Lecture 14

Nonlinear Problems
Grid Search and Monte Carlo Methods

Syllabus

Lecture 01	Describing Inverse Problems
Lecture 02	Probability and Measurement Error, Part 1
Lecture 03	Probability and Measurement Error, Part 2
Lecture 04	The L_2 Norm and Simple Least Squares
Lecture 05	A Priori Information and Weighted Least Squared
Lecture 06	Resolution and Generalized Inverses
Lecture 07	Backus-Gilbert Inverse and the Trade Off of Resolution and Variance
Lecture 08	The Principle of Maximum Likelihood
Lecture 09	Inexact Theories
Lecture 10	Nonuniqueness and Localized Averages
Lecture 11	Vector Spaces and Singular Value Decomposition
Lecture 12	Equality and Inequality Constraints
Lecture 13	L_1 , L_∞ Norm Problems and Linear Programming
Lecture 14	**Nonlinear Problems: Grid and Monte Carlo Searche**s
Lecture 15	Nonlinear Problems: Newton's Method
Lecture 16	Nonlinear Problems: Simulated Annealing and Bootstrap Confidence Intervals
Lecture 17	Factor Analysis
Lecture 18	Varimax Factors, Empircal Orthogonal Functions
Lecture 19	Backus-Gilbert Theory for Continuous Problems; Radon's Problem
Lecture 20	Linear Operators and Their Adjoints
Lecture 21	Fréchet Derivatives
Lecture 22	Exemplary Inverse Problems, incl. Filter Design
Lecture 23	Exemplary Inverse Problems, incl. Earthquake Location
Lecture 24	Exemplary Inverse Problems, incl. Vibrational Problems

