Optimal Estimation Methods

(Lecture 4 – Sequential Estimation and Nonlinear Least Squares)

Dr. John L. Crassidis

University at Buffalo – State University of New York
Department of Mechanical & Aerospace Engineering
Amherst, NY 14260-4400
johnc@buffalo.edu
http://www.buffalo.edu/~johnc

University at Buffalo The State University of New York

Diagonalization of Weight (i)

- For reasons seen later we want to show that the weight W can always be diagonalized in weighted least squares
  - Take eigenvalue/eigenvector decomposition (remember W is a symmetric matrix)

eigenvector matrix

diag(eigenvalues)

later constrained to be + definite

$$W = V \, \Lambda \, V^T$$

  - Substitute this into the weighted least squares loss function

$$J = \frac{1}{2}(\tilde{\mathbf{y}} - H\,\hat{\mathbf{x}})^T V \, \Lambda \, V^T (\tilde{\mathbf{y}} - H\,\hat{\mathbf{x}})$$

  - Define new variables

$$\tilde{\mathbf{z}} \equiv V^T \tilde{\mathbf{y}} \quad \text{and} \quad \mathcal{H} \equiv V^T H$$

  - Loss function becomes

$$J = \frac{1}{2}(\tilde{\mathbf{z}} - \mathcal{H}\,\hat{\mathbf{x}})^T \Lambda (\tilde{\mathbf{z}} - \mathcal{H}\,\hat{\mathbf{x}})$$

Diagonalization of Weight (ii)

- Note that the weight is now diagonal
- Minimizing the loss function leads to

$$\hat{\mathbf{x}} = (\mathcal{H}^T \Lambda \mathcal{H})^{-1} \mathcal{H}^T \Lambda \, \tilde{\mathbf{z}}$$

- Solution is identical to original weighted least squares solution
  - Substitute new variables to prove this

$$\begin{aligned}\hat{\mathbf{x}} &= (\mathcal{H}^T \Lambda \mathcal{H})^{-1} \mathcal{H}^T \Lambda \, \tilde{\mathbf{z}} \\ &= (H^T V \, \Lambda \, V^T H)^{-1} H^T V \, \Lambda V^T \tilde{\mathbf{y}}\end{aligned}$$

  - Using $W = V \Lambda V^T$ now gives

$$\hat{\mathbf{x}} = (H^T W H)^{-1} H^T W \, \tilde{\mathbf{y}}$$

- So in general we can assume that W is always diagonal without loss in generality

Sequential Estimation (i)

- In standard least squares an implicit assumption is present, namely, that all measurements are available for simultaneous ("batch") processing
  - In numerous real-world applications, the measurements may become available sequentially in subsets and, immediately upon receipt of a new data subset, it may be desirable to determine new estimates based upon all previous measurements (including the current subset)
  - Employ a sequential process instead of previous batch process in least squares solution
  - To simplify the initial discussion, consider two measurement sets

$$\tilde{\mathbf{y}}_1 = \begin{bmatrix} \tilde{y}_{11} & \tilde{y}_{12} & \cdots & \tilde{y}_{1m_1} \end{bmatrix}^T = \text{an } m_1 \times 1 \text{ vector of measurements}$$

$$\tilde{\mathbf{y}}_2 = \begin{bmatrix} \tilde{y}_{21} & \tilde{y}_{22} & \cdots & \tilde{y}_{2m_2} \end{bmatrix}^T = \text{an } m_2 \times 1 \text{ vector of measurements}$$

Sequential Estimation (ii)

- The associated observation equations are given by

$$\tilde{\mathbf{y}}_1 = H_1\mathbf{x} + \mathbf{v}_1$$
$$\tilde{\mathbf{y}}_2 = H_2\mathbf{x} + \mathbf{v}_2$$

where

$$\begin{aligned} H_1 &= \text{an } m_1 \times n \text{ known coefficient matrix of maximum rank } n \le m_1 \\ H_2 &= \text{an } m_2 \times n \text{ known coefficient matrix} \\ \mathbf{v}_1, \mathbf{v}_2 &= \text{vectors of measurement errors} \\ \mathbf{x} &= \text{the } n \times 1 \text{ vector of unknown parameters} \end{aligned}$$

- Least squares estimate from first measurement set

$$\hat{\mathbf{x}}_1 = (H_1^T W_1 H_1)^{-1} H_1^T W_1 \tilde{\mathbf{y}}_1$$

where W_1 is an $m_1 \times m_1$ symmetric, positive definite matrix

Sequential Estimation (iii)

- It is possible to consider both measurement subsets simultaneously and determine an estimate based upon both measurement subsets
- Consider the following merged form

$$\tilde{\mathbf{y}} = H\mathbf{x} + \mathbf{v}$$

where

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\mathbf{y}}_1 \\ \dots \\ \tilde{\mathbf{y}}_2 \end{bmatrix}, \quad H = \begin{bmatrix} H_1 \\ \dots \\ H_2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_2 \end{bmatrix}$$

- Assume a block diagonal structure for weighting matrix

$$W = \begin{bmatrix} W_1 & \vdots & 0 \\ \dots & & \dots \\ 0 & \vdots & W_2 \end{bmatrix}$$

- As shown before we can always assume it's diagonal if needed

Sequential Estimation (iv)

- Solution based on all measurements is denoted by $\hat{\mathbf{x}}_2$

$$\hat{\mathbf{x}}_2 = (H^T W H)^{-1} H^T W \tilde{\mathbf{y}}$$

(Based on all measurements)

- Note that this is equivalent to $\hat{\mathbf{x}}$ in standard squares
  - Does not use only the second measurement set (careful here)
- Goal is to find $\hat{\mathbf{x}}_2$ based on $\hat{\mathbf{x}}_1$ and $\tilde{\mathbf{y}}_2$ only
- Because of block structure in W, we have

$$\hat{\mathbf{x}}_2 = [H_1^T W_1 H_1 + H_2^T W_2 H_2]^{-1}(H_1^T W_1 \tilde{\mathbf{y}}_1 + H_2^T W_2 \tilde{\mathbf{y}}_2)$$

- Define the following

$$P_1 \equiv [H_1^T W_1 H_1]^{-1}$$

$$P_2 \equiv [H_1^T W_1 H_1 + H_2^T W_2 H_2]^{-1}$$

- Then we have (assuming inverses exist)