Minimization of Static Cost Functions
Robert Stengel
Optimal Control and Estimation, MAE 546, Princeton University, 2018

- $J$ = Static cost function with constant control parameter vector, $u$
- Conditions for a minimum in $J$ with respect to $u$
- Analytical and numerical solutions

Copyright 2018 by Robert Stengel. All rights reserved. For educational use only.
http://www.princeton.edu/~stengel/MAE546.html
http://www.princeton.edu/~stengel/OptConEst.html

Static Cost Function

- Minimum value of the cost, $J$, is fixed, but may be unknown
- Corresponding control parameter, $u^*$, also is fixed but possibly unknown
- Cost function preferably has
  - Single minimum
  - Locally smooth, monotonic contours away from the minimum
Vector Norms for Real Variables

- “Norm” = Measure of length or magnitude of a vector, $\mathbf{x}$
  - Scalar quantity
- Taxicab or Manhattan norm
  \[
  L^1 \text{ norm} = \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|
  \]
- Euclidean or Quadratic Norm
  \[
  L^2 \text{ norm} = \|\mathbf{x}\|_2 = (\mathbf{x}^T \mathbf{x})^{1/2} = \left( x_1^2 + x_2^2 + \cdots + x_n^2 \right)^{1/2}
  \]
- $p$ Norm
  \[
  L^p \text{ norm} = \|\mathbf{x}\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}
  \]
- Infinity Norm:
  \[
  L^\infty \text{ norm} = \|\mathbf{x}\|_\infty = \max_{i} x_i
  \]

$p$ Norm ($L^p$ Norm) Example

\[
L^p \text{ norm} \triangleq \|\mathbf{x}\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}
\]

15-dimensional vector

As $p$ increases, the norm approaches the value of the maximum component of the vector.
"Apples and Oranges" Problem

Suppose elements of \( x \) represent quantities with different units

\[
x = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} = \begin{bmatrix}
Velocity, m/s \\
Angle, rad \\
\vdots \\
Temperature, ^\circ K
\end{bmatrix}
\]

What is a reasonable definition for the norm?

One solution: Vector, \( y \), normalized by ranges of elements of \( x \)

\[
y = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} \approx \begin{bmatrix}
d_1 x_1 \\
d_2 x_2 \\
\vdots \\
d_m x_n
\end{bmatrix} = \begin{bmatrix}
x_1/\left(x_{\text{max}} - x_{\text{min}}\right) \\
x_2/\left(x_{\text{max}} - x_{\text{min}}\right) \\
\vdots \\
x_m/\left(x_{\text{max}} - x_{\text{min}}\right)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
d_1 & 0 & \cdots & 0 \\
0 & d_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & d_m
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} = Dx
\]

Weighted Euclidean (Quadratic)
Norm of \( x \)

\[
\|y\|_2 = \left(y^T y\right)^{1/2} = \left(y_1^2 + y_2^2 + \cdots + y_m^2\right)^{1/2}
\]

\[
= \left(x^T D^T Dx\right)^{1/2} = \|Dx\|_2
\]

- If \( m = n \),
  - \( D \) is square
  - \( D^T D \) is square and symmetric
- If \( D \) is diagonal
  - \( D^T D \) is diagonal

- If \( m \neq n \),
  - \( D \) is not square
  - \( D^T D \) is square and symmetric
- \( \text{Rank}(D) \leq m, n > m \)
- \( \text{Rank}(D) \leq n, n < m \)
**Rank of Matrix $D$**

- Maximal number of linearly independent rows or columns of $D$
- Order of the largest non-zero minor determinant of $D$
- Examples

$$D = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}; \text{ maximal rank } \leq 2$$

$$D = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}; \text{ maximal rank } \leq 2$$

---

**Quadratic Forms**

$$x^T D^T D x \triangleq x^T Q x = \text{ Quadratic form}$$

$$D^T D \triangleq Q = \text{ Defining matrix of the quadratic form}$$

- $\dim(Q) = n \times n$
- $Q$ is symmetric
- $x^T Q x$ is a scalar
- $\text{Rank}(Q) \leq m, n > m$
- $\text{Rank}(Q) \leq n, n < m$

**Useful identity for the trace of the quadratic form**

$$x^T Q x = Tr(x^T Q x) = Tr(x x^T Q) = Tr(Q x x^T)$$

$$\begin{bmatrix} (1 \times n)(n \times n)(n \times 1) \end{bmatrix} = \begin{bmatrix} (1 \times 1) \end{bmatrix} = Tr(1 \times 1) = Tr[(n \times n)] = Tr[(n \times n)] = \text{ Scalar}$$
Why are Quadratic Forms Useful?