Purpose of the Lecture

Discuss two important issues related to probability

Introduce linearizing transformations

Introduce the Grid Search Method

Introduce the Monte Carlo Method

Part 1

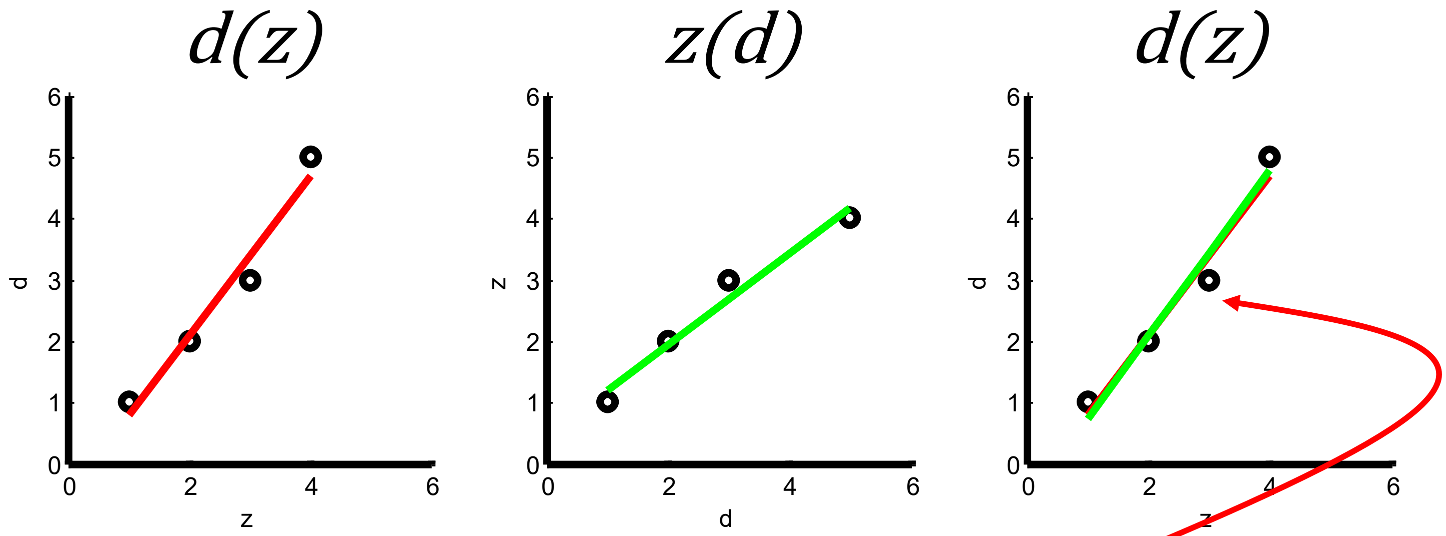
two issue related to probability

not limited to nonlinear problems
but
they tend to arise there a lot

issue #1

distribution of the data matters

$d(z)$ vs. $z(d)$



not quite the same

intercept -0.500000 slope 1.300000

intercept -0.615385 slope 1.346154

$d(z)$

d are Gaussian distributed

z are error free

$z(d)$

z are Gaussian distributed

d are error free

$d(z)$

d are Gaussian distributed

z are error free

not the same

$z(d)$

z are Gaussian distributed

d are error free

lesson learned

you must properly account for how the noise is distributed

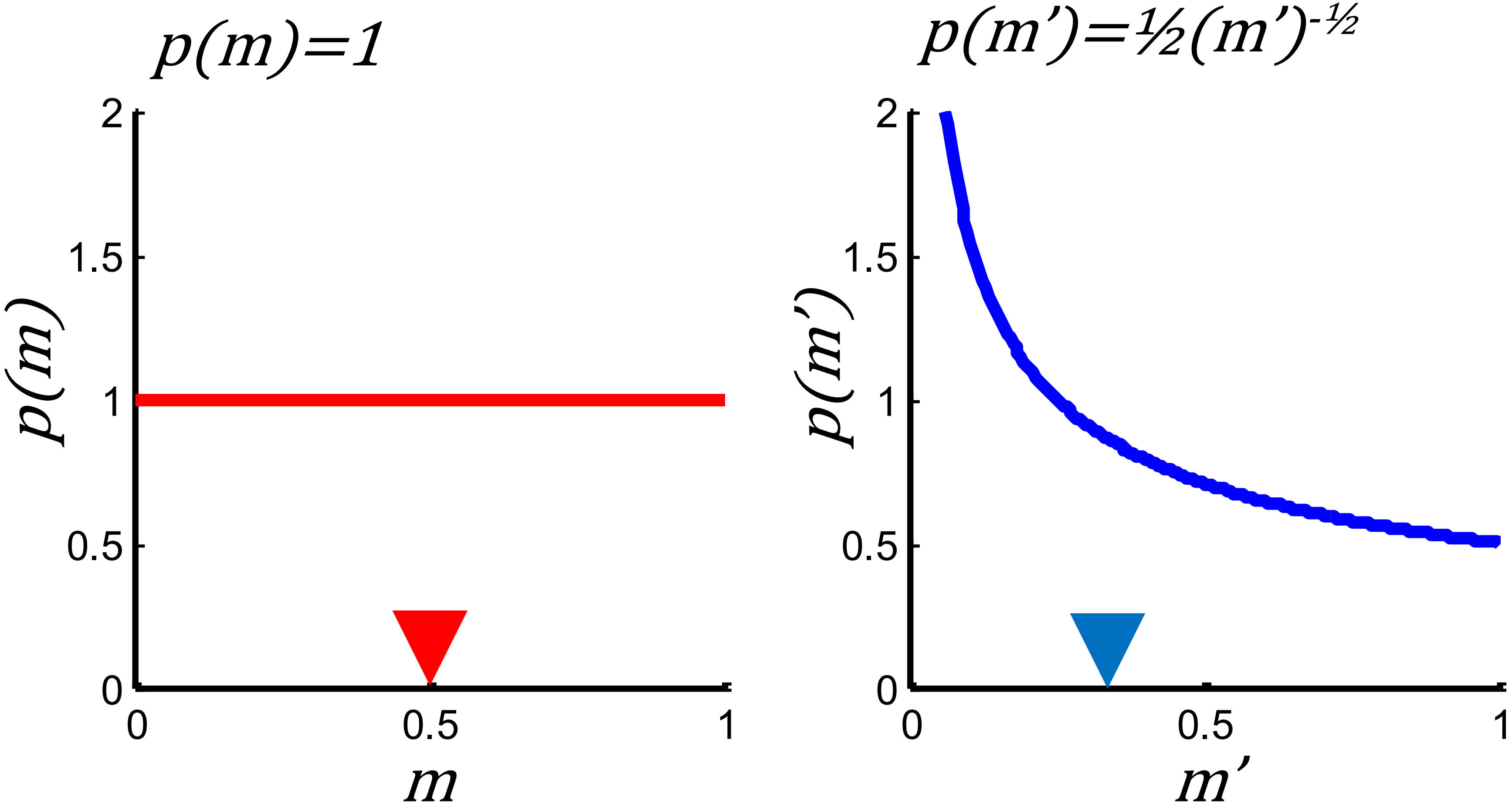
issue #2

mean and maximum likelihood point can change under reparameterization consider the non-linear transformation

$$m' = m^2$$

with

$p(m)$ uniform on (0,1)



p(m)=1
p(m')=½(m')^-½
p(m)
p(m')
2
1.5
1
0.5
0
0
0.5
1
m
m'

