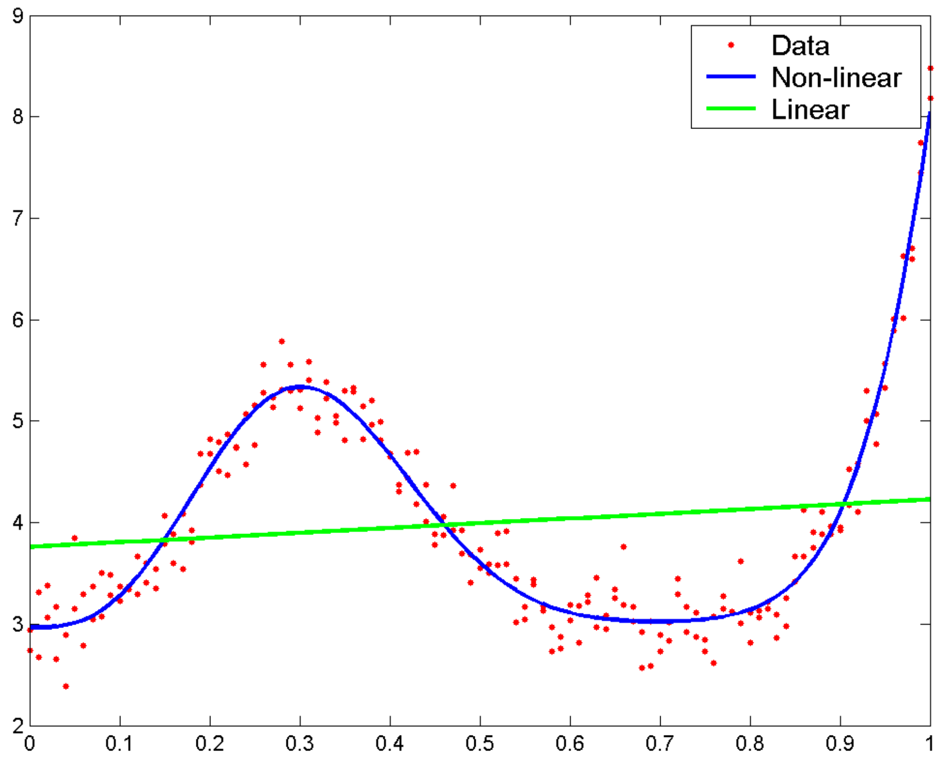
Introduction to Numerical Analysis for Engineers

Mathews

- Minimization Problems 5
- Least Square Approximation
  - Normal Equation
  - Parameter Estimation
  - Curve fitting
- Optimization Methods
  - Simulated Annealing
    - Traveling salesman problem
  - Genetic Algorithms