2 x 2 example

\[
J(x) \triangleq [x^T Q x] = \begin{bmatrix} x_1 & x_2 \\ \vdots & \vdots \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}
\]

\[
= q_{11}x_1^2 + q_{22}x_2^2 + (q_{12} + q_{21})x_1x_2
\]

\[
= q_{11}x_1^2 + q_{22}x_2^2 + 2q_{12}x_1x_2 \text{ for symmetric matrix}
\]

- 2nd-order term of Taylor series expansion of any vector function

\[
f(x_0 + \Delta x) = f(x_0) + \Delta x^T \left[ \frac{\partial f(x)}{\partial x} \right]_{x=x_0} \Delta x + \ldots \text{H.O.T}
\]

- Gradient (as row vector) well-defined everywhere

\[
\frac{\partial J}{\partial x} \triangleq \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2q_{11}x_1 + 2q_{12}x_2 \\ 2q_{22}x_2 + 2q_{12}x_1 \end{bmatrix}
\]

- Large values weighted more heavily than small values
- Some terms can count more than others
- Coupling between terms can be considered
- Asymmetric Q can always be replaced by a symmetric Q

Definiteness
Definiteness of a Matrix

If \( J = x^T Q x > 0 \) for all \( ||x|| \neq 0 \)

a) The scalar, \( J \), is definitely positive
b) The matrix \( Q \) is a **positive definite matrix**

If \( J = x^T Q x \geq 0 \) for all \( ||x|| \neq 0 \)

\( Q \) is a **positive semi-definite matrix**

If \( J = x^T Q x < 0 \) for all \( ||x|| \neq 0 \)

\( Q \) is a **negative definite matrix**

\[
Q = \begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 3
\end{bmatrix}
\]

\[
Q = \begin{bmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
Q = \begin{bmatrix}
-1 & 0 & 0 \\
0 & -2 & 0 \\
0 & 0 & -3
\end{bmatrix}
\]

Positive-Definite Matrix

- \( Q \) is positive-definite if
  - All leading principal minor determinants are positive
  - All eigenvalues are real and positive

\[
Q = \begin{bmatrix}
q_{11} & q_{12} & q_{13} \\
q_{21} & q_{22} & q_{23} \\
q_{31} & q_{32} & q_{33}
\end{bmatrix}
\]

\[
\det(sI - Q) = s^3 + a_2 s^2 + a_1 s + a_0 = (s - \lambda_1)(s - \lambda_2)(s - \lambda_3)
\]

\( \lambda_1, \lambda_2, \lambda_3 > 0 \)

\[
q_{11} > 0, \quad \begin{vmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{vmatrix} > 0, \quad \begin{vmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{vmatrix} > 0
\]
Characteristic Polynomial

Characteristic polynomial of a matrix, $F$

$$|sI - F| \triangleq \det(sI - F) = \text{Scalar}$$

$$\equiv \Delta(s) = s^n + a_{n-1}s^{n-1} + \ldots + a_1s + a_0$$

$$= s^n - Tr(F)s^{n-1} + \ldots + a_1s + a_0$$

where

$$(sI - F) = \begin{pmatrix}
  (s - f_{11}) & -f_{12} & \ldots & -f_{1n} \\
  -f_{21} & (s - f_{22}) & \ldots & -f_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  -f_{n1} & -f_{n2} & \ldots & (s - f_{nn})
\end{pmatrix} \quad (n \times n)$$

Eigenvalues

Characteristic equation of the matrix

$$\Delta(s) = s^n + a_{n-1}s^{n-1} + \ldots + a_1s + a_0 \equiv 0$$

$$= (s - \lambda_1)(s - \lambda_2)(\ldots)(s - \lambda_n) \equiv 0$$

$\lambda_i$ are solutions that set $\Delta(s) = 0$

- They are called
  - the eigenvalues of $F$
  - the roots of the characteristic polynomial

$$a_{n-1} = -Tr(F), \quad a_0 = (-1)^i \prod_{i=1}^{n} \lambda_i$$
Examples of Positive Definite Matrices

Diagonal matrix with positive elements

\[ Q = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 19 \end{bmatrix} \]

Inertia matrix of a rigid body

\[ I = \int_{\text{Body}} \begin{bmatrix} (y^2 + z^2) & -xy & -xz \\ -xy & (x^2 + z^2) & -yz \\ -xz & -yz & (x^2 + y^2) \end{bmatrix} \, dm = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \]

Rotation (Direction Cosine) Matrix

\[ y = Hx \]

- Projections of unit vector components of one Cartesian reference frame on another
- Rotational orientation of one Cartesian reference frame with respect to another

\[ H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \cos \delta_{11} & \cos \delta_{21} & \cos \delta_{31} \\ \cos \delta_{12} & \cos \delta_{22} & \cos \delta_{32} \\ \cos \delta_{13} & \cos \delta_{23} & \cos \delta_{33} \end{bmatrix} \]
**Euler Angles**