Calculation of Expectations

$$\langle m \rangle = \int_0^1 m\, p(m)\, \mathrm{d}m = \int_0^1 m\, \mathrm{d}m = 1/2$$

$$\langle m' \rangle = \int_0^1 m' p(m')\, \mathrm{d}m' = \int_0^1 ½\, m'^{½}\, \mathrm{d}m' = 1/3$$

$$(1/3) \neq (1/2)^2.$$

although

$$m' = m^2$$

$$\langle m' \rangle \neq \langle m \rangle^2$$

right way

$$\text{p.d.f for } \mathbf{m} \quad \rightarrow \quad \text{p.d.f for } \mathbf{m}' \quad \rightarrow \quad \text{estimate of } \mathbf{m}'$$

wrong way

$$\text{p.d.f for } \mathbf{m} \quad \rightarrow \quad \text{estimate of } \mathbf{m} \quad \rightarrow \quad \text{estimate of } \mathbf{m}'$$

Part 2

linearizing transformations

Non-Linear Inverse Problem

$$\mathbf{d} = \mathbf{g}(\mathbf{m})$$

transformation

$d \rightarrow d'$

$m \rightarrow m'$

Linear Inverse Problem

$$\mathbf{d}' = \mathbf{G}\mathbf{m}'$$

solve with least-squares

Non-Linear Inverse Problem

$$\mathbf{d} = \mathbf{g}(\mathbf{m})$$

sformation

rarely possible, of course

Linear Inverse Problem

$$\mathbf{d}' = \mathbf{G}\mathbf{m}'$$

solve with least-squares

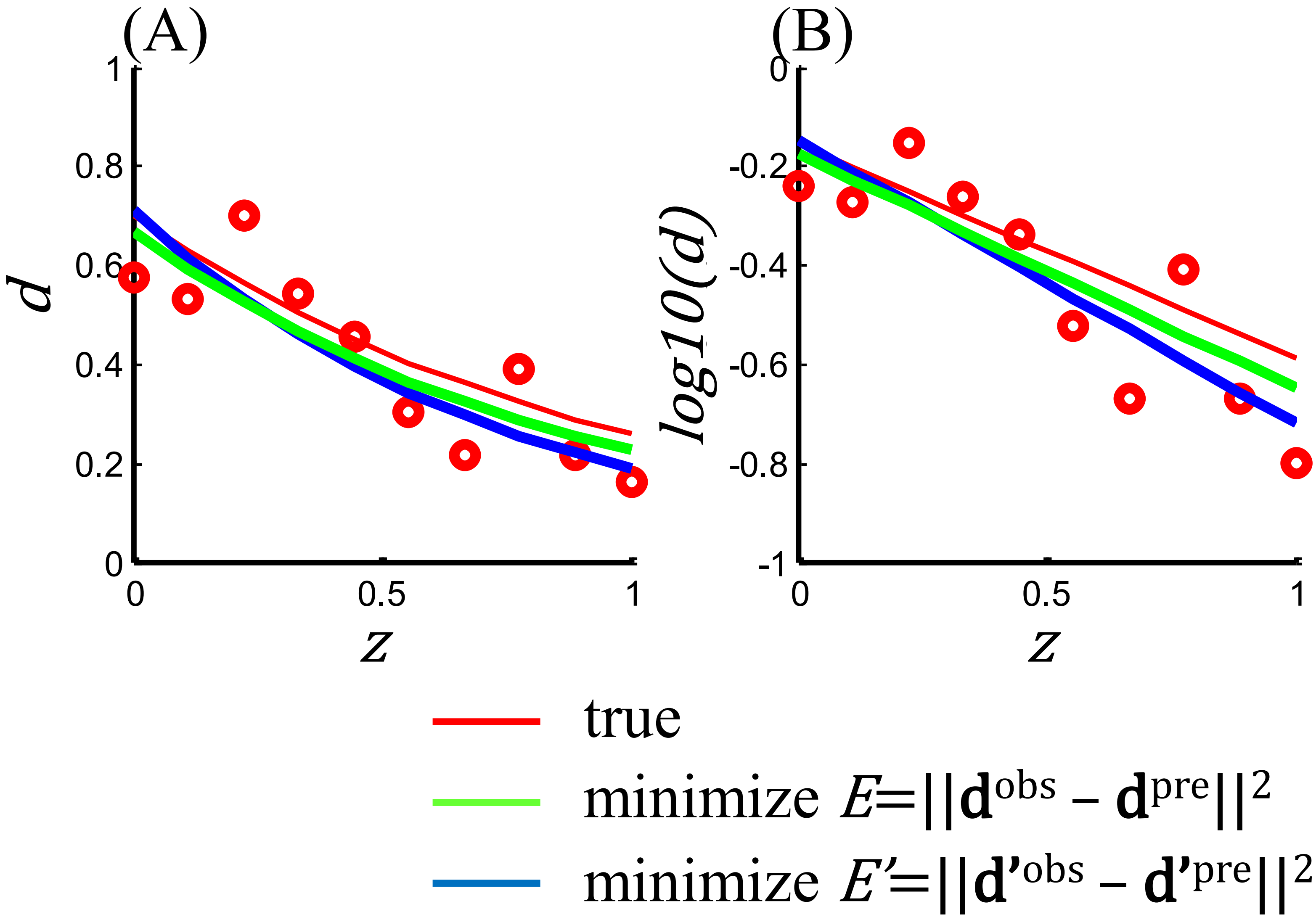
an example

$$d_i = m_1 \exp(m_2 z_i)$$

