directions are $A$-conjugate (i.e, $p_n^T A p_j = 0$ for $j < n$).

**Theorem**

If $r_{n-1} \neq 0$, then error $e_n = x_* - x_n$ are minimized in $A$-norm in $\mathcal{K}_n$.

- Because $\mathcal{K}_n$ grows monotonically, error decreases monotonically.
Some important convergence results for solving linear systems

- If $A$ has $n$ distinct eigenvalues, CG converges in at most $n$ steps.
- If $A$ has 2-norm condition number $\kappa$, the errors are

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n$$

which is $\approx 2 \left( 1 - \frac{2}{\sqrt{\kappa}} \right)^n$ as $\kappa \to \infty$. So convergence is expected in $O(\sqrt{\kappa})$ iterations.

- In general, CG performs well with clustered eigenvalues.
Example: Conjugate Gradient Method

- Use CG method to minimize \( f(\mathbf{x}) = 0.5x_1^2 + 2.5x_2^2 \)
- Gradient is given by \( \nabla f(\mathbf{x}) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \)
- Taking \( \mathbf{x}_0 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}^T \), initial search direction is negative gradient,
  \[
  \mathbf{s}_0 = -\mathbf{g}_0 = -\nabla f(\mathbf{x}_0) = \begin{bmatrix} -5 \\ -5 \end{bmatrix}
  \]
- Exact minimum along line is given by \( \alpha_0 = 1/3 \), so next approximation is \( \mathbf{x}_1 = \begin{bmatrix} 3.333 \\ -0.667 \end{bmatrix}^T \), and we compute new gradient,
  \[
  \mathbf{g}_1 = \nabla f(\mathbf{x}_1) = \begin{bmatrix} 3.333 \\ -3.333 \end{bmatrix}
  \]
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:

\[ \beta_1 = \frac{g_1^T g_1}{g_0^T g_0} = 0.444 \]

which gives as next search direction:

\[ s_1 = -g_1 + \beta_1 s_0 = \begin{bmatrix} -3.333 \\ 3.333 \end{bmatrix} + 0.444 \begin{bmatrix} -5 \\ -5 \end{bmatrix} = \begin{bmatrix} -5.556 \\ 1.111 \end{bmatrix} \]

- Minimum along this direction is given by \( \alpha_1 = 0.6 \), which gives exact solution at origin, as expected for quadratic function.
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 \(x\) of parameters that gives “best fit” in least squares sense to model function \(f(t, x)\), where \(f\) is nonlinear function of \(x\).

- Define components of residual function

\[
  r_i(x) = y_i - f(t_i, x), \quad i = 1, \ldots, m
\]

so we want to minimize \(\phi(x) = \frac{1}{2} r^T(x) r(x)\).

- Gradient vector is \(\nabla \phi(x) = J^T(x) r(x)\) and Hessian matrix is

\[
  H_{\phi}(x) = J^T(x) J(x) + \sum_{i=1}^{m} r_i(x) H_i(x)
\]

where \(J(x)\) is Jacobian of \(r(x)\), and \(H_i(x)\) is Hessian of \(r_i(x)\).
Nonlinear Least Squares, continued

- Linear system for Newton step is

\[
\left( J^T(x_k)J(x_k) + \sum_{i=1}^{m} r_i(x_k)H_i(x_k) \right) s_k = -J^T(x_k)r(x_k)
\]

- \( m \) Hessian matrices \( H_i \) are usually inconvenient and expensive to compute

- Moreover, in \( H_\phi \) each \( H_i \) 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

\[ J^T(x_k)J(x_k)s_k = -J^T(x_k)r(x_k) \]

is solved for approximate Newton step \( s_k \) at each iteration.

- This is system of normal equations for linear least squares problem

\[ J(x_k)s_k \approx -r(x_k) \]

which can be solved better by QR factorization.