Minimization Problems

Data Modeling – Curve Fitting



Linear Model

$$y = cx$$

Non-linear Model

$$y = c(x)$$

Minimimize Overall Error

$$E(c) = \sum_i (y_i - c(x_i))^2$$

Objective: Find c that minimizes error

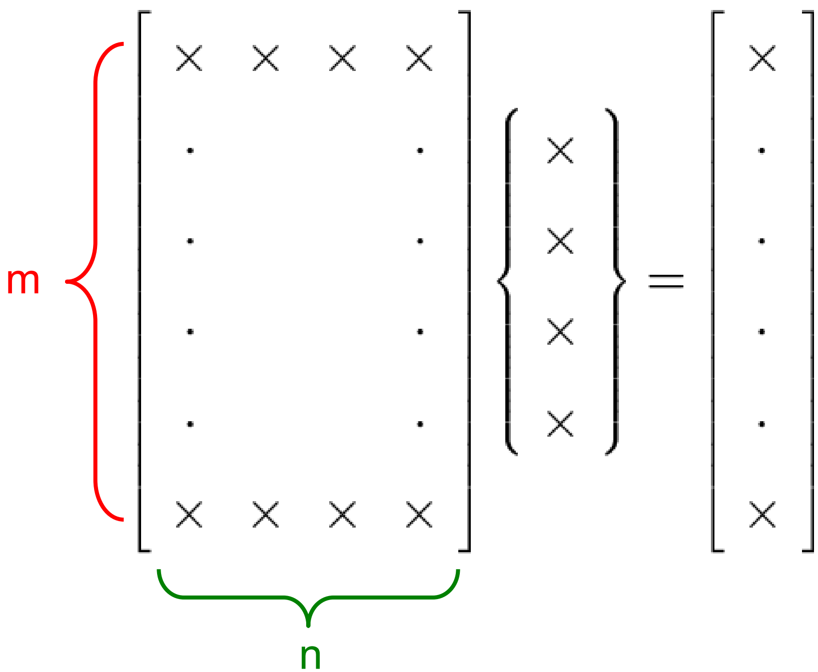
Least Square Approximation

Linear Measurement Model

$$\overline{\overline{\mathbf{A}}}\,\overline{\mathbf{x}} = \overline{\mathbf{b}}$$

n model parameters ($\overline{\mathbf{x}}$)

m measurements ($\overline{\mathbf{b}}$)



Overdetermined System

m measurements
n unknowns
m > n

Least Square Solution

Minimize Residual Norm

$$\overline{\mathbf{r}} = \overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\,\overline{\mathbf{x}}$$

$$||r||_2 = (\overline{\mathbf{r}}^T \overline{\mathbf{r}})^{1/2}$$

Least Square Approximation

Theorem

$$\text{If } \overline{\overline{\mathbf{A}}}^T\left(\overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{x}}\right) = 0 \Rightarrow \forall\, y \mid \left\|\overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{x}}\right\|_2 \leq \left\|\overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{y}}\right\|_2$$

Proof

$$\left.\begin{aligned} \overline{\mathbf{r}}_x &= \overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{x}} \\ \overline{\mathbf{r}}_y &= \overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{y}} \end{aligned}\right\} \Rightarrow$$

$$\overline{\mathbf{r}}_y = (\overline{\mathbf{b}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{x}}) + (\overline{\overline{\mathbf{A}}}\overline{\mathbf{x}} - \overline{\overline{\mathbf{A}}}\overline{\mathbf{y}}) = \overline{\mathbf{r}}_x + \overline{\overline{\mathbf{A}}}(\overline{\mathbf{x}} - \overline{\mathbf{y}})$$

$$\overline{\mathbf{r}}_y^T\overline{\mathbf{r}}_y = \overline{\mathbf{r}}_x^T\overline{\mathbf{r}}_x + (\overline{\mathbf{x}} - \overline{\mathbf{y}})^T\overline{\overline{\mathbf{A}}}^T\overline{\overline{\mathbf{A}}}(\overline{\mathbf{x}} - \overline{\mathbf{y}}) + (\overline{\mathbf{x}} - \overline{\mathbf{y}})^T\overline{\overline{\mathbf{A}}}^T\overline{\mathbf{r}}_x + r_x^T\overline{\overline{\mathbf{A}}}(\overline{\mathbf{x}} - \overline{\mathbf{y}})$$

$$\begin{aligned} \overline{\overline{\mathbf{A}}}^T\overline{\mathbf{r}}_x &= \overline{\mathbf{0}} \\ \overline{\mathbf{r}}_x^T\overline{\overline{\mathbf{A}}} &= \left(\overline{\overline{\mathbf{A}}}^T\overline{\mathbf{r}}_x\right)^T = \overline{\mathbf{0}} \end{aligned}$$

$$\begin{aligned} \overline{\mathbf{r}}_y^T\overline{\mathbf{r}}_y &= \overline{\mathbf{r}}_x^T\overline{\mathbf{r}}_x + (\overline{\mathbf{x}} - \overline{\mathbf{y}})^T\overline{\overline{\mathbf{A}}}^T\overline{\overline{\mathbf{A}}}(\overline{\mathbf{x}} - \overline{\mathbf{y}}) \\ &\Rightarrow \\ \|\overline{\mathbf{r}}_y\|_2^2 &= \|\overline{\mathbf{r}}_x\|_2^2 + \left\|\overline{\overline{\mathbf{A}}}(\overline{\mathbf{x}} - \overline{\mathbf{y}})\right\|_2^2 \geq \|\overline{\mathbf{r}}_x\|_2^2 \end{aligned}$$