$$d'_i = \log(d_i)$$
$$m'_1 = \log(m_1)$$
$$m'_2 = m_2$$

$$\log(d_i) = \log(m_1) + m_2 z_i$$

$$d_i' = m'_1 + m'_2 z_i$$



(A)
(B)
d
log10(d)
z
z
0
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.5
true
minimize $E = ||\mathbf{d}^{obs} - \mathbf{d}^{pre}||^2$
minimize $E' = ||\mathbf{d}'^{obs} - \mathbf{d}'^{pre}||^2$

again
measurement error is being treated inconsistently

if **d** is Gaussian-distributed

then **d'** is not

so why are we using least-squares?

we should really use a technique appropriate for the new error ...

... but then a linearizing transformation is not really much of a simplification

non-uniqueness

consider

$$d_i = m_1^2 + m_1 m_2 z_i$$

consider

$$d_i = m_1^2 + m_1 m_2 z_i$$

linearizing transformation

$m'_1 = m_1^2$ and $m'_2 = m_1 m_2$

$$d_i = m'_1 + m'_2 z_i$$

consider

$$d_i = m_1^2 + m_1 m_2 z_i$$

linearizing transformation

$$m'_1 = m_1^2 \text{ and } m'_2 = m_1 m_2$$

$$d_i = m'_1 + m'_2 z_i$$

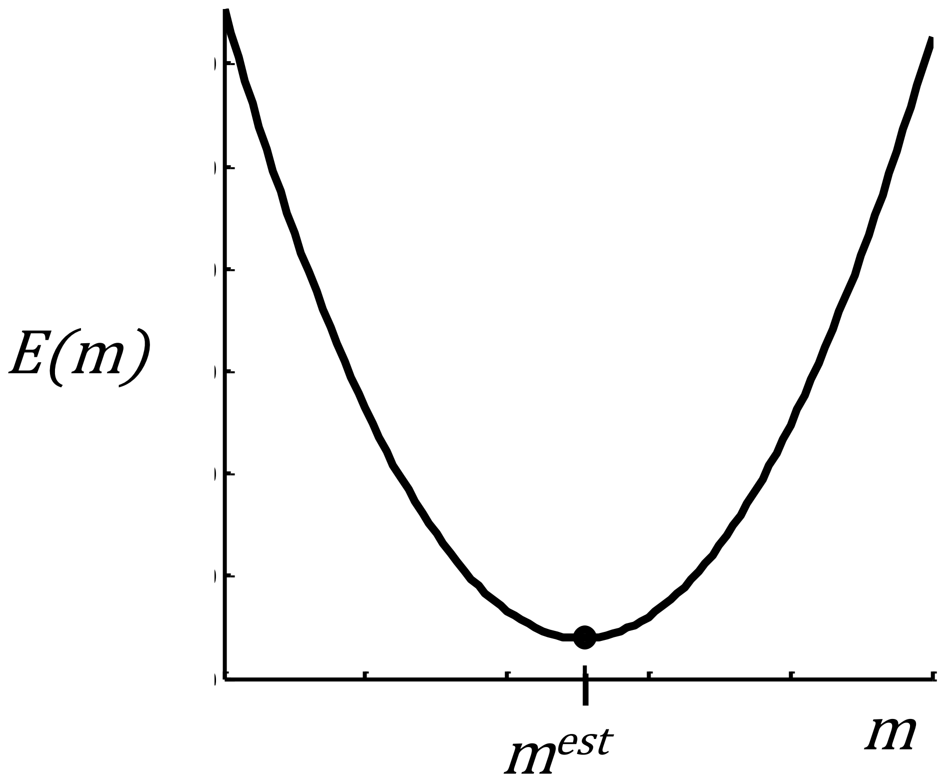
but actually the problem is nonunique
if **m** is a solution, so is –**m**
a fact that can easily be overlooked when focusing on
the transformed problem