- Next approximate solution is then given by

\[ x_{k+1} = x_k + s_k \]

and process is repeated until convergence.
Example: Gauss-Newton Method

- Use Gauss-Newton method to fit nonlinear model function

\[ f(t, x) = x_1 \exp(x_2t) \]

- For this model function, entries of Jacobian matrix of residual function \( r \) are given by

\[ \{ J(x) \}_{i,1} = \frac{\partial r_i(x)}{\partial x_1} = -\exp(x_2t_i) \]

\[ \{ J(x) \}_{i,2} = \frac{\partial r_i(x)}{\partial x_2} = -x_1t_i \exp(x_2t_i) \]
Example, continued

If we take $x_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, then Gauss-Newton step $s_0$ is given by linear least squares problem

$$
\begin{bmatrix}
-1 & 0 \\
-1 & -1 \\
-1 & -2 \\
-1 & -3 \\
\end{bmatrix}
\begin{bmatrix}
s_0 \\
\end{bmatrix} \approx
\begin{bmatrix}
-1 \\
0.3 \\
0.7 \\
0.9 \\
\end{bmatrix}
$$

whose solution is $s_0 = \begin{bmatrix} 0.69 \\ -0.61 \end{bmatrix}$

Then next approximate solution is given by $x_1 = x_0 + s_0$, and process is repeated until convergence
Example, continued

\[
\begin{array}{ccc}
\mathbf{x}_k & \| \mathbf{r}(\mathbf{x}_k) \|_2^2 \\
1.000 & 0.000 & 2.390 \\
1.690 & -0.610 & 0.212 \\
1.975 & -0.930 & 0.007 \\
1.994 & -1.004 & 0.002 \\
1.995 & -1.009 & 0.002 \\
1.995 & -1.010 & 0.002 \\
\end{array}
\]

< interactive example >
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

\[
(J^T(x_k)J(x_k) + \mu_k I)s_k = -J^T(x_k)r(x_k)
\]

where \(\mu_k\) is scalar parameter chosen by some strategy.

- Corresponding linear least squares problem is

\[
\begin{bmatrix}
J(x_k) \\
\sqrt{\mu_k}I
\end{bmatrix}
\begin{bmatrix}
s_k
\end{bmatrix}
\approx
\begin{bmatrix}
-r(x_k) \\
0
\end{bmatrix}
\]

- With suitable strategy for choosing \(\mu_k\), this method can be very robust in practice, and it forms basis for several effective software packages. < interactive example >
Equality-Constrained Minimization

- Equality-constrained problem has form

\[
\min_{x \in \mathbb{R}^n} f(x) \text{ subject to } g(x) = 0
\]

where objective function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and constraints \( g : \mathbb{R}^n \rightarrow \mathbb{R}^m \), where \( m \leq n \)

- Necessary condition for feasible point \( x \) to be solution is that negative gradient of \( f \) lie in space spanned by constraint normals, i.e.,

\[
-\nabla f(x^*) = J_g^T(x^*)\lambda,
\]

where \( J_g \) is Jacobian matrix of \( g \), and \( \lambda \) is vector of Lagrange multipliers

- Therefore, constrained local minimum must be critical point of Lagrangian function

\[
\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)
\]
First-Order and Second-Order Optimality Conditions

- Equality-constrained minimization can be reduced to solving
  \[
  \nabla \mathcal{L}(x, \lambda) = \begin{bmatrix}
  \nabla f(x) + J_g^T(x) \lambda \\
  g(x)
\end{bmatrix} = 0,
  \]
  which is known as \textit{Karush-Kuhn-Tucker} (or \textit{KKT}) condition for constrained local minimum.
- Hessian of Lagrangian function
  \[
  H_{\mathcal{L}}(x, \lambda) = \begin{bmatrix}
  B(x, \lambda) & J_g^T(x) \\
  J_g(x) & 0
\end{bmatrix}
  \]
  where \(B(x, \lambda) = H_f(x) + \sum_{i=1}^{m} \lambda_i H_{g_i}(x)\). \(H_{\mathcal{L}}\) is sometimes called KKT (Karush-Kuhn-Tucker) matrix. \(H_{\mathcal{L}}\) is symmetric, but not in general positive definite.
- Critical point \((x^*, \lambda^*)\) of \(\mathcal{L}\) is constrained minimum if \(B(x^*, \lambda^*)\) is positive definite on null space of \(J_g(x^*)\).
- Let \(Z\) form basis of null \((J_g(x^*))\), then projected Hessian \(Z^T B Z\) should be positive definite.
Sequential Quadratic Programming