- Body attitude measured with respect to inertial frame
- Three-angle orientation expressed by sequence of three orthogonal single-angle rotations from inertial to body frame

\[
\begin{pmatrix}
\cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\
-\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\
\sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta
\end{pmatrix}
\]

\[
\det(H) = 1
\]

**Rotational Transformation**

Position of a particle in two coordinate frames with common origin

\[
y = Hx \quad x = H^{-1}y
\]

Orthonormal transformation \([H^{-1} = H^T]\)

Non-singular matrix \([\det(H) = 1 \neq 0]\)

Not a positive-definite matrix

\[
\|y\|_2 = (y^T y)^{1/2} = (y_1^2 + y_2^2 + y_3^2)^{1/2}
\]

\[
= (x^T H^T H x)^{1/2} = \|Hx\|_2 = \|x\|_2 = \left(x_1^2 + x_2^2 + x_3^2\right)^{1/2}
\]

as \(H^T H = H^{-1}H = I_3\) (positive-definite)
The Static Minimization Problem

Find the value of a continuous control parameter, \( u \), that minimizes a continuous scalar cost function, \( J \)

**Single control parameter**

\[
\min_{\text{wrt } u} J(u); \quad \text{dim}(J) = 1, \quad \text{dim}(u) = 1
\]

\[ u^* = \arg\min_{u} J(u) \triangleq \text{Argument that minimizes } J \]

**Many control parameters**

\[
\min_{\text{wrt } u} J(u); \quad \text{dim}(J) = 1, \quad \text{dim}(u) = m \times 1
\]

\[ u^* = \arg\min_{u} J(u) \]
Necessary Condition for Static Optimality

Single control

\[ \frac{dJ}{du} \bigg|_{u=u^*} = 0 \]

i.e., the slope is zero at the optimum point

Example:

\[ J = (u - 4)^2 \]
\[ \frac{dJ}{du} = 2(u - 4) \]
\[ = 0 \quad \text{when } u^* = 4 \]

Optimum point is a stationary point

Necessary Condition for Static Optimality

Multiple controls

\[ \left. \frac{\partial J}{\partial u} \right|_{u=u^*} = \begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} & \ldots & \frac{\partial J}{\partial u_m} \end{bmatrix}_{u=u^*} = 0 \]

Gradient, defined as a row vector

i.e., all the slopes are concurrently zero at the optimum point

Example:

\[ J = (u_1 - 4)^2 + (u_2 - 8)^2 \]
\[ \frac{dJ}{du_1} = 2(u_1 - 4) = 0 \quad \text{when } u_1^* = 4 \]
\[ \frac{dJ}{du_2} = 2(u_2 - 8) = 0 \quad \text{when } u_2^* = 8 \]

\[ \left. \frac{\partial J}{\partial u} \right|_{u-u^*} = \begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} \end{bmatrix}_{u-u^*} \begin{bmatrix} 4 \\ 8 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} \]

Optimum point is a stationary point
... But the Slope can be Zero for More than One Reason

Minimum

Maximum

Either

Neither (Inflection Point)

... But the Gradient can be Zero for More than One Reason

Minimum

Maximum

Either

Neither (Saddle Point)
Sufficient Condition for Extremum

Single control

- **Minimum**
  - Satisfy necessary condition
  - plus
    \[ \left. \frac{d^2 J}{du^2} \right|_{u=u^*} > 0 \]
  - i.e., the curvature is positive at the optimum point
- **Example:**
  \[
  J = (u - 4)^2 \\
  \frac{dJ}{du} = 2(u - 4) \\
  \frac{d^2 J}{du^2} = 2 > 0
  \]

- **Maximum**
  - Satisfy necessary condition
  - plus
    \[ \left. \frac{d^2 J}{du^2} \right|_{u=u^*} < 0 \]
  - i.e., the curvature is negative at the optimum point
- **Example:**
  \[
  J = -(u - 4)^2 \\
  \frac{dJ}{du} = -2(u - 4) \\
  \frac{d^2 J}{du^2} = -2 < 0
  \]

Sufficient Condition for a Local Minimum

Multiple controls

Satisfy necessary condition

- plus
  \[
  \left. \frac{\partial^2 J}{\partial u_i \partial u_j} \right|_{u=u^*} > 0
  \]
  i.e., \( J \) must be **convex** for a minimum
Minimized Cost Function, \( J^* \)

- **Gradient** is zero at the minimum
- **Hessian matrix** is positive-definite at the local minimum

\[
J(u^* + \Delta u) \approx J(u^*) + \Delta J(u^*) + \Delta^2 J(u^*) + \ldots
\]

\[
\Delta J(u^*) = \Delta u^T \left[ \frac{\partial J}{\partial u} \right]_{u = u^*} = 0
\]

\[
\Delta^2 J(u^*) = \frac{1}{2} \Delta u^T \left[ \frac{\partial^2 J}{\partial u^2} \right]_{u = u^*} \Delta u \geq 0
\]

\[
\therefore J(u^* + \Delta u) \approx J(u^*) + \Delta^2 J(u^*) + \ldots
\]