linear Gaussian problems have well-understood non-uniqueness

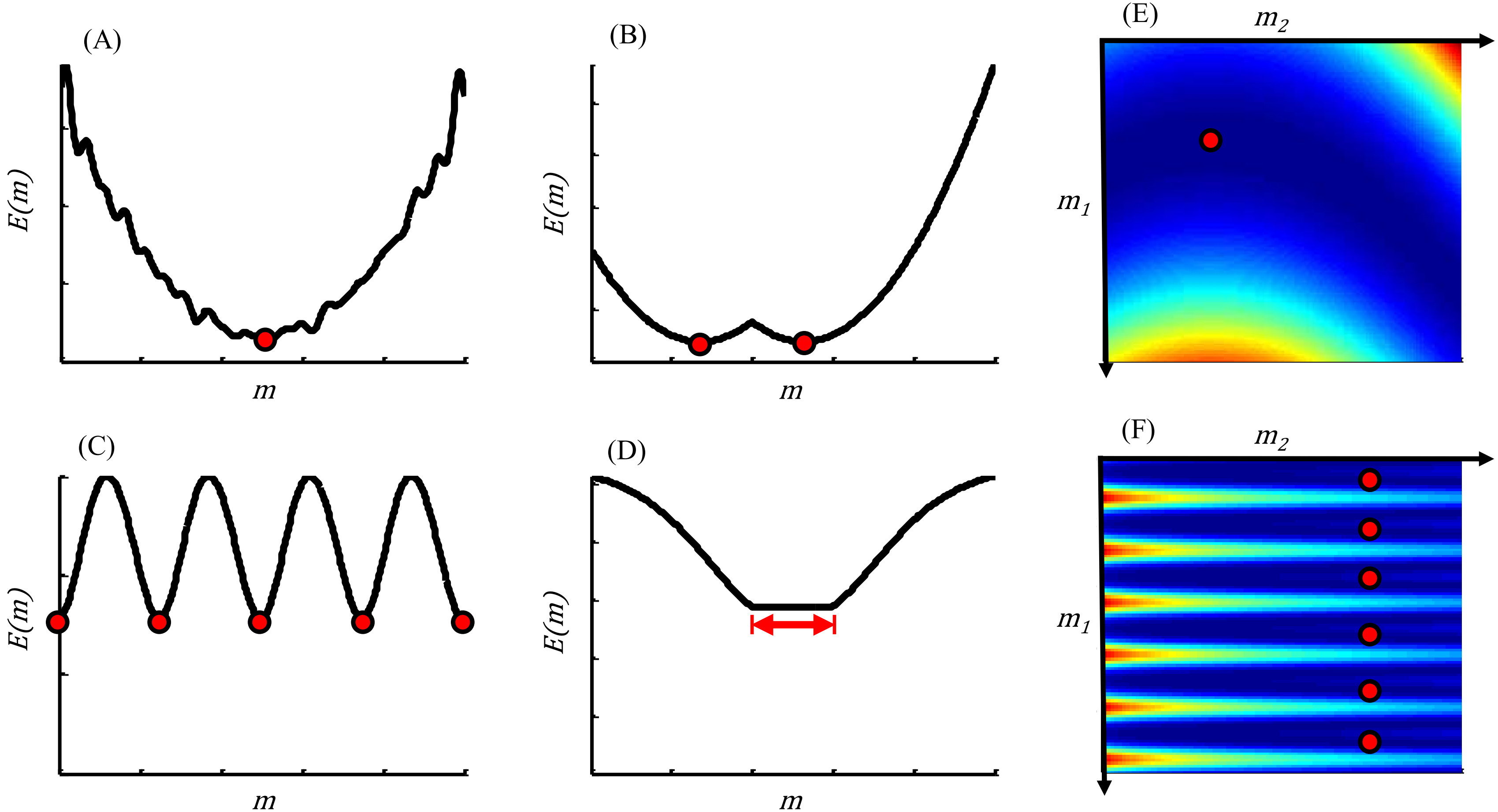
The error $E(\mathbf{m})$ is always a multi-dimensioanl quadratic