- $\nabla L(x, \lambda) = 0$ can be solved using Newton's method. $k$th iteration of Newton’s step is

$$
\begin{bmatrix}
B(x_k, \lambda_k) & J_g^T(x_k) \\
J_g(x_k) & 0
\end{bmatrix}
\begin{bmatrix}
s_k \\
\delta_k
\end{bmatrix}
= -
\begin{bmatrix}
\nabla f(x_k) + J_g^T(x_k)\lambda_k \\
g(x_k)
\end{bmatrix},
$$

and then $x_{k+1} = x_k + s_k$ and $\lambda_{k+1} = \lambda_k + \delta_k$

- Above system of equations is first-order optimality condition for constrained optimization problem

$$
\min_s \frac{1}{2} s^T B(x_k, \lambda_k) s + s^T \left( \nabla f(x_k) + J_g^T(x_k)\lambda_k \right)
$$

subject to

$$
J_g(x_k)s + g(x_k) = 0.
$$

- This problem is quadratic programming problem, so approach using Newton’s method is known as sequential quadratic programming
Sequential Quadratic Programming

- Foregoing block $2 \times 2$ linear system is equivalent to quadratic programming problem, so this approach is known as *sequential quadratic programming*.

- Types of solution methods include:
  - *Direct solution* methods, in which entire block $2 \times 2$ system is solved directly.
  - *Range space* methods, based on block elimination in block $2 \times 2$ linear system.
  - *Null space* methods, based on orthogonal factorization of matrix of constraint normals, $J_g^T(x)$.

< interactive example >
Solving KKT System

- KKT system \[
\begin{bmatrix}
B & J^T \\
J & 0
\end{bmatrix}
\begin{bmatrix}
s \\
\delta
\end{bmatrix}
= \begin{bmatrix}
w \\
g
\end{bmatrix}
\] can be solved in several ways

- Direct solution
  - Solve system using method for symmetric indefinite factorization, such as \(LDL^T\) with pivoting, or
  - Use iterative method such as GMRES, MINRES

- Range-space method
  - Use block elimination and obtain symmetric system
    \[
    \left(JB^{-1}J^T\right)\delta = g - JB^{-1}w
    \]
    and then
    \[
    Bs = -w - J^T\delta
    \]
  - First equation finds \(\delta\) by projecting \(B^{-1}\) onto range space of \(J^T\)
  - It is attractive when number of constraints \(m\) is relatively small, because \(JB^{-1}J^T\) is \(m \times m\)
  - However, it requires \(B\) to be nonsingular and \(J\) has full rank. Also, condition number of \(JB^{-1}J^T\) may be large
Solving KKT System Cont’d

Null space method

- Compute QR factorization $J^T = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$
- Partition $Q = \begin{bmatrix} Y & Z \end{bmatrix}$, where $JY = R^T$ and $JZ = 0$
- Write $s = Yu + Zv$. Second block row yields
  \[ Js = J(Yu + Zv) = R^T u = -g \]
  and premultiplying first block row by $Z^T$ yields
  \[ \left(Z^T B Z \right)v = -Z^T (w + BYu) \]
- Finally,
  \[ Y^T J^T \delta = R\delta = -Y^T (w + Bs) \]
- This method is advantageous when $n - m$ is small
- It is more stable than range-space method. Also, $B$ does not need to be nonsingular
Once Newton step \((s, \delta)\) determined, we need *merit function* to measure progress toward overall solution for use in line search or trust region.

Popular choices include *penalty function*

\[
\phi_\rho(x) = f(x) + \frac{1}{2} \rho g(x)^T g(x)
\]

and *augmented Lagrangian function*

\[
\mathcal{L}_\rho(x, \lambda) = f(x) + \lambda^T g(x) + \frac{1}{2} \rho g(x)^T g(x)
\]

where parameter \(\rho > 0\) determines relative weighting of optimality vs feasibility.