**Numerical Optimization**
Numerical Optimization

- What if $J$ is too complicated to find an analytical solution for the minimum?
- ... or $J$ has multiple minima?
- Use numerical optimization to find local and/or global solutions

Two Approaches to Numerical Minimization

**Gradient-Based Search**

Evaluate $\frac{\partial J}{\partial u}$ and search for zero

\[
\left( \frac{\partial J}{\partial u} \right)_n = \frac{\partial J}{\partial u} \bigg|_{u=u_0} = \text{Starting guess}
\]

Iterate:

\[
\left( \frac{\partial J}{\partial u} \right)_n = \frac{\partial J}{\partial u} \bigg|_{u=u_{n-1}} + \Delta \left( \frac{\partial J}{\partial u} \right) \bigg|_{u=u_{n-1}} \leq \frac{\partial J}{\partial u} \bigg|_{u=u_n} \text{ such that } \left| \frac{\partial J}{\partial u} \right|_n < \left| \frac{\partial J}{\partial u} \right|_{n-1}
\]

**Gradient-Free Search**

Evaluate $J$ and search directly for $\min J$
Gradient-Free Search
[Based on $J(u)$]

- Exhaustive grid search
- Unstructured random search
- Structured random search

Structured random search
- Nelder-Mead (Downhill Simplex) algorithm
- Simulated annealing
- Genetic algorithm
- Particle-swarm optimization
Downhill Simplex Search
(Nelder-Mead Algorithm)*

- **Simplex**: $N$-dimensional figure in control space defined by
  - $N + 1$ vertices
  - $(N + 1)N/2$ straight edges between vertices

![Diagram of Simplex](https://www.mathworks.com/help/matlab/ref/fminsearch.html)

Search Procedure for Downhill Simplex Method

- Select starting set of vertices, $u(0)$
- Evaluate cost at each vertex
- Determine vertex with largest cost (e.g., $J_1$ at right)

- Project search from this vertex through middle of opposite face (or edge for $N = 2$)
  - **Reflection** [equal distance along direction]
  - **Expansion** [longer distance along direction]
  - **Contraction** [shorter distance along direction]
  - **Shrink** [replace all but best point with points contracted toward best point]

- Evaluate cost at new vertex (e.g., $J_a$)
- Drop $J_1$ vertex, and form simplex with new vertex
- Repeat until cost is “small enough” (termination)
Simulated Annealing Algorithm

- **Goal**: Find global minimum among local minima
- **Approach**: Randomized search, with convergence that emulates *physical annealing*
  - Evaluate cost, $J_i$, with $u(i)$
  - Accept if $J_i < J_{i-1}$
  - Accept with probability $\Pr(E)$ if $J_i > J_{i-1}$
- **Probability distribution of energy state, $E$** *(Boltzmann Distribution)*
  \[
  \Pr(E) \propto e^{-E/kT}
  \]
  $k$: Boltzmann's constant
  $T$: Temperature

- Algorithm’s “cooling schedule” accepts many bad guesses at first, fewer as iteration number, $i$, increases

**Application of Annealing Principle to Search**

- If cost decreases ($J_2 < J_1$), always accept new point
- If cost increases ($J_2 > J_1$), accept new point with probability proportional to *generalized Boltzmann distribution*
  \[
  \Pr(\text{Accept}) \propto e^{-(J_2-J_1)/kT}
  \]
- Occasional diversion from convergent path intended to prevent entrapment by a local minimum
- As search progresses, decrease $kT$, making probability of accepting a cost increase smaller

MATLAB

https://www.mathworks.com/discovery/simulated-annealing.html
Broad Characteristics of Genetic Algorithms

- Search based on the coding of a parameter set, not the parameters themselves
- Search evolves from a population of points
- "Blind" search, i.e., without gradient
- Probabilistic transitions from one control state to another (using random number generator)
- Control parameters assembled as genes of a single chromosome strand (Example: four 4-bit parameters = four "genes")

<table>
<thead>
<tr>
<th>p_1</th>
<th>p_2</th>
<th>p_3</th>
<th>p_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*John Holland, 1975*

Progression of a Genetic Algorithm

Most fit chromosome evolves from a sequence of reproduction, crossover, and mutation

- Initialize algorithm with \( N \) (even) random chromosomes, \( c_n \) (two 8-bit genes or parameters in example)
- Evaluate fitness, \( F_n \), of each chromosome
- Compute total fitness, \( F_{total} \), of chromosome population

\[
F_{total} = \sum_{n=1}^{N} F_n
\]