But $E(\mathbf{m})$ can be constant in some directions in model space (the null space). Then the problem is non-unique.

If non-unique, there are an infinite number of solutions, each with a different combination of null vectors.


E(m)
m
m^est

a nonlinear Gaussian problems can be non-unique in a variety of ways


(A)
E(m)
m
(B)
E(m)
m
(C)
E(m)
m
(D)
E(m)
m
(E)
m2
m1
(F)
m2
m1

Part 3

the grid search method

sample inverse problem

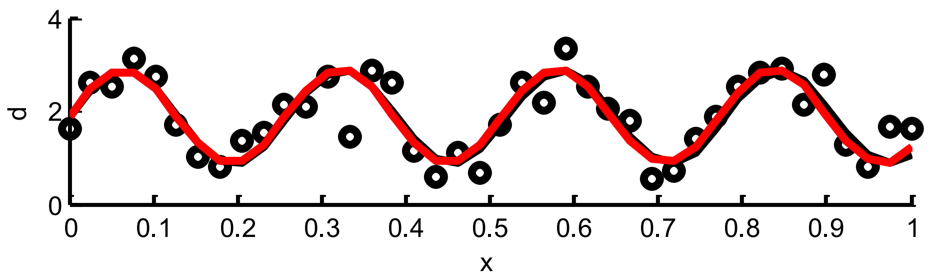
$$d_i(x_i) = sin(\omega_0 m_1 x_i) + m_1 m_2$$

