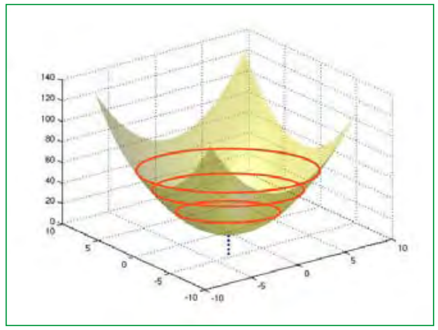
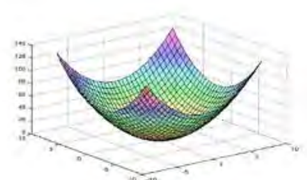
Minimization of Static Cost Functions

Robert Stengel
Optimal Control and Estimation, MAE 546, Princeton University, 2015

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



Static Cost Function



- Minimum value of the cost, J, is fixed, but may be unknown
- Corresponding control parameter, $\mathbf{u}^*$, 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, x
  - Scalar quantity

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

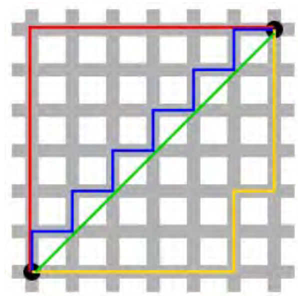
$$\dim(\mathbf{x}) = n \times 1$$

- Taxicab or Manhattan norm

$$L^1\ norm = \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

- Euclidean or Quadratic Norm

$$L^2\ norm = \|\mathbf{x}\|_2 = \left(\mathbf{x}^T\mathbf{x}\right)^{1/2} = \left(x_1^2 + x_2^2 + \cdots + x_n^2\right)^{1/2}$$



- *p* Norm

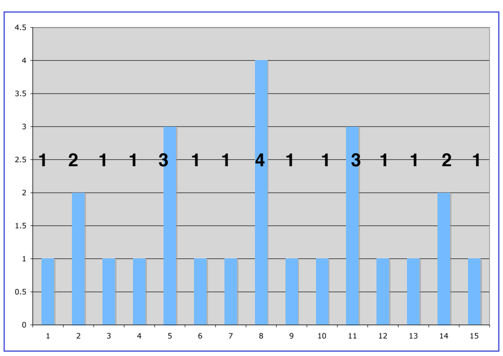
$$L^p\ norm = \|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

3

p Norm Example

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

- 15-dimensional vector



- *p* norms of the vector

p	L^p
1	24.0000
2	7.2111
4	4.6312
8	4.0957
16	4.0050
32	4.0000

- 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**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} Velocity, m/s \\ Angle, rad \\ \cdots \\ Temperature, ^\circ K \end{bmatrix}$$

- **What is a reasonable definition for the norm?**
- **<u>One solution</u>: Vector, y, normalized by ranges of elements of x**

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} \equiv \begin{bmatrix} d_1 x_1 \\ d_2 x_2 \\ \cdots \\ d_n x_n \end{bmatrix} \equiv \begin{bmatrix} x_1 / \left(x_{1_{\max}} - x_{1_{\min}} \right) \\ x_2 / \left(x_{2_{\max}} - x_{2_{\min}} \right) \\ \cdots \\ x_n / \left(x_{n_{\max}} - x_{n_{\min}} \right) \end{bmatrix} = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \mathbf{Dx}$$

Weighted Euclidean (Quadratic) Norm of x

$$\|\mathbf{y}\|_2 = \left(\mathbf{y}^T \mathbf{y} \right)^{1/2} = \left(y_1^2 + y_2^2 + \cdots + y_m^2 \right)^{1/2}$$
$$= \left(\mathbf{x}^T \mathbf{D}^T \mathbf{D} \mathbf{x} \right)^{1/2} = \|\mathbf{Dx}\|_2$$

$$\dim(\mathbf{y}) = m \times 1$$
$$\dim(\mathbf{x}) = n \times 1$$
$$\dim(\mathbf{D}) = m \times n$$

- **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**
- **Rank(D) ≤ m, $n > m$**
- **Rank(D) ≤ n, $n < m$**

Rank of Matrix D

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

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

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

Quadratic Forms

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

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

- dim(**Q**) = n x n
- **Q** is symmetric
- $\mathbf{x}^T\mathbf{Q}\mathbf{x}$ is a scalar

- Rank(**Q**) ≤ m, $n > m$
- Rank(**Q**) ≤ n, $n < m$

- Useful identity for the trace of the quadratic form

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} = Tr\left(\mathbf{x}^T\mathbf{Q}\mathbf{x}\right) = Tr\left(\mathbf{x}\mathbf{x}^T\mathbf{Q}\right) = Tr\left(\mathbf{Q}\mathbf{x}\mathbf{x}^T\right)$$

$$\left[(1\times n)(n\times n)(n\times 1)\right] = \left[(1\times 1)\right] = Tr(1\times 1) = Tr\left[(n\times n)\right] = Tr\left[(n\times n)\right] = \text{ Scalar}$$

Why are Quadratic Forms Useful?

2 x 2 example

$$J = \left[\mathbf{x}^T \mathbf{Q}\mathbf{x}\right] = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$= q_{11}x_1^2 + q_{22}x_2^2 + \left(q_{12} + q_{21}\right)x_1 x_2$$
$$= q_{11}x_1^2 + q_{22}x_2^2 + 2q_{12}x_1 x_2 \text{ for symmetric matrix}$$

- **Asymmetric Q can always be replaced by a symmetric Q**
- **Large values are weighted more heavily than small values**
- **Some terms can count more than others**
- **Coupling between terms can be considered**
- **<u>Gradient</u> is well-defined everywhere**

$$\partial J/\partial \mathbf{x} = \begin{bmatrix} \partial J/\partial x_1 & \partial J/\partial x_2 \end{bmatrix} = \begin{bmatrix} \left(2q_{11}x_1 + 2q_{12}x_2\right) & \left(2q_{22}x_2 + 2q_{12}x_1\right) \end{bmatrix}$$

Definiteness

Definiteness of a Matrix

If $\mathbf{x}^T\mathbf{Qx} > 0$ for all $\|\mathbf{x}\| \neq 0$

a) The scalar is definitely positive

b) The matrix $\mathbf{Q}$ is a **positive definite matrix**

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

If $\mathbf{x}^T\mathbf{Qx} \geq 0$ for all $\|\mathbf{x}\| \neq 0$

$\mathbf{Q}$ is a **positive semi - definite matrix**

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

If $\mathbf{x}^T\mathbf{Qx} < 0$ for all $\|\mathbf{x}\| \neq 0$

$\mathbf{Q}$ is a **negative definite matrix**

$$\mathbf{Q} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix}$$

Positive-Definite Matrix

- **Q is positive-definite if**
  - **All leading principal minor determinants are positive**
  - **All eigenvalues are real and positive**

3 x 3 example

$$\det(s\mathbf{I} - \mathbf{Q}) = s^3 + a_2 s^2 + a_1 s + a_0$$
$$= (s - \lambda_1)(s - \lambda_2)(s - \lambda_3)$$
$$\lambda_1, \lambda_2, \lambda_3 > 0$$

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

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

What's an eigenvalue?