Given starting guess \(x_0\), good starting guess for \(\lambda_0\) can be obtained from least squares problem

\[
J_g^T(x_0) \lambda_0 \approx -\nabla f(x_0)
\]
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 $x^*_\rho$ is solution to

$$\min_x \phi_\rho(x) = f(x) + \frac{1}{2} \rho g(x)^T g(x)$$

then under appropriate conditions

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

- This enables use of unconstrained optimization methods, but problem becomes ill-conditioned for large $\rho$, so we solve sequence of problems with gradually increasing values of $\rho$, with minimum for each problem used as starting point for next problem
Barrier Methods

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

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

or

\[ \phi_\mu(x) = f(x) - \mu \sum_{i=1}^{p} \log(-h_i(x)) \]

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

Again, solutions of unconstrained problem approach \( x^* \) as \( \mu \to 0 \), but problems are increasingly ill-conditioned, so solve sequence of problems with decreasing values of \( \mu \)

Barrier functions are basis for *interior point* methods for linear programming
Example: Constrained Optimization

Consider quadratic programming problem

$$\min_x f(x) = 0.5x_1^2 + 2.5x_2^2$$

subject to

$$g(x) = x_1 - x_2 - 1 = 0$$

Lagrangian function is given by

$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x) = 0.5x_1^2 + 2.5x_2^2 + \lambda(x_1 - x_2 - 1)$$

Since

$$\nabla f(x) = \begin{bmatrix} x_1 \\ 5x_2 \end{bmatrix} \quad \text{and} \quad J_g(x) = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

we have

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

- So system to be solved for critical point of Lagrangian is

\[
\begin{align*}
 x_1 + \lambda &= 0 \\
 5x_2 - \lambda &= 0 \\
 x_1 - x_2 &= 1
\end{align*}
\]

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

contours of $0.5x_1^2 + 2.5x_2^2$

line $x_1 - x_2 = 1$
Linear Programming

- Linear programming has linear objective function and linear equality and inequality constraints
- Example: Maximize profit of combination of wheat and barley, but with limited budget of land, fertilizer, and insecticide. Let $x_1$ and $x_2$ be areas planted for wheat and barley, we have linear programming problem

$$\text{maximize } c_1 x_1 + c_2 x_2 \quad \{\text{maximize revenue}\}$$
$$0 \leq x_1 + x_2 \leq L \quad \{\text{limit on area}\}$$
$$F_1 x_1 + F_2 x_2 \leq F \quad \{\text{limit on fertilizer}\}$$
$$P_1 x_1 + P_2 x_2 \leq P \quad \{\text{limit on insecticide}\}$$
$$x_1 \geq 0, \ x_2 \geq 0 \quad \{\text{nonnegative land}\}$$

- Linear programming is typically solved by *simplex methods* or *interior point methods*
Standard Form of Linear Programming

- Linear programming has many forms. A standard form (called *slack form*) is
  \[
  \min \ c^T x \text{ subject to } Ax = b \text{ and } x \geq 0
  \]

- Simplex method and interior-point method requires slack form

- Previous example can be converted into standard form

  minimize \((-c_1)x_1 + (-c_2)x_2\) \{maximize revenue\}

  \[
  x_1 + x_2 + x_3 = L \text{ \{limit on area\}}
  \]

  \[
  F_1 x_1 + F_2 x_2 + x_4 = F \text{ \{limit on fertilizer\}}
  \]

  \[
  P_1 x_1 + P_2 x_2 + x_5 = P \text{ \{limit on insecticide\}}
  \]

  \[
  x_1, x_2, x_3, x_4, x_5 \geq 0 \text{ \{nonnegativity\}}
  \]

  Here, \(x_3, x_4, \text{ and } x_5\) are called *slack variables*
Duality

- $m$ equations $Ax = b$ have $m$ corresponding Lagrange multipliers in $y$
- Primal problem

  \[
  \text{Minimize } c^T x \text{ subject to } Ax = b \text{ and } x \geq 0
  \]

- Dual problem

  \[
  \text{Maximize } b^T y \text{ subject to } A^T y \leq c
  \]

- Weak duality: $b^T y \leq c^T x$ for any feasible $x$ and $y$
  - because $b^T y = (Ax)^T y = x^T (A^T y) \leq x^T c = c^T x$