Bigger \( F \) is better
Genetic Algorithm: Reproduction

- Reproduce $N$ additional copies of the $N$ originals with probabilistic weighting based on relative fitness, $F_n/F_{total}$, of originals (Survival of the fittest)
- Roulette wheel selection:
  - $Pr(c_n) = F_n/F_{total}$
  - Multiple copies of most-fit chromosomes
  - No copies of least-fit chromosomes

Genetic Algorithm: Crossover

- Arrange $N$ new chromosomes in $N/2$ pairs chosen at random
- Interchange tails that are cut at random locations
Create New Generations By
Reproduction, Crossover, and Mutation
Until Solution Converges

Chromosomes narrow in on best values with advancing generations

$F_{max}$ and $F_{total}$ increase with advancing generations

<table>
<thead>
<tr>
<th>Generation</th>
<th>Fitness</th>
<th>Temperature (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>20000</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>30000</td>
</tr>
</tbody>
</table>

MATLAB: Minimizes cost rather than maximizing fitness
https://www.mathworks.com/discovery/genetic-algorithm.html?

Particle Swarm Optimization

- Converse of the GA: Uses multiple cost evaluations to guide parameter search directly
- Stochastic, population-based algorithm
- Search for optimizing parameters modeled on social behavior of groups that possess cognitive consistency
- Particles = Parameter vectors
- Particles have position and velocity
- Projection of own best (Local best)
- Knowledge of swarm’s best
  - Neighborhood best
  - Global best

Peregrine Falcon Hunting Murmuration of Starlings in Rome
https://www.youtube.com/watch?v=V-mCuFYfJdI
Particle Swarm Optimization

Find $\min_u J(u) = J^*(u^*)$

**Jargon:** $\arg\min J(u) = u^*$
* i.e., argument of $J$ that minimizes $J$

Recursive algorithm to find best particle or configuration of particle values

**u:** Parameter vector $\sim$ "Position" of the particles
**v:** "Velocity" of $u$

$\dim(u) = \dim(v) = \text{Number of particles}$


Particle Swarm Optimization

- **Local best:** RNG, downhill simplex, or SA step for each particle
- **Neighborhood best:** argmin of closest $n$ neighboring points
- **Global best:** argmin of all particles

$u_k = u_{k-1} + av_{k-1}$

$v_k = bv_k + c\left(u_{\text{best.LOCAL}_{k-1}} - u_{k-1}\right) + d\left(u_{\text{best.NEIGHBORHOOD}_{k-1}} - u_{k-1}\right) + e\left(u_{\text{best.GLOBAL}_{k-1}} - u_{k-1}\right)$

$u_0$: Starting value from random number generator
$v_0$: Zero

$a, b, c, d$: Search tuning parameters
Gradient Search

Gradient Search to Minimize a Quadratic Function

- Cost function, gradient, and Hessian matrix of a quadratic function
- Guess a starting value of \( u, u_o \)
- Solve gradient equation

\[
J = \frac{1}{2} (u - u*)^T R (u - u*) , \quad R > 0
\]

\[
= \frac{1}{2} (u^T R u - u^T R u* - u*^T R u + u*^T R u*)
\]

- \( \frac{\partial J}{\partial u} = (u - u*)^T R = 0 \) when \( u = u^* \)
- \( \frac{\partial^2 J}{\partial u^2} = R = \text{symmetric constant} \)
- \( \frac{\partial J}{\partial u|_{u=u_o}} = (u_o - u*)^T R = (u_o - u*)^T \frac{\partial^2 J}{\partial u^2|_{u=u_o}} \)

\[
(u_o - u*)^T R^{-1} (row)
\]

\[
u^* = u_o - R^{-1} \left[ \frac{\partial J}{\partial u|_{u=u_o}} \right]^T (column)
\]
Optimal Value Found in a Single Step

• For a quadratic cost function

\[ u^* = u_0 - R^{-1} \left[ \frac{\partial J}{\partial u} \bigg|_{u=u_0} \right]^T \]

• Gradient establishes general search direction
• Hessian fine-tunes direction and tells exactly how far to go

\[ \Delta J(u^*) = \Delta u^T \frac{\partial J}{\partial u} \bigg|_{u=u^*} = 0 \]

\[ \Delta^2 J(u^*) = \frac{1}{2} \Delta u^T \left[ \frac{\partial^2 J}{\partial u^2} \bigg|_{u=u^*} \right] \Delta u \geq 0 \]

Optimal solution requires multiple steps

Newton-Raphson Iteration

• Many cost functions are not quadratic
• However, the surface is well-approximated by a quadratic in the vicinity of the optimum, \( u^* \)
Newton-Raphson Iteration

Newton-Raphson algorithm is an iterative search patterned after the quadratic search

\[ \mathbf{u}_{k+1} = \mathbf{u}_k - \left[ \frac{\partial^2 J}{\partial \mathbf{u}^2} \right]^{-1} \left[ \frac{\partial J}{\partial \mathbf{u}} \right]_{\mathbf{u}=\mathbf{u}_k}^T \]