Characteristic Polynomial

Characteristic polynomial of a matrix, F

$$
\begin{aligned}
|s\mathbf{I}-\mathbf{F}| &= \det(s\mathbf{I}-\mathbf{F}) = \textbf{Scalar}\\
&\equiv \Delta(s) = s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0\\
&= s^n - Tr(\mathbf{F})s^{n-1} + \ldots + a_1 s + a_0
\end{aligned}
$$

where

$$
(s\mathbf{I}-\mathbf{F}) = \begin{pmatrix} (s-f_{11}) & -f_{12} & \ldots & -f_{1n} \\ -f_{21} & (s-f_{22}) & \ldots & -f_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ -f_{n1} & -f_{n2} & \ldots & (s-f_{nn}) \end{pmatrix} \quad (n\ x\ n)
$$

Eigenvalues

Characteristic equation of the matrix

$$
\begin{aligned}
\Delta(s) &= s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0 \equiv 0\\
&= (s-\lambda_1)(s-\lambda_2)(\ldots)(s-\lambda_n) \equiv 0
\end{aligned}
$$

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

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

$$
a_{n-1} = ?\\
a_0 = ?
$$

$$
a_{n-1} = -Tr(\mathbf{F})\\
a_0 = (-1)^i \prod_{i=1}^{n} \lambda_i
$$

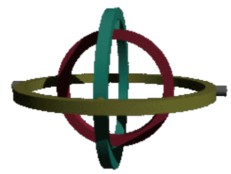
Examples of Positive Definite Matrices

Inertia matrix of a rigid body

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

Diagonal matrix with positive elements

$$\mathbf{Q} = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 19 \end{bmatrix}$$



Rotation matrix

$$\mathbf{H}_I^B(\phi,\theta,\psi) = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}$$

Conditions for a Static Minimum

The Static Minimization Problem

- **Find the value of a continuous control parameter, u, that minimizes a continuous scalar cost junction, *J***
  - **Single control parameter**

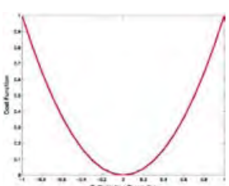
$$\boxed{\min_{wrt\ u} J(u); \quad \dim(J) = 1, \quad \dim(u) = 1}$$

$$\boxed{u^* = \arg\min_{u} J(u)}$$

  - **Many control parameters**

$$\boxed{\min_{wrt\ \mathbf{u}} J(\mathbf{u}); \quad \dim(J) = 1, \quad \dim(\mathbf{u}) = m \times 1}$$

$$\boxed{\mathbf{u}^* = \arg\min_{\mathbf{u}} J(\mathbf{u})}$$



Necessary Condition for Static Optimality

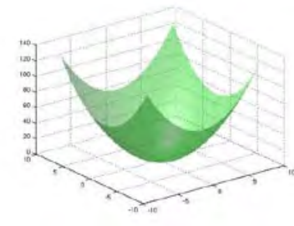
- **Single control**

$$\boxed{\left.\frac{dJ}{du}\right|_{u=u^*} = 0}$$

- **i.e., the slope is zero at the optimum point**
- **Example:**

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

Necessary Condition for Static Optimality



- **Multiple controls**

$$\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^*} = \left.\begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} & \cdots & \frac{\partial J}{\partial u_m} \end{bmatrix}\right|_{\mathbf{u}=\mathbf{u}^*} = \mathbf{0}$$

Gradient*, *defined as a* *row vector

- **i.e., all the slopes are concurrently zero at the optimum point**
- **Example:**

$$J = (u_1 - 4)^2 + (u_2 - 8)^2$$

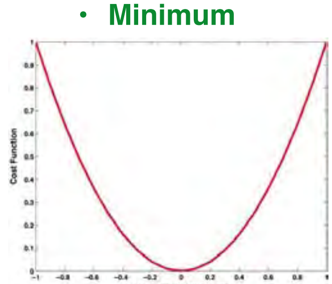
$$\frac{dJ}{du_1} = 2(u_1 - 4) = 0 \quad when\ u_1^* = 4$$

$$\frac{dJ}{du_2} = 2(u_2 - 8) = 0 \quad when\ u_2^* = 8$$

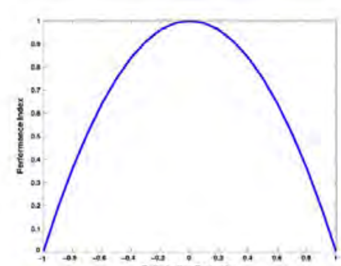
$$\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^*} = \left.\begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} \end{bmatrix}\right|_{\mathbf{u}=\mathbf{u}^*=\begin{bmatrix} 4 \\ 8 \end{bmatrix}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

… But the Slope can be Zero for More than One Reason

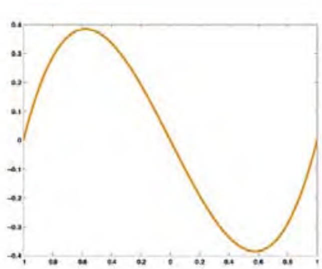
- **Minimum**



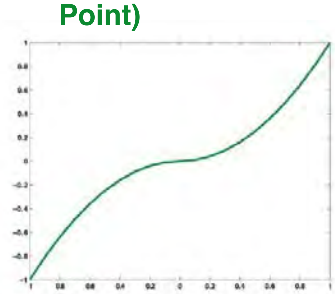
- **Maximum**



- **Either**

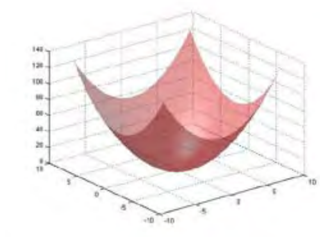


- **Neither (Inflection Point)**

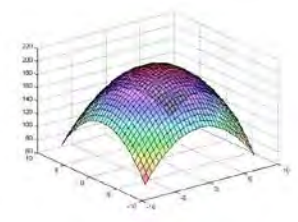


… But the Gradient can be Zero for More than One Reason

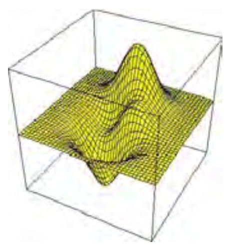
- Minimum



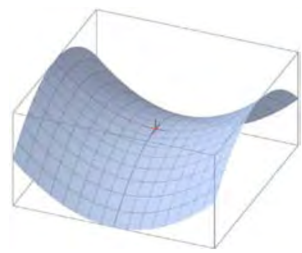
- Maximum

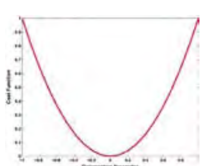


- Either

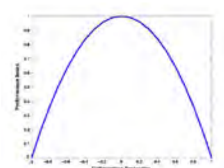


- Neither (Saddle Point)