q.e.d

Normal Equation

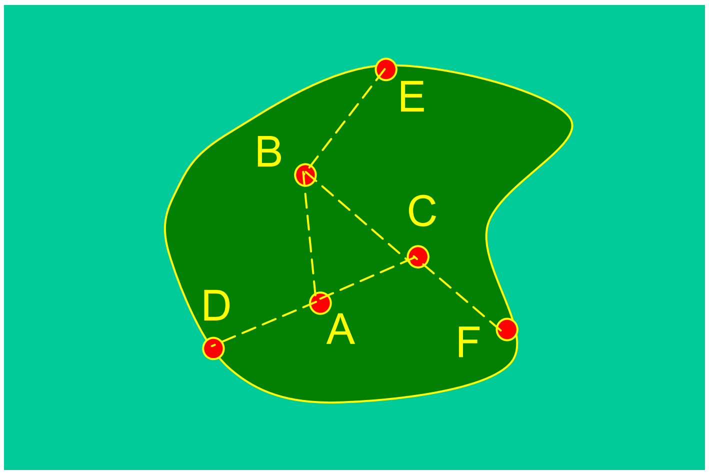
$$\left(\overline{\overline{\mathbf{A}}}^T\overline{\overline{\mathbf{A}}}\right)\overline{\mathbf{x}} = \overline{\overline{\mathbf{A}}}^T\overline{\mathbf{b}}$$

$$\overline{\overline{\mathbf{C}}} = \overline{\overline{\mathbf{A}}}^T\overline{\overline{\mathbf{A}}}$$

Symmetric n x n matrix. Non-singular if columns of A are linearly independent

Least Square Approximation
Parameter estimation

Example
Island Survey



Points D, E, and F at sea level. Find altitude of inland points A, B, and C.

```
A=[ [1 0 0 -1 0 -1]' [0 1 0 1 -1 0]' [0 0 1 0 1 1]']
b=[1 2 3 1 2 1]';
C=A'*A
c=A'*b
% Least square solution
z=inv(C)*c
% Residual
r=b-A*z
rn=sqrt(r'*r)
```

lstsq.m

Measured Altitude Differences

$$h_{DA} = 1\,,\; h_{EB} = 2\,,\; h_{FC} = 3\,,\; h_{AB} = 1\,,\; h_{BC} = 2\,,\; h_{AC} = 1$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{Bmatrix} z_A \\ z_B \\ z_C \end{Bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Normal Equation

$$\begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} \begin{Bmatrix} z_A \\ z_B \\ z_C \end{Bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 6 \end{bmatrix} \Rightarrow \begin{cases} z_A = \frac{5}{4} \\ z_B = \frac{7}{4} \\ z_C = 3 \end{cases}$$