Difficulties with Newton-Raphson Iteration

- Good when close to the optimum, but ...
- Hessian matrix (i.e., the curvature) may be
  - Difficult to estimate from local measurements of the cost
  - May have the wrong sign (e.g., not positive-definite)
  - May lead to large errors in incremental control variation
Steepest-Descent Algorithm
Multiplies Gradient by a Scalar Constant ("Gain")

\[ u_{k+1} = u_k - \varepsilon \left[ \frac{\partial J}{\partial u} \bigg|_{u=u_k} \right]^T \]

- Replace Hessian matrix by a scalar constant
- Gradient is orthogonal to equal-cost contours

Choice of Steepest-Descent Gain

If gain is too small
Convergence is slow

If gain is too large
Convergence oscillates or may fail

Solution: Make gain adaptive
Optimal Steepest-Descent Gain

Find optimal gain by evaluating cost, \( J \), for intermediate solutions (with same \( \partial J / \partial u \))

Adjustment rule for \( \varepsilon \)
- Starting estimate, \( J_0 \)
- First estimate, \( J_1 \), using \( \varepsilon \)
- Second estimate, \( J_2 \), using \( 2\varepsilon \)

If \( J_2 > J_1 \)
- Quadratic fit through three points to find \( \varepsilon^* \)
- Else, third estimate, \( J_3 \), using \( 4\varepsilon \)
- ...  

Use optimal gain, \( \varepsilon^* \), on each major iteration

Equality Constraints
Static Cost Functions with Equality Constraints

- Minimize $J(u')$, subject to $c(u') = 0$
  - $\dim(c) = [n \times 1]$
  - $\dim(u') = [(m + n) \times 1]$

Two Approaches to Optimization with a Constraint

1. Use constraint to reduce control dimension

2. Augment the cost function to recognize the constraint

Lagrange multiplier, $\lambda$, is an unknown constant

$$\dim(\lambda) = \dim(c) = n \times 1$$

Example: \[
\begin{align*}
\min_{u_1, u_2} J(u_1, u_2) & \quad \text{subject to} \\
c(u') = c(u_1, u_2) &= 0 \rightarrow u_2 = fcn(u_1) \\
\end{align*}
\]
then
\[
J(u') = J(u_1, u_2) = J[u_1, fcn(u_1)] = J'(u_1)
\]

$$J_A(u') = J(u') + \lambda^T c(u')$$

Warning

Lagrange multiplier, $\lambda$, is not the same as an eigenvalue, $\lambda$. 

55

56
Solution Example:
First Approach

Cost function

\[ J = u_1^2 - 2u_1u_2 + 3u_2^2 - 40 \]

Constraint

\[ c = u_2 - u_1 - 2 = 0 \]

\[ \therefore u_2 = u_1 + 2 \]

Solution Example:
Reduced Control Dimension

Cost function and gradient with substitution

\[ J = u_1^2 - 2u_1u_2 + 3u_2^2 - 40 \]

\[ = u_1^2 - 2u_1(u_1 + 2) + 3(u_1 + 2)^2 - 40 \]

\[ = 2u_1^2 + 8u_1 - 28 \]

\[ \frac{\partial J}{\partial u_1} = 4u_1 + 8 = 0 \]

Optimal solution

\[ u_1^* = -2 \]

\[ u_2^* = 0 \]

\[ J^* = -36 \]
Solution: Second Approach

- Partition \( u' \) into a state, \( x \), and a control, \( u \), such that
  - \( \text{dim}(x) = [n \times 1] = \text{dim}(c) \)
  - \( \text{dim}(u) = [m \times 1] \)

\[
\begin{bmatrix}
  x \\
  u
\end{bmatrix}
\]

- Add constraint to the cost function, weighted by Lagrange multiplier, \( \lambda \)

\[
\begin{align*}
J_A(u') &= J(u') + \lambda^T c(u') \\
J_A(x,u) &= J(x,u) + \lambda^T c(x,u)
\end{align*}
\]

- \( c \) is required to be zero when \( J_A \) is a minimum

\[
c(u') = c\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) = 0
\]

Solution: Adjoin Constraint with Lagrange Multiplier

Gradients with respect to \( x, u, \) and \( \lambda \) are zero at the optimum point

\[
\frac{\partial J_A}{\partial x} = \frac{\partial J}{\partial x} + \lambda^T \frac{\partial c}{\partial x} = 0
\]

\[
\frac{\partial J_A}{\partial u} = \frac{\partial J}{\partial u} + \lambda^T \frac{\partial c}{\partial u} = 0
\]

\[
\frac{\partial J_A}{\partial \lambda} = c = 0
\]
Simultaneous Solutions for State and Control

(2n + m) values must be found: \((x,u,\lambda)\)