Sufficient Condition for Extremum



Single control

- **Minimum**
  - Satisfy necessary condition
  - plus

$$\boxed{\left.\frac{d^2J}{du^2}\right|_{u=u^*} > 0}$$

  - i.e., the curvature is positive at the optimum point
- Example:

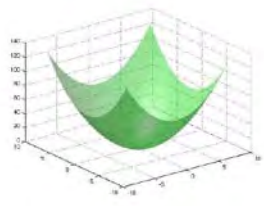
$$\boxed{\begin{aligned} J &= (u-4)^2 \\ \frac{dJ}{du} &= 2(u-4) \\ \frac{d^2J}{du^2} &= 2 > 0 \end{aligned}}$$

- **Maximum**
  - Satisfy necessary condition
  - plus

$$\boxed{\left.\frac{d^2J}{du^2}\right|_{u=u^*} < 0}$$

  - i.e., the curvature is negative at the optimum point
- Example:

$$\boxed{\begin{aligned} J &= -(u-4)^2 \\ \frac{dJ}{du} &= -2(u-4) \\ \frac{d^2J}{du^2} &= -2 < 0 \end{aligned}}$$



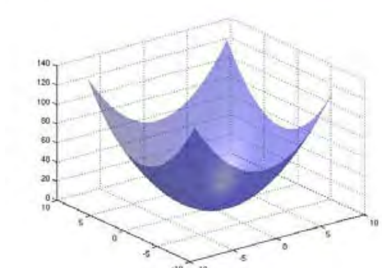
Sufficient Condition for a Minimum

Multiple controls

- Satisfy necessary condition
  - plus

Hessian matrix

$$\left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}^*} = \left.\begin{bmatrix} \frac{\partial^2 J}{\partial u_1^2} & \frac{\partial^2 J}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 J}{\partial u_1 \partial u_m} \\ \frac{\partial^2 J}{\partial u_2 \partial u_1} & \frac{\partial^2 J}{\partial u_2^2} & \cdots & \frac{\partial^2 J}{\partial u_2 \partial u_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 J}{\partial u_m \partial u_1} & \frac{\partial^2 J}{\partial u_2 \partial u_m} & \cdots & \frac{\partial^2 J}{\partial u_m^2} \end{bmatrix}\right|_{\mathbf{u}=\mathbf{u}^*} > \mathbf{0}$$



Minimized Cost Function, *J**

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

$$J(\mathbf{u}^* + \Delta\mathbf{u}) \approx J(\mathbf{u}^*) + \Delta J(\mathbf{u}^*) + \Delta^2 J(\mathbf{u}^*) + \ldots$$

$$\Delta J(\mathbf{u}^*) = \left[\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^*}\right]\Delta\mathbf{u} = 0$$

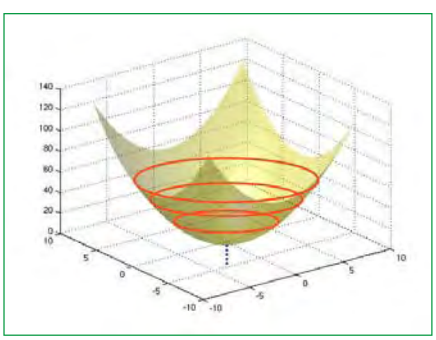
$$\Delta^2 J(\mathbf{u}^*) = \frac{1}{2}\Delta\mathbf{u}^T \left[\left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}^*}\right]\Delta\mathbf{u} \geq 0$$

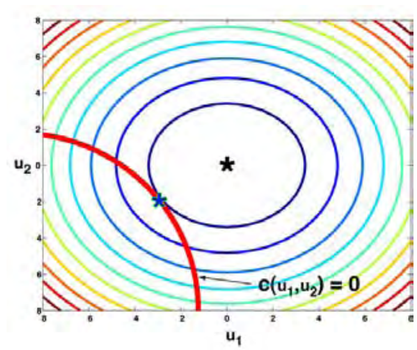
- First variation is zero at the minimum
- Second variation is positive at the minimum

Equality Constraints

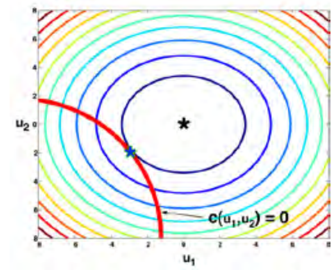
Static Cost Functions with Equality Constraints

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





Two Approaches to Static Optimization with a Constraint



1. **Use constraint to reduce control dimension**

$$\mathbf{u}' = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$Example: \quad \min_{u_1,u_2} J \quad subject\ to$$
$$c(\mathbf{u}') = c(u_1,u_2) = 0 \rightarrow u_2 = fcn(u_1)$$
then
$$J(\mathbf{u}') = J(u_1,u_2) = J\left[u_1, fcn(u_1)\right] = J'(u_1)$$

2. **Augment the cost function to recognize the constraint**

Lagrange multiplier, $\boldsymbol{\lambda}$, is an unknown constant

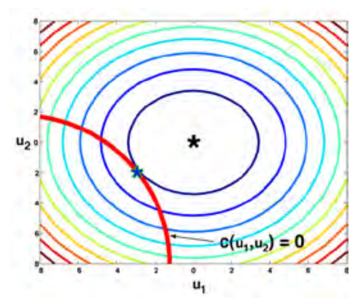
$$\dim(\boldsymbol{\lambda}) = \dim(\mathbf{c}) = n \times 1$$

$$J_A(\mathbf{u}') = J(\mathbf{u}') + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{u}')$$

Warning

Lagrange multiplier, $\boldsymbol{\lambda}$, is not the same as an eigenvalue, λ

Solution Example: First Approach



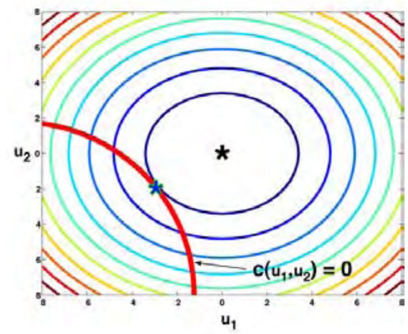
Cost function

$$J = u_1^2 - 2u_1u_2 + 3u_2^2 - 40$$

Constraint

$$c = u_2 - u_1 - 2 = 0$$
$$\therefore u_2 = u_1 + 2$$

Solution Example: Reduced Control Dimension

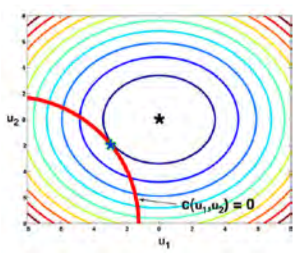


Cost function and gradient with substitution

$$
\begin{aligned}
J &= u_1^2 - 2u_1u_2 + 3u_2^2 - 40 \\
&= u_1^2 - 2u_1(u_1+2) + 3(u_1+2)^2 - 40 \\
&= 2u_1^2 + 8u_1 - 28 \\
\frac{\partial J}{\partial u_1} &= 4u_1 + 8 = 0
\end{aligned}
$$