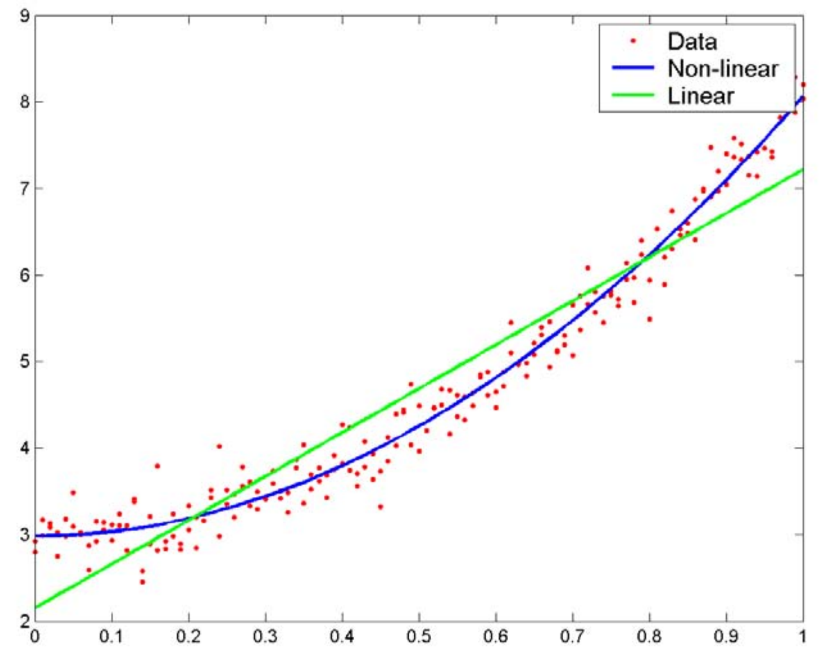
Residual Vector

$$\bar{\mathbf{r}} = \frac{1}{4}[-1, 1, 0, 2, 3, -3]^T$$

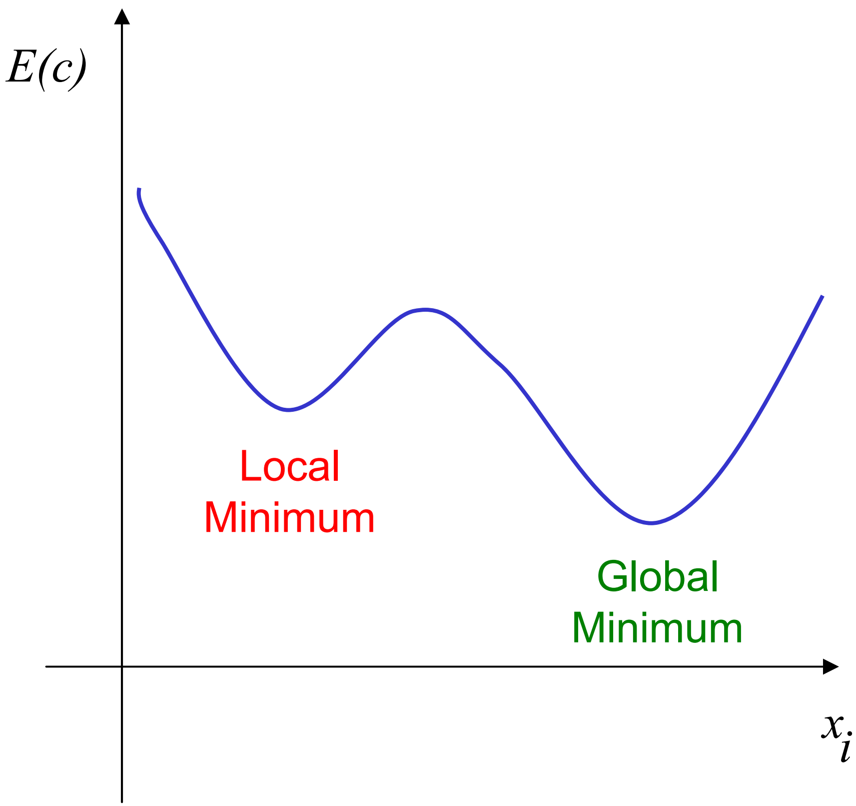
Least Square Approximation Curve Fitting

curve.m

```
% Quadratic data model
fxy='a*x.^2+b'
f=inline(fxy,'x','a','b');
x=[0:0.01:1]; x=reshape([x' x']',1,2*length(x));
n=length(x); y=zeros(n,1);
a=5;  b=3;
% Generate noisy data
amp=0.05*(max(f(x,a,b))-min(f(x,a,b)));
for i=1:n
    y(i) =f(x(i),a,b)+random('norm',0,amp);
end
figure(1); clf; hold off; p=plot(x,y,'.r');
set(p,'MarkerSize',10)
% Non-linear, quadrati model
A=ones(n,2); A(:,1)=f(x,1,0)'; bb=y;
%Normal matrix
C=A'*A; c=A'*bb;
z=inv(C)*c
% Residuals
r=bb-A*z; rn=sqrt(r'*r)/n
hold on; p=plot(x,f(x,z(1),z(2)),'b'); set(p,'LineWidth',2)
% Linear model
A(:,1)=x';
C=A'*A; c=A'*bb;
z=inv(C)*c
% Residuals
r=bb-A*z; rn=sqrt(r'*r)/n
hold on; p=plot(x,z(1)*x+z(2),'g'); set(p,'LineWidth',2)
p=legend('Data','Non-linear','Linear'); set(p,'FontSize',14);
```