- **First equation:** find optimal Lagrange multiplier \((n\) scalar equations)
- **Second and third equations:** specify the state and control \((n + m)\) scalar equations

\[
\lambda^* = \frac{-\partial J}{\partial x} \left( \frac{\partial c}{\partial x} \right)^{-1} \quad [\text{Row}]
\]

\[
\lambda^* = -\left[ \left( \frac{\partial c}{\partial x} \right)^{-1} \right]^T \left( \frac{\partial J}{\partial x} \right)^T \quad [\text{Column}]
\]

\[
\frac{\partial J}{\partial u} + \lambda^* \frac{\partial c}{\partial u} = 0
\]

\[
\frac{\partial J}{\partial u} - \frac{\partial J}{\partial x} \left( \frac{\partial c}{\partial x} \right)^{-1} \frac{\partial c}{\partial u} = 0
\]

\[
c(x,u) = 0
\]

Solution Example: Lagrange Multiplier

- **Cost function**

\[
J = u^2 - 2xu + 3x^2 - 40
\]

- **Constraint**

\[
c = x - u - 2 = 0
\]

- **Partial derivatives**

\[
\frac{\partial J}{\partial x} = -2u + 6x \quad \frac{\partial c}{\partial x} = 1
\]

\[
\frac{\partial J}{\partial u} = 2u - 2x \quad \frac{\partial c}{\partial u} = -1
\]
Solution Example: Lagrange Multiplier

- From first equation
  \[ \lambda^* = 2u - 6x \]

- From second equation
  \[ (2u - 2x) + (2u - 6x)(-1) = 0; \quad \therefore \quad x = 0 \]

- From constraint
  \[ u = -2 \]

Optimal solution

\[ x^* = 0 \]
\[ u^* = -2 \]
\[ J^* = -36 \]

Next Time:
Principles for Optimal Control of Dynamic Systems

Reading:
OCE: Sections 3.1, 3.2, 3.4
Supplemental Material

Infinity Norm

• $\infty$ Norm
  – As $p \to \infty$
  – Norm is the value of the maximum component

\[
L^p \text{ norm } = \|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p} \quad \xrightarrow{p \to \infty} \quad \left( \sum_{i=1}^{n} |x_i|^\infty \right)^{1/\infty} = x_{\max} = L^\infty \text{ norm}
\]

• Also called
  – Supremum norm
  – Chebyshev norm
  – Uniform norm

\[
L^\infty \text{ norm } = \|x\|_\infty = \max \{ |x_1|, |x_2|, \ldots, |x_n| \}
\]

\[
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
\]

\[
\dim(\mathbf{x}) = n \times n
\]
Gradient-Free vs. Gradient-Based Searches

- $J$ is a scalar
- $J$ provides no search direction
- Search may provide a global minimum

- $\partial J/\partial u$ is a vector
- $\partial J/\partial u$ indicates feasible search direction
- Search defines a local minimum

Numerical Example

- Cost function and derivatives
  \[ J = \frac{1}{2} (\mathbf{u} - \mathbf{u}^*)^T \mathbf{R} (\mathbf{u} - \mathbf{u}^*), \quad \mathbf{R} > 0 \]
  \[ J = \frac{1}{2} \left[ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right]^T \begin{bmatrix} 1 & 2 & 9 \\ 1 & 2 & 9 \end{bmatrix} \left[ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \right] \]
  \[ \frac{\partial J}{\partial \mathbf{u}} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 9 \\ 1 & 2 & 9 \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix} \]

- First guess at optimal control (details of the actual cost function are unknown)
  \[ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_0 = \begin{bmatrix} 4 \\ 7 \end{bmatrix} \]

- Derivatives evaluated at starting point
  \[ \frac{\partial J}{\partial \mathbf{u}} \bigg|_{\mathbf{u}_0} = \begin{bmatrix} 4 & 7 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 9 \\ 1 & 2 & 9 \end{bmatrix} = \begin{bmatrix} 11 \\ 42 \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix} \]

Solution from starting point

\[ \mathbf{u}^* = \mathbf{u}_0 - \mathbf{R}^{-1} \left[ \frac{\partial J}{\partial \mathbf{u}} \bigg|_{\mathbf{u} = \mathbf{u}_0} \right]^T \]

\[ \mathbf{u}^* = \left[ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right]^* = \left[ \begin{bmatrix} 4 \\ 7 \end{bmatrix} \right] - \begin{bmatrix} 9/5 & -2/5 \\ -2/5 & 1/5 \end{bmatrix} \begin{bmatrix} 11 \\ 42 \end{bmatrix} = \left[ \begin{bmatrix} 4 \\ 7 \end{bmatrix} \right] - \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \]
Conjugate Gradient Algorithm

First step

\[
u_{i+1}(t) = u_i(t) - K_0 \left[ \frac{\partial H}{\partial u}(t) \right]_0^t
\]

Calculate gradient of improved trajectory

Calculate ratio of integrated gradient magnitudes squared

\[
b = \frac{\int \left[ \frac{\partial H}{\partial u}(t) \right] \left[ \frac{\partial H}{\partial u}(t) \right]^T dt}{\int \left[ \frac{\partial H}{\partial u}(t) \right] \left[ \frac{\partial H}{\partial u}(t) \right]^T dt}
\]

Second step

\[
u_{i+1}(t) = u_i(t) - K \left\{ \left[ \frac{\partial H}{\partial u}(t) \right]_0^t - b \left[ \frac{\partial H}{\partial u}(t) \right]^T \right\}
\]

Rosenbrock Function

Typical test function for numerical optimization algorithms

\[
J(u_1, u_2) = (1 - u_1)^2 + 100(u_2 - u_1^2)^2
\]

One minimum
How Many Maxima/Minima does the “Mexican Hat” Have?

\[ z = \text{sinc}(R) = \frac{\sin R}{R} \]

\[ \frac{\partial J}{\partial u_{i} = u^{*}} = 0 \]

\[ \frac{\partial^{2} J}{\partial u_{i} \partial u_{j}} = 0 \]

One maximum

Physical Annealing

- **Produce a strong, hard object made of crystalline material**
  - High temperature allows molecules to redistribute to relieve stress, remove dislocations
  - Gradual cooling allows **large, strong crystals** to form
  - Low temperature “working” (e.g., squeezing, bending, drawing, shearing, and hammering) produces desired crystal structure and shape

**Turbojet Engine**

**Turbines**

**Turbine Blade Casting**

**Single-Crystal**

**Turbine Blade**
Combination of Simulated Annealing with Downhill Simplex Method

- Introduce random "wobble" to simplex search
  - Add random components to costs evaluated at vertices
  - Project new vertex as before based on modified costs
  - With large $T$, this becomes a random search
  - Decrease random components on a "cooling" schedule

- Same annealing strategy as before
  - If cost decreases ($J_2 < J_1$), always accept new point
  - If cost increases ($J_2 > J_1$), accept new point probabilistically
  - As search progresses, decrease $T$

$$J_{1_{SA}} = J_1 + \Delta J_1(rng)$$
$$J_{2_{SA}} = J_2 + \Delta J_2(rng)$$
$$J_{3_{SA}} = J_3 + \Delta J_3(rng)$$
$$\ldots = \ldots$$

Genetic Coding:
Replication, Recombination, and Mutation of Chromosomes
Crossover Creates New Chromosome Population Containing Old Gene Sequences

Reproduction Eliminates Least Fit Chromosomes Probabilistically
Genetic Algorithm: Mutation

Flip a bit, 0 -> 1 or 1 -> 0, at random every 1,000 to 5,000 bits

Crossover Set

Mutated Set

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
P1

| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
P2

Comments on GA

• Short, fit genes tend to survive crossover
• Random location of crossover
  – produces large and small variations in genes
  – interchanges genes in chromosomes
• Multiple copies of best genes evolve
• Alternative implementations
  – Real numbers rather than binary numbers
  – Retention of “elite” chromosomes
  – Clustering in “fit” regions to produce elites

GA Mona Lisa
http://www.youtube.com/watch?v=rGt3iMAJVT8
Comparison of Algorithms in Caterpillar Gait-Training Example

Y. Bourquin, U. Sussex, BIRG
Generalized Direct Search Algorithm

- Choose optimal elements of $K$ by sequential line search before recalculating the gradient

$$u_{k+1}(t) = u_k(t) - K_k \left[ \frac{\partial J}{\partial u}(t) \right]_k^T$$

$$K_k = \begin{bmatrix} k_{11} & \ldots & \ldots \\ \ldots & k_{22} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}$$

Ad Hoc Modifications to the Newton-Raphson Search

$$u_{k+1} = u_k - \varepsilon \left[ \frac{\partial^2 J}{\partial u^2} \right]_{u_k} \left[ \frac{\partial J}{\partial u} \right]_{u_k}^T, \quad \varepsilon < 1$$

$$u_{k+1} = u_k - \begin{bmatrix} \frac{\partial^2 J}{\partial u_1^2} & 0 & 0 \\ 0 & \ldots & 0 \\ 0 & 0 & \frac{\partial^2 J}{\partial u_m^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial J}{\partial u_1} \big|_{u_k} \\ \vdots \\ \frac{\partial J}{\partial u_m} \big|_{u_k} \end{bmatrix}^T$$

$$u_{k+1} = u_k - \left[ \frac{\partial^2 J}{\partial u^2} \big|_{u_k} \right]^{-1} \left[ \frac{\partial J}{\partial u} \right]_{u_k}^T + \varepsilon I$$

$K > 0$, diagonal

- With varying $\varepsilon$, this is the Levenberg-Marquardt algorithm