Optimal solution

$$
\begin{aligned}
u_1^* &= -2 \\
u_2^* &= 0 \\
J^* &= -36
\end{aligned}
$$



Solution: Second Approach

- Partition **u'** into a state, **x**, and a control, **u**, such that
  - dim(x) = [*n* x 1] = dim(c)
  - dim(u) = [*m* x 1]

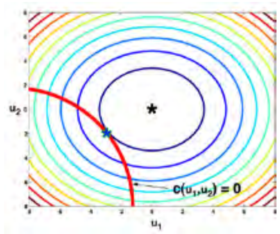
$$\mathbf{u}' = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix}$$

- Add constraint to the cost function, weighted by **Lagrange multiplier,** λ

$$
\begin{aligned}
J_A(\mathbf{u}') &= J(\mathbf{u}') + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{u}') \\
J_A(\mathbf{x},\mathbf{u}) &= J(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x},\mathbf{u})
\end{aligned}
$$

- **c** is required to be zero when J_A is a minimum

$$\mathbf{c}(\mathbf{u}') = \mathbf{c}\begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix} = \mathbf{0}$$



Solution: Adjoin Constraint with Lagrange Multiplier

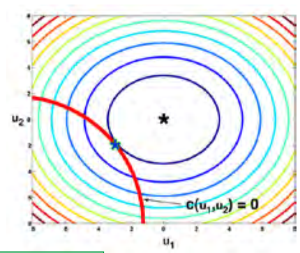
Gradients with respect to $\mathbf{x}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$ are zero at the optimum point

$$\frac{\partial J_A}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{c}}{\partial \mathbf{x}} = \mathbf{0}$$

$$\frac{\partial J_A}{\partial \mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{c}}{\partial \mathbf{u}} = \mathbf{0}$$

$$\frac{\partial J_A}{\partial \boldsymbol{\lambda}} = \mathbf{c} = \mathbf{0}$$

Simultaneous Solutions for State and Control



$(2n+m)$ values must be found: $(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})$

- **First equation: form for optimizing Lagrange multiplier (*n* scalar equations)**
- **Second and third equations: (*n + m*) scalar equations that specify the state and control**

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

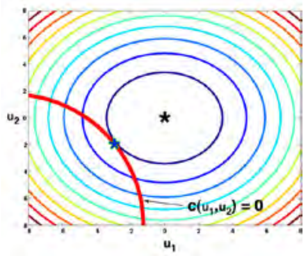
or

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

$$\frac{\partial J}{\partial \mathbf{u}} + \boldsymbol{\lambda}^{*T} \frac{\partial \mathbf{c}}{\partial \mathbf{u}} = \mathbf{0}$$

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

$$\mathbf{c}(\mathbf{x},\mathbf{u}) = \mathbf{0}$$



Solution Example: Lagrange Multiplier

- **Cost function**

$$\boxed{J = u^2 - 2xu + 3x^2 - 40}$$

- **Constraint**

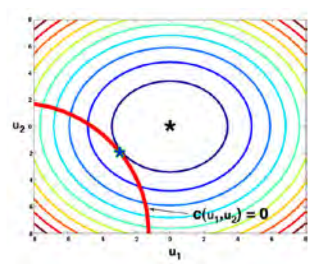
$$\boxed{c = x - u - 2 = 0}$$

- **Partial derivatives**

$$\frac{\partial J}{\partial x} = -2u + 6x \qquad \frac{\partial c}{\partial x} = 1$$

$$\frac{\partial J}{\partial u} = 2u - 2x \qquad \frac{\partial c}{\partial u} = -1$$

Solution Example: Lagrange Multiplier



- **From first equation** $\boxed{\lambda^* = 2u - 6x}$

- **From second equation**

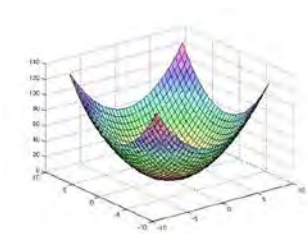
$$(2u - 2x) + (2u - 6x)(-1)$$
$$\therefore x = 0$$

- **From constraint** $\boxed{u = -2}$

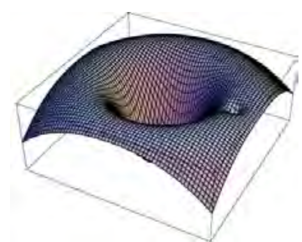
Optimal solution

$$x^* = 0$$
$$u^* = -2$$
$$J^* = -36$$

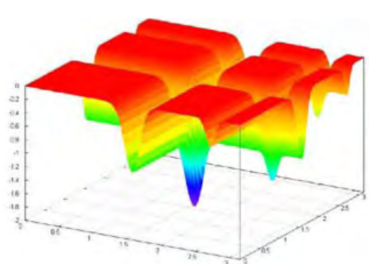
Numerical Optimization

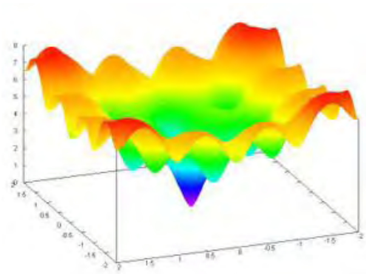


Numerical Optimization

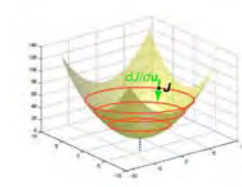


- What if *J* is too complicated to find an analytical solution for the minimum?
- … or *J* has multiple minima?
- Use numerical optimization to find local and/or global solutions





Two Approaches to Numerical Minimization



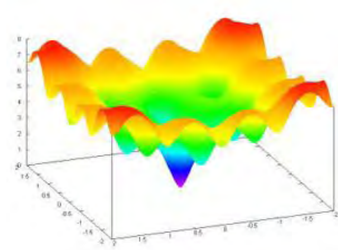
- **Evaluate $\partial J/\partial \mathbf{u}$ and search for zero [Gradient-Based Search]**

$$\left(\frac{\partial J}{\partial \mathbf{u}}\right)_o = \left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_0} = \textit{starting guess}$$