- Strong duality: If both feasible sets of primal and dual problems are nonempty, then $c^T x^* = b^T y^*$ at optimal $x^*$ and $y^*$
Simplex Methods

- Developed by George Dantzig in 1947
- Key observation: Feasible region is convex polytope in $\mathbb{R}^n$, and minimum must occur at one of its vertices
- Basic idea: Construct a feasible solution at a vertex of the polytope, walk along a path on the edges of the polytope to vertices with non-decreasing values of the objective function, until an optimum is reached
- Simplex method in the worst case can be slow, because number of corners is exponential with $m$ and $n$
- However, its average-case complexity is polynomial time, and in practice, best corner is often found in $2m$ steps
Example: Linear Programming

To illustrate linear programming, consider

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

subject to linear inequality constraints

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

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 $-88.2$
Example, continued

\[ 5x_1 + 4x_2 = 40 \]

\[ -x_1 + 3x_2 = 12 \]
Interior Point Methods

- First proposed by Narendra Karmarkar in 1984
- In contrast to simplex methods, interior point methods move through the interior of the feasible region
- Barrier problem

\[
\text{minimize } c^T x - \theta (\log x_1 + \cdots + \log x_n) \quad \text{with } Ax = b
\]

When any \(x_i\) touches zero, extra cost \(-\theta \log x_i\) blows up

- Barrier problem gives approximate problem for each \(\theta\). Its Lagrangian is

\[
\mathcal{L}(x, y, \theta) = c^T x - \theta \left( \sum \log x_i \right) - y^T (Ax - b)
\]

- The derivatives \(\partial \mathcal{L} / \partial x_j = c_j - \frac{\theta}{x_j} - (A^T y)_j = 0\), or \(x_j s_j = \theta\), where \(s = c - A^T y\)
Newton Step

- $n$ optimality equations $x_j s_j = \theta$ are nonlinear, and are solved iteratively using Newton’s method.
- To determine increment $\Delta x$, $\Delta y$, and $\Delta s$, we need to solve $(x_i + \Delta x_i)(s_i + \Delta s_i) = \theta$. It is typical to ignore second order term $\Delta x_i \Delta s_i$. Then linear equations become

\[
A \Delta x = 0
\]
\[
A^T \Delta y + \Delta s = 0
\]
\[
s_j \Delta x_j + x_j \Delta s_j = \theta - x_j s_j.
\]
- The iteration has quadratic convergence for each $\theta$, and $\theta$ approaches zero.
Example

Minimize $c^T x = 5x_1 + 3x_2 + 8x_3$ with $x_i \geq 0$ and $Ax = x_1 + x_2 + 2x_3 = 4$.

- Barrier Lagrangian is
  $$L = (5x_1 + 3x_2 + 8x_3) + \theta(\log x_1 + \log x_2 + \log x_3) - y(x_1 + x_2 + 2x_3 - 4)$$

- Optimality equation gives us:
  $$(s = c - ATy) \quad s_1 = 5 - y, \quad s_2 = 3 - y, \quad s_3 = 8 - 2y$$
  $$\frac{\partial L}{\partial x_i} = 0 \quad x_1s_1 = x_2s_2 = x_3s_3 = \theta$$
  $$\frac{\partial L}{\partial y} = 0 \quad x_1 + x_2 + 2x_3 = 4$$

- Start from an interior point $x_1 = x_2 = x_3 = 1$, $y = 1$, and $s = (3, 1, 4)$. From $A\Delta x = 0$ and $x_j s_j + s_j \Delta x_j + x_j \Delta s_j = \theta$, we obtain equations
  $$3\Delta x_1 - 1\Delta y = \theta - 3$$
  $$1\Delta x_2 - 1\Delta y = \theta - 1$$
  $$4\Delta x_3 - 2\Delta y = \theta - 4$$
  $$\Delta x_1 + \Delta x_2 + 2\Delta x_3 = 0.$$ 

- Given $\theta = 4/3$, we then obtain $x_{new} = (2/3, 2, 2/3)$ and $y_{new} = 8/3$, whereas $x^* = (0, 4, 0)$ and $y^* = 3$. 

Xiangmin Jiao (SUNY Stony Brook)  AMS527: Numerical Analysis II