Optimization Problems
Non-linear Models



Non-linear models

$$y = c(x)$$

Minimimize Overall Error

$$E(c) = \sum_i (y_i - c(x_i))^2$$

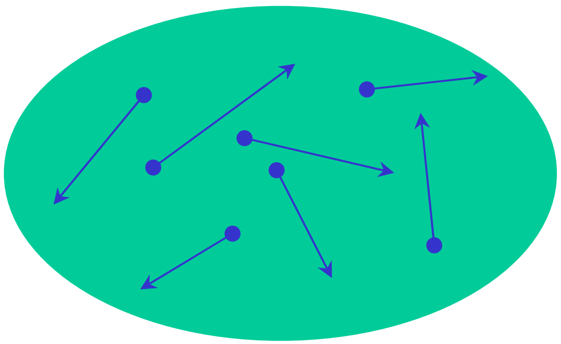
Measured values (y_i) — Model Parameters (c)

Non-linear models often have multiple, local minima. A locally linear, least square approximation may therefore find a local minimum instead of the global minimum.

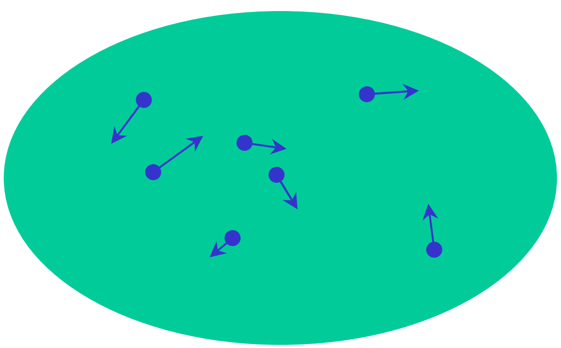
Optimization Algorithms
Simulated Annealing

Analogy: Freezing of a Liquid

High temperature T



Low temperature T



Crystal: Minimum energy of system.
Slow cooling -> global minimum: crystal.
Fast cooling -> local minimum: glass.

Boltzman Probability Distribution

Energy probabilistically distributed among all states.
Higher energy states possible even at low temperature!

$$p(E) \sim exp(-E/kT)$$

Optimization Problem: Minimize residual 'energy'

$$E(x) = \bar{\mathbf{r}}^T(x)\bar{\mathbf{r}}(x) = ||\bar{\mathbf{r}}(x)||_2$$

Simulated thermodynamic system changes its energy from E_1 to E_2 with probability

$$p = exp(-(E_2 - E_1)/kT)$$

Lower energy always accepted

$$E_2 < E_1 : \qquad p > 1 \Rightarrow p = 1$$

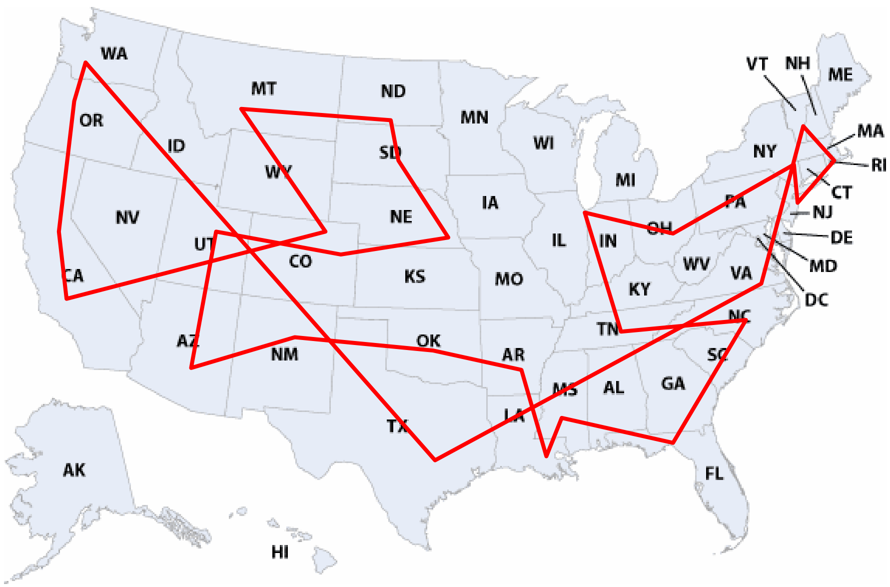
Higher energy accepted with probability p:
Allows escape from local minimum