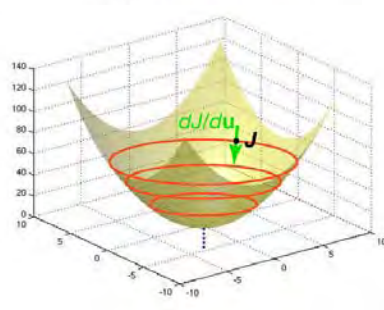
$$\left(\frac{\partial J}{\partial \mathbf{u}}\right)_n = \left(\frac{\partial J}{\partial \mathbf{u}}\right)_{n-1} + \Delta\left(\frac{\partial J}{\partial \mathbf{u}}\right)_n = \left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_n} \quad \textit{such that} \quad \left|\frac{\partial J}{\partial \mathbf{u}}\right|_n < \left|\frac{\partial J}{\partial \mathbf{u}}\right|_{n-1}$$

Gradient-Free vs. Gradient-Based Searches

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



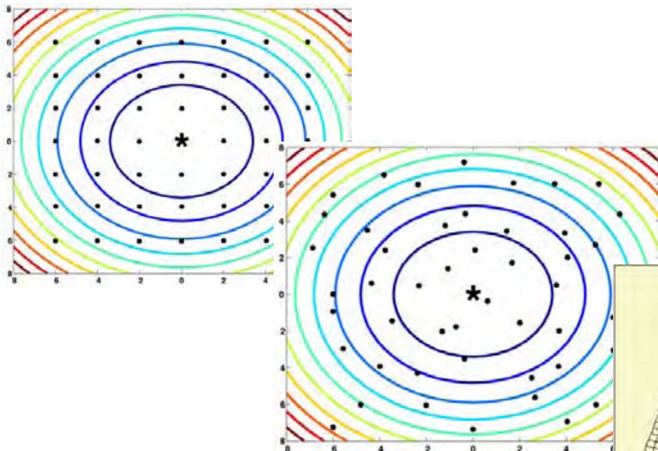
- **$\partial J/\partial \mathbf{u}$ is a vector**
- **$\partial J/\partial \mathbf{u}$ indicates feasible search direction**
- **Search defines a local minimum**

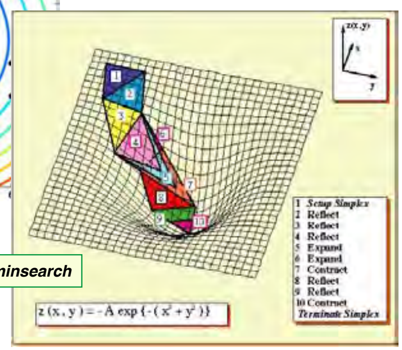


Gradient-Free Search

(Based on J)

- Exhaustive grid search
- Unstructured random search
- Downhill-Simplex (Nelder-Mead) algorithm

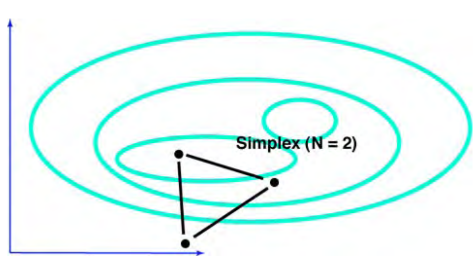


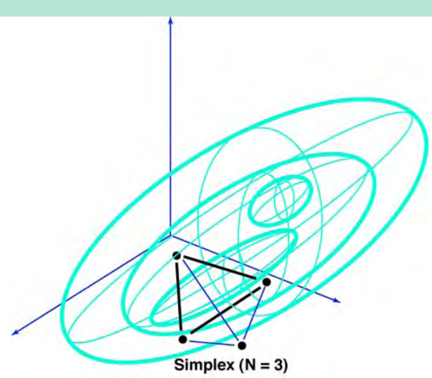


MATLAB's *fminsearch*

Downhill Simplex Search (Nelder-Mead Algorithm)

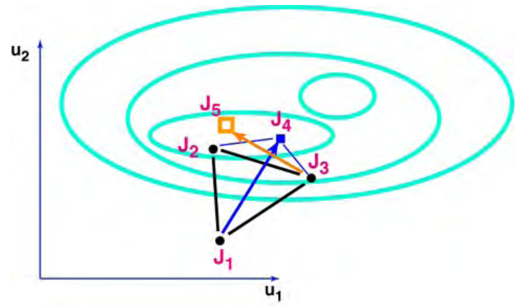
- Simplex: *N*-dimensional figure in control space defined by
  - $N + 1$ vertices
  - $(N + 1)\ N / 2$ straight edges between vertices





Search Procedure for Downhill Simplex Method

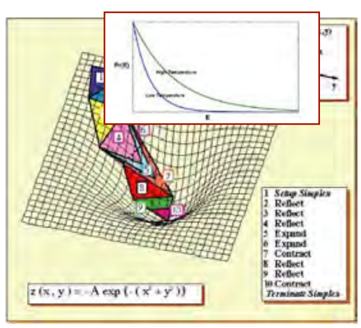
- **Select starting set of vertices**
- **Evaluate cost at each vertex**
- **Determine vertex with largest cost (e.g., J_1 at right)**

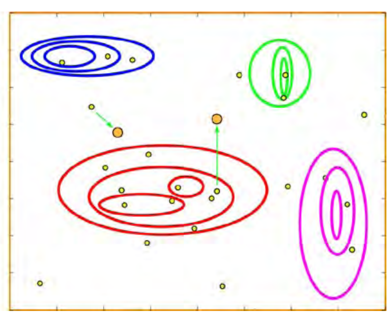


  - **Project search from this vertex through middle of opposite face (or edge for $N = 2$)**
  - **Evaluate cost at new vertex (e.g., J_4 at right)**
  - **Drop J_1 vertex, and form simplex with new vertex**
  - **Repeat until cost is small**

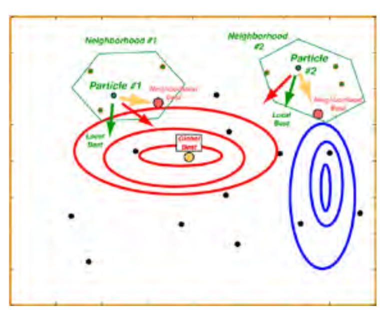
Humanoid Walker optimized via Nelder-Mead
http://www.youtube.com/watch?v=BcYPLR_j5dg

Gradient-Free Search



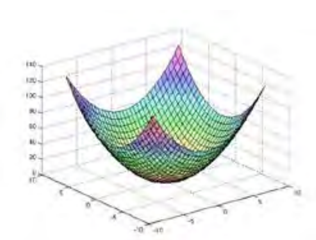


- **Structured random search**
  - **Nelder-Mead (Downhill Simplex) algorithm**
  - **Simulated annealing**
    - **shown as modification to D-S method**
  - **Genetic algorithm**
    - **crossover, reproduction, mutation of parameter codes**
  - **Particle-swarm optimization**
    - **positions and velocities of parameter swarm**