with $\omega_0 = 20$

true solution

$$m_1 = 1.21,\ m_2 = 1.54$$

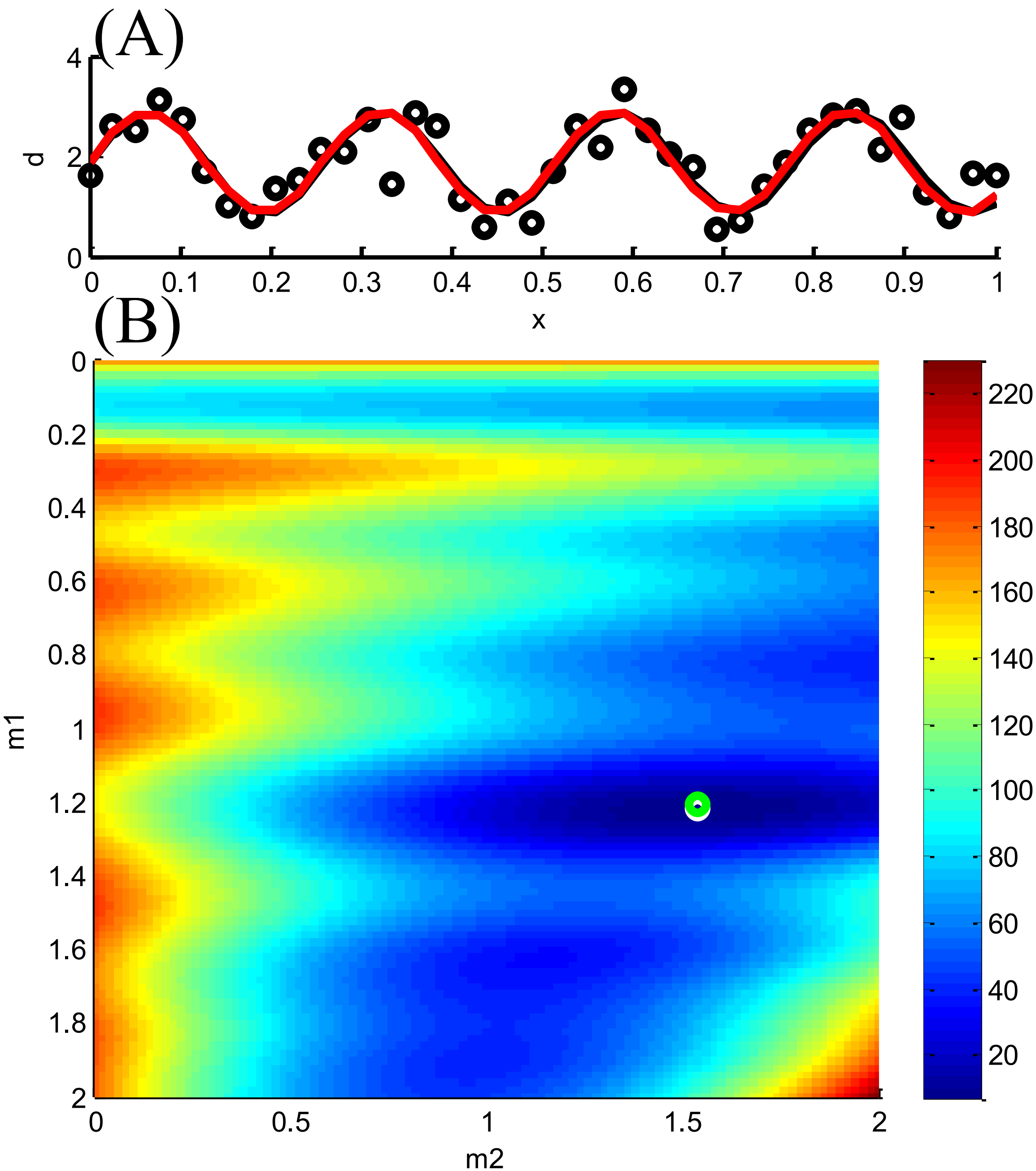
N=40 noisy data



strategy

compute the error on a multi-dimensional grid in model space

choose the grid point with the smallest error as the estimate of the solution



(A)
d
x
0
2
4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(B)
m1
m2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
0.5
1
1.5
2
220
200
180
160
140
120
100
80
60
40
20

to be effective

The total number of model parameters are small, say $M<7$. The grid is M-dimensional, so the number of trial solution is proportional to L^M, where L is the number of trial solutions along each dimension of the grid.

The solution is known to lie within a specific range of values, which can be used to define the limits of the grid.

The forward problem $\mathbf{d}=\mathbf{g}(\mathbf{m})$ can be computed rapidly enough that the time needed to compute L^M of them is not prohibitive.

The error function $E(\mathbf{m})$ is smooth over the scale of the grid spacing, Δm, so that the minimum is not missed through the grid spacing being too coarse.

MatLab

```
% 2D grid of m’s
L = 101;
Dm = 0.02;
m1min=0;
m2min=0;
m1a = m1min+Dm*[0:L-1]';
m2a = m2min+Dm*[0:L-1]';
m1max = m1a(L);
m2max = m2a(L);
```

```
% grid search, compute error, E
E = zeros(L,L);
for j = [1:L]
for k = [1:L]
   dpre=sin(w0*m1a(j)*x)+m1a(j)*m2a(k);
   E(j,k) = (dobs-dpre)'*(dobs-dpre);
end
end
```

```
% find the minimum value of E
[Erowmins, rowindices] = min(E);
[Emin, colindex] = min(Erowmins);
rowindex = rowindices(colindex);
m1est = m1min+Dm*(rowindex-1);
m2est = m2min+Dm*(colindex-1);
```

Definition of Error for non-Gaussian statistcis

Gaussian p.d.f.: $E = \sigma_d^{-2} ||e||_2^2$

but since

$$p(\mathbf{d}) \propto \exp(-½E)$$

and

L=log(p($\mathbf{d}$))=c-½E

$$E = 2(c - L) \rightarrow -2L$$

since constant does not affect location of minimum

in non-Gaussian cases:

define the error in terms of the likelihood L

$$E = -2L$$

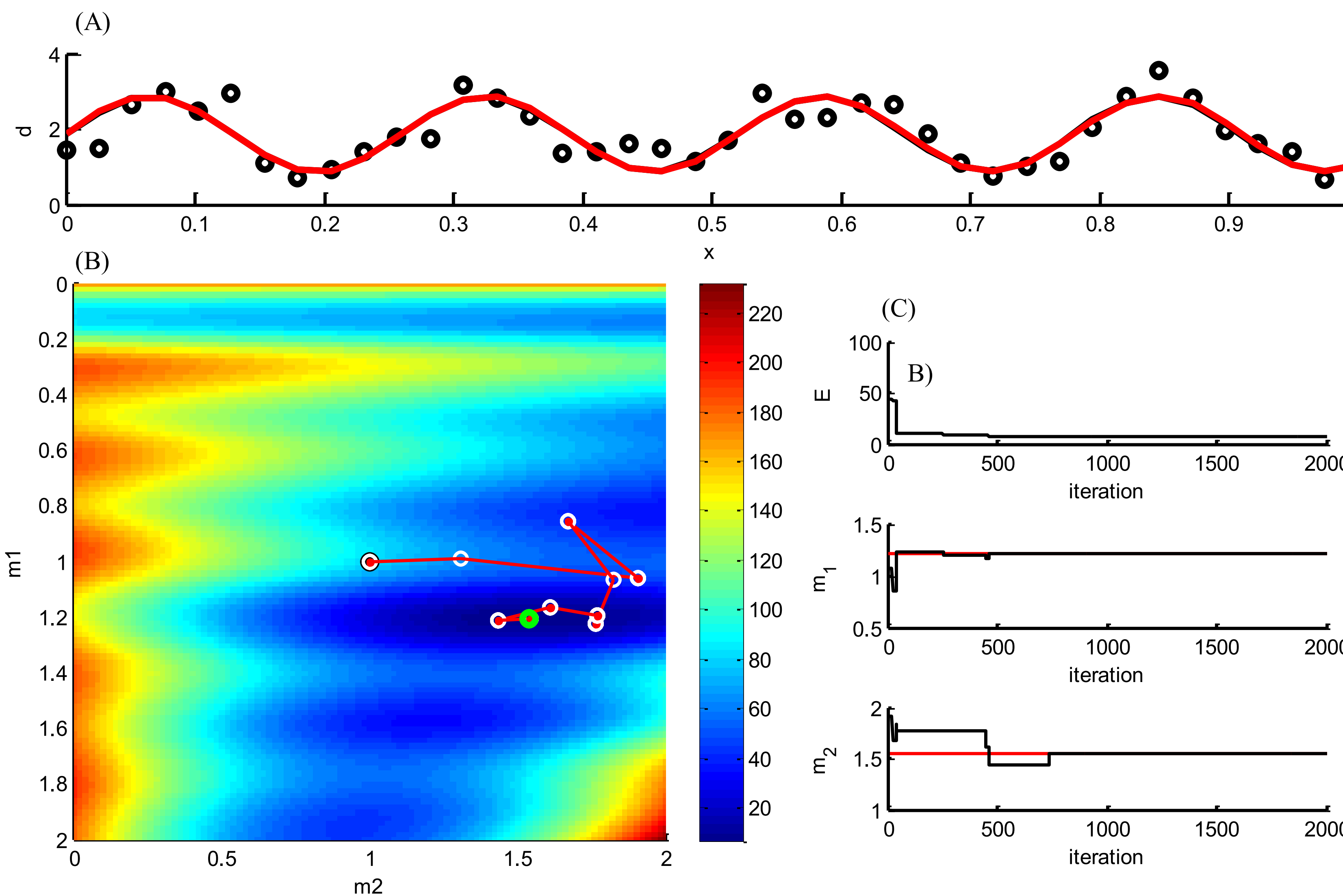
Part 4

the Monte Carlo method

strategy

compute the error at randomly generated points in model space

choose the point with the smallest error as the estimate of the solution



(A)
d
x
(B)
m1
m2
(C)
B)
E
m_1
m_2
iteration

advantages over a grid search

doesn't require a specific decision about grid

model space interrogated uniformly so
process can be stopped when acceptable
error is encountered
process is open ended, can be continued
as long as desired

disadvantages

might require more time to generate a point in model space

results different every time; subject to “bad luck”

MatLab

```
% initial guess and corresponding error
mg=[1,1]';
dg = sin(w0*mg(1)*x) + mg(1)*mg(2);
Eg = (dobs-dg)'*(dobs-dg);
```

```
ma = zeros(2,1);
for k = [1:Niter]
    % randomly generate a solution
    ma(1) = random('unif',m1min,m1max);
    ma(2) = random('unif',m2min,m2max);
    % compute its error
    da = sin(w0*ma(1)*x) + ma(1)*ma(2);
    Ea = (dobs-da)'*(dobs-da);
    % adopt it if its better
    if( Ea < Eg )
        mg=ma;
        Eg=Ea;
    end
end
```