$$E_2 > E_1 : \qquad p = exp(-(E_2 - E_1)/kT)$$

Elements of Metropolis algorithm

1. Description of possible system configurations
2. Random number generator for changing parameters
3. Cost function – 'energy' E
4. Control parameter – 'temperature' T.

Simulated Annealing
Example: Traveling Salesman Problem



Adapted by MIT OCW.

Objective:
Visit N cities across the US in arbitrary order, in the shortest time possible.

Metropolis Algorithm

1. Configuration: Cities I = 1,2, …N. Order can vary
2. Rearrangements: Change the order of any two cities.
3. Cost function: Distance traveled, number of Mississippi crossings.
4. Annealing schedule. Experimentation with ‘cooling’ schedule. T held constant for e.g. 100 re-orderings (heat-bath method).

Cost function: Distance Traveled

$$E = \sum_{i=1}^{N} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

Penalty for crossing Mississippi

East: $\mu_i = 1$

$$E = \sum_{i=1}^{N} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + \lambda(\mu_i - \mu_{i+1})^2}$$

West: $\mu_i = -1$

Simulated Annealing
Example: Traveling Salesman Problem

salesman.m

```
% Travelling salesman problem
% Create random city distribution
n=20; x=random('unif',-1,1,n,1); y=random('unif',-1,1,n,1);
gam=1; mu=sign(x);
% End up where you start. Add starting point to end
x=[x' x(1)]'; y=[y' y(1)]'; mu=[mu' mu(1)]';
figure(1); hold off; g=plot(x,y,'.r'); set(g,'MarkerSize',20);
c0=cost(x,y,mu,gam); k=1; % Boltzman constant
nt=50; nr=200; % nt: temp steps. nr: city switches each T
cp=zeros(nr,nt);
iran=inline('round(random(d,1.5001,n+0.4999))','d','n');
for i=1:nt
    T=1.0 -(i-1)/nt
    for j=1:nr
        % switch two random cities
        ic1=iran('unif',n); ic2=iran('unif',n);
        xs=x(ic1); ys=y(ic1); ms=mu(ic1);
        x(ic1)=x(ic2); y(ic1)=y(ic2); mu(ic1)=mu(ic2);
        x(ic2)=xs; y(ic2)=ys; mu(ic2)=ms;
        p=random('unif',0,1); c=cost(x,y,mu,gam);
        if (c < c0 | p < exp(-(c-c0)/(k*T))) % accept
            c0=c;
        else                              % reject and switch back
            xs=x(ic1); ys=y(ic1); ms=mu(ic1);
            x(ic1)=x(ic2); y(ic1)=y(ic2); mu(ic1)=mu(ic2);
            x(ic2)=xs; y(ic2)=ys; mu(ic2)=ms;
        end
        cp(j,i)=c0;
    end
    figure(2); plot(reshape(cp,nt*nr,1)); drawnow;
    figure(1); hold off; g=plot(x,y,'.r'); set(g,'MarkerSize',20);
    hold on; plot(x,y,'b');
    g=plot(x(1),y(1),'.g'); set(g,'MarkerSize',30);
    p=plot([0 0],[-1 1],'r--'); set(g,'LineWidth',2); drawnow;
end
```

```
function [c] = cost(x,y,mu,gam)
n=length(x);
c=0;
for i=1:n-1
   c =c+sqrt((x(i+1)-x(i))^2
                 +(y(i+1)-y(i))^2)
        + gam*(mu(i+1)-mu(i));
end
```

cost.m