MATLAB Implementations of Gradient-Free Search

- **Nelder-Mead (Downhill Simplex) algorithm**
  - http://www.mathworks.com/help/matlab/ref/fminsearch.html
- **Simulated annealing**
  - http://www.mathworks.com/help/gads/simulated-annealing.html
- **Genetic algorithm**
  - http://www.mathworks.com/help/gads/genetic-algorithm.html
  - Example: edit gaconstrained
- **Particle-swarm optimization**
  - http://www.mathworks.com/help/gads/particle-swarm.html



Gradient Search to Minimize a Quadratic Function

- **Cost function, gradient, and Hessian matrix of a quadratic function**

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

$$= \frac{1}{2}\left(\mathbf{u}^T\mathbf{R}\mathbf{u} - \mathbf{u}^T\mathbf{R}\mathbf{u}^* - \mathbf{u}^{*T}\mathbf{R}\mathbf{u} + \mathbf{u}^{*T}\mathbf{R}\mathbf{u}^*\right)$$

$$\frac{\partial J}{\partial \mathbf{u}} = (\mathbf{u}-\mathbf{u}^*)^T \mathbf{R} = \mathbf{0} \; when \; \mathbf{u} = \mathbf{u}^*$$

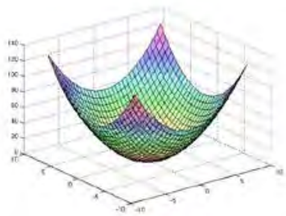
$$\frac{\partial^2 J}{\partial \mathbf{u}^2} = \mathbf{R} = symmetric\ constant$$

- **Guess a starting value of u, $\mathbf{u}_o$**
- **Solve gradient equation**

$$\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_o} = (\mathbf{u}_o-\mathbf{u}^*)^T \mathbf{R} = (\mathbf{u}_o-\mathbf{u}^*)^T \left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}_o}$$

$$(\mathbf{u}_o-\mathbf{u}^*)^T = \left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_o} \mathbf{R}^{-1} \quad (row)$$

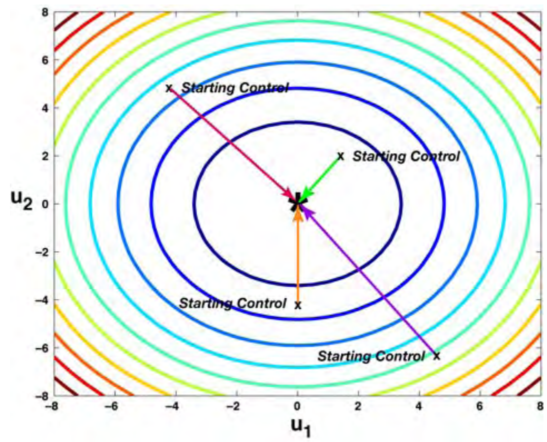
$$\mathbf{u}^* = \mathbf{u}_o - \mathbf{R}^{-1}\left[\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_o}\right]^T \quad (column)$$



Optimal Value Found in a Single Step

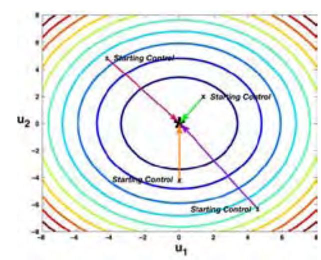
- **For a quadratic cost function**

$$\mathbf{u}^* = \mathbf{u}_o - \mathbf{R}^{-1}\left[\frac{\partial J}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}_o}\right]^T$$
$$= \mathbf{u}_o - \left[\frac{\partial^2 J}{\partial \mathbf{u}^2}\bigg|_{\mathbf{u}=\mathbf{u}_o}\right]^{-1}\left[\frac{\partial J}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}_o}\right]^T$$



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

Numerical Example



- **Cost function and derivatives**

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

$$J = \frac{1}{2}\left\{\left[\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - \begin{pmatrix} 1 \\ 3 \end{pmatrix}\right]^T \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix}\left[\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - \begin{pmatrix} 1 \\ 3 \end{pmatrix}\right]\right\}$$

$$\left(\frac{\partial J}{\partial \mathbf{u}}\right)^T = \left[\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - \begin{pmatrix} 1 \\ 3 \end{pmatrix}\right]^T \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix}$$

- **First guess at optimal control (details of the actual cost function are unknown)**

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}_0 = \begin{pmatrix} 4 \\ 7 \end{pmatrix}$$

- **Derivatives evaluated at starting point**

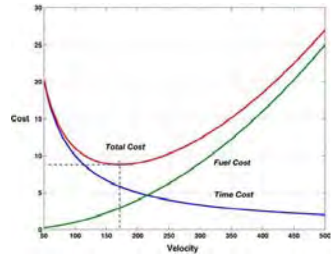
$$\frac{\partial J}{\partial \mathbf{u}}\bigg|_{\mathbf{u}=\mathbf{u}_0} = \left[\begin{pmatrix} 4 \\ 7 \end{pmatrix} - \begin{pmatrix} 1 \\ 3 \end{pmatrix}\right]^T \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix} = \begin{pmatrix} 11 \\ 42 \end{pmatrix}; \quad \mathbf{R} = \begin{bmatrix} 1 & 2 \\ 2 & 9 \end{bmatrix}$$

Solution from starting point

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

$$\mathbf{u}^* = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}^* = \begin{pmatrix} 4 \\ 7 \end{pmatrix} - \begin{bmatrix} 9/5 & -2/5 \\ -2/5 & 1/5 \end{bmatrix}\begin{pmatrix} 11 \\ 42 \end{pmatrix}$$
$$= \begin{pmatrix} 4 \\ 7 \end{pmatrix} - \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

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***

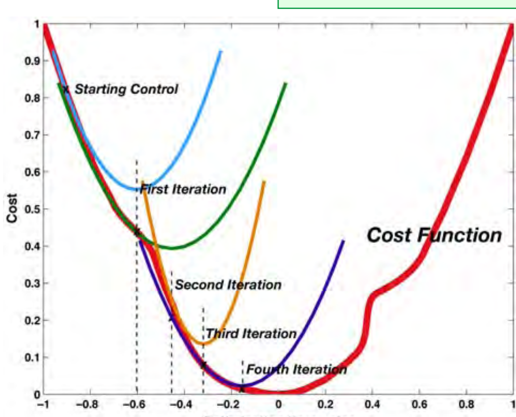
$$
\begin{aligned}
&J(\mathbf{u}^* + \Delta\mathbf{u}) \approx J(\mathbf{u}^*) + \Delta J(\mathbf{u}^*) + \Delta^2 J(\mathbf{u}^*) + \ldots \\
&\Delta J(\mathbf{u}^*) = \Delta\mathbf{u}^T \left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^*} = 0 \\
&\Delta^2 J(\mathbf{u}^*) = \frac{1}{2}\Delta\mathbf{u}^T \left[\left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}^*}\right]\Delta\mathbf{u} \geq 0
\end{aligned}
$$

Optimal solution requires multiple steps

Newton-Raphson Iteration

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

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

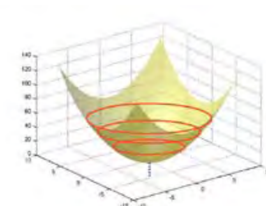


Difficulties with Newton-Raphson Iteration

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

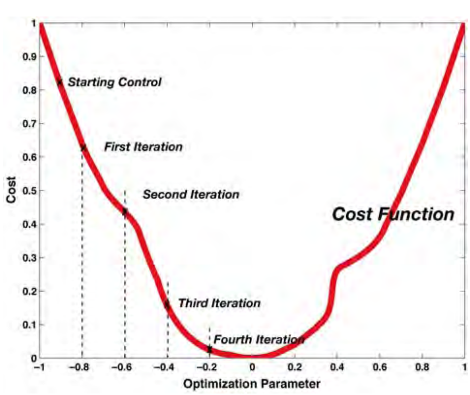
- **Good when close to the optimum, but …**
- **Hessian matrix (i.e., the curvature) may be**
  - **Difficult to estimate from local measurements of the cost**
  - **May have the wrong sign (e.g., not positive-definite)**
  - **May lead to large errors in incremental control variation**

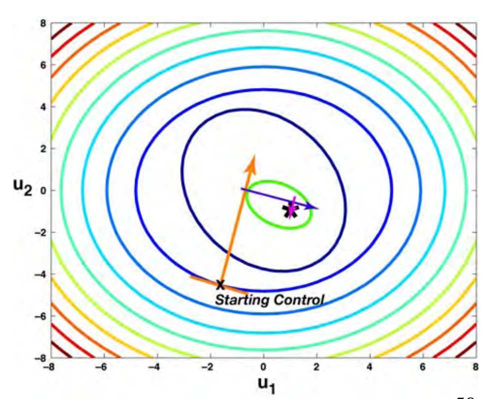
Steepest-Descent Algorithm Multiplies Gradient by a Scalar Constant ("Gain")

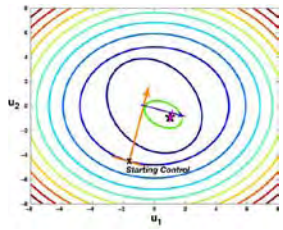


$$\mathbf{u}_{k+1} = \mathbf{u}_k - \varepsilon \left[\left. \frac{\partial J}{\partial \mathbf{u}} \right|_{\mathbf{u}=\mathbf{u}_k} \right]^T$$

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

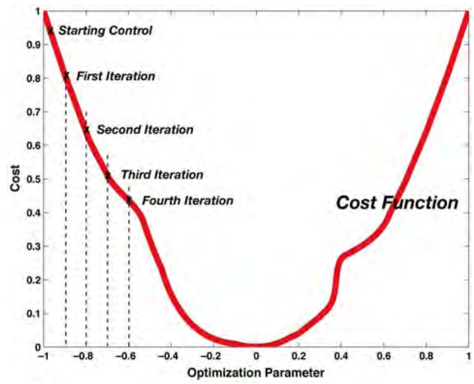




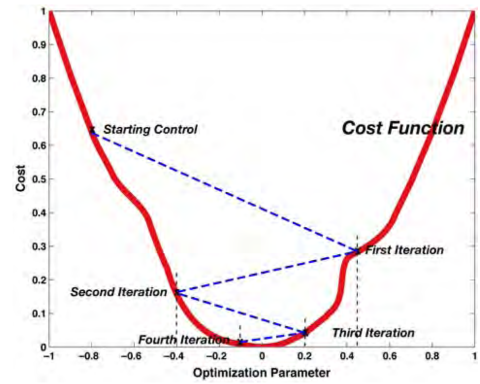


Choice of Steepest-Descent Constant

- **If gain is too small**
  - Convergence is slow

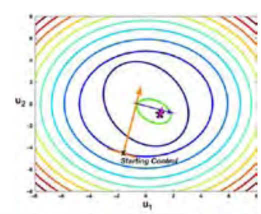


- **If gain is too large**
  - Convergence oscillates or may fail



Solution: Make gain adaptive

Optimal Steepest-Descent Gain

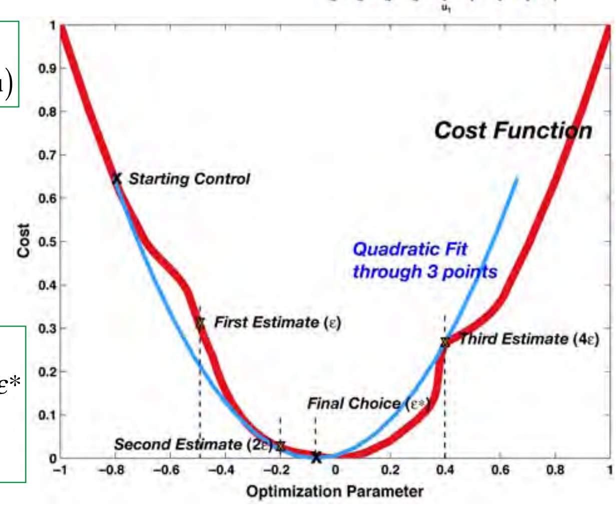


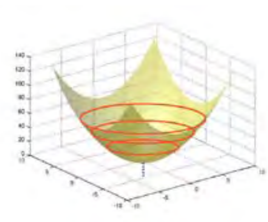
Find optimal gain by evaluating cost, J, for intermediate solutions (with same $\partial J/\partial \mathbf{u}$)

Adjustment rule for ε
- Starting estimate, J_0
- First estimate, J_1, using ε
- Second estimate, J_2, using 2ε

If $J_2 > J_1$
- Quadratic fit through three points to find ε^*
- Else, third estimate, J_3, using 4ε
- …

Use optimal gain, ε^*, on each major iteration



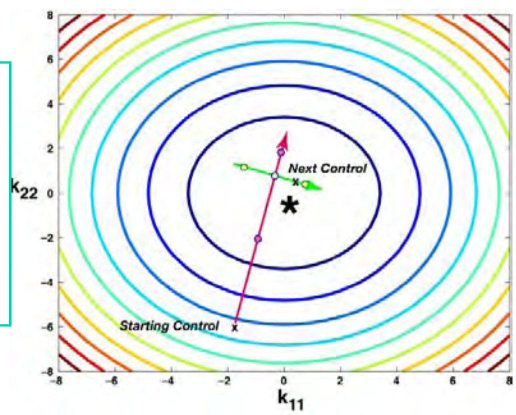


Generalized Direct Search Algorithm

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t) - \mathbf{K}_k \left[\frac{\partial J}{\partial \mathbf{u}}(t)\right]_k^T$$

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

$$\mathbf{K}_k = \begin{bmatrix} k_{11} & \dots & \dots \\ \dots & k_{22} & \dots \\ \dots & \dots & \dots \end{bmatrix}$$



Ad Hoc Modifications to the Newton-Raphson Search

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

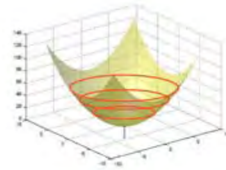
$$\mathbf{u}_{k+1} = \mathbf{u}_k - \left[\begin{bmatrix} \frac{\partial^2 J}{\partial u_1^2} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & \frac{\partial^2 J}{\partial u_m^2} \end{bmatrix}\right]^{-1} \left[\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_k}\right]^T$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \left[\left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}_k} + \mathbf{K}\right]^{-1} \left[\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_k}\right]^T, \quad \mathbf{K} > 0, \text{ diagonal}$$

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

⬅ With varying ε, this is the Levenberg-Marquardt algorithm

Conjugate Gradient Algorithm



- First step

$$\mathbf{u}_1(t) = \mathbf{u}_k(t) - \mathbf{K}_0 \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_0^T$$

- Calculate ratio of integrated gradient magnitudes squared

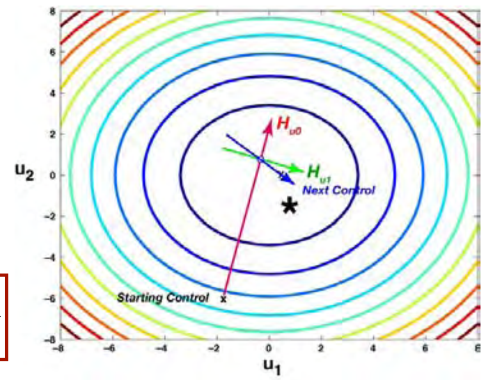
$$b = \frac{\int_{t_o}^{t_f} \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_1 \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_1^T dt}{\int_{t_o}^{t_f} \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_0 \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_0^T dt}$$

- Second step

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_2(t) = \mathbf{u}_1(t) - \mathbf{K}_1 \left\{ \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_1^T - b \left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_0^T \right\}$$

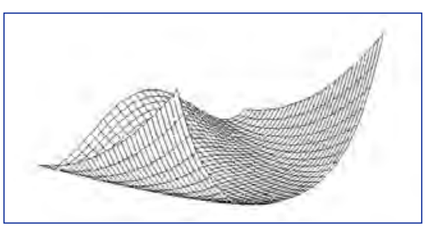
- Calculate gradient of improved trajectory

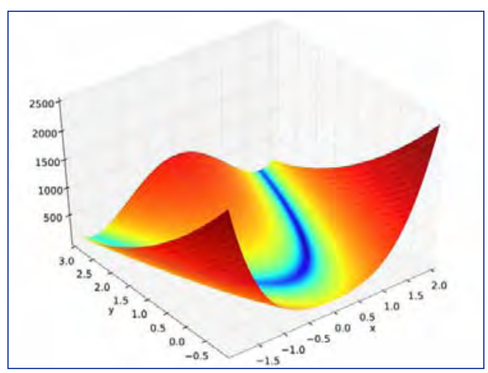
$$\left[\frac{\partial H}{\partial \mathbf{u}}(t) \right]_1^T$$



Rosenbrock Function

- Typical test function for numerical optimization algorithms





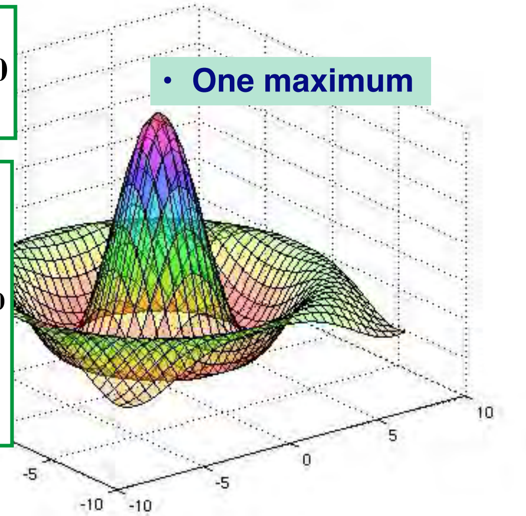
$$J(u_1, u_2) = (1 - u_1)^2 + 100\left(u_2 - u_1^2\right)^2$$

- Wolfram Alpha
  - **Plot (D ((1-u1)^2+100 (u2 - u1^2)^2, u1), D ((1-u1)^2+100 (u2 - u1^2)^2, u2))**
  - **Minimize[(1-u1)^2+100 (u2 - u1^2)^2, u1, u2]**

How Many Maxima/Minima does the "Mexican Hat" [$z = (\sin R)/R$] Have?

$$\left.\frac{\partial J}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^*} = \left.\begin{bmatrix} \frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2} & \cdots & \frac{\partial J}{\partial u_m} \end{bmatrix}\right|_{\mathbf{u}=\mathbf{u}^*} = \mathbf{0}$$

$$\left.\frac{\partial^2 J}{\partial \mathbf{u}^2}\right|_{\mathbf{u}=\mathbf{u}^*} = \left.\begin{bmatrix} \frac{\partial^2 J}{\partial u_1^2} & \frac{\partial^2 J}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 J}{\partial u_1 \partial u_m} \\ \frac{\partial^2 J}{\partial u_2 \partial u_1} & \frac{\partial^2 J}{\partial u_2^2} & \cdots & \frac{\partial^2 J}{\partial u_2 \partial u_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 J}{\partial u_m \partial u_1} & \frac{\partial^2 J}{\partial u_2 \partial u_m} & \cdots & \frac{\partial^2 J}{\partial u_m^2} \end{bmatrix}\right|_{\mathbf{u}=\mathbf{u}^*} \gtrless \mathbf{0}$$



- One maximum

- Wolfram Alpha
  - (sin(sqrt(u1^2 + u2^2))) / (sqrt(u1^2 + u2^2))
  - maximize(sin(sqrt(u1^2 + u2^2))) / (sqrt(u1^2 + u2^2))

Next Time: Principles for Optimal Control of Dynamic Systems

Reading:
OCE: Sections 3.1, 3.2, 3.4

Supplemental Material

Infinity Norm

- **∞ Norm**
  - **As *p* -> ∞**
  - **Norm is the value of the maximum component**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

$$\dim(\mathbf{x}) = n \times n$$

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

- **Also called**
  - **Supremum norm**
  - **Chebyshev norm**
  - **Uniform norm**

$$L^\infty\ norm = \|\mathbf{x}\|_\infty = \max\left\{|x_1|,|x_2|,\ldots,|x_n|\right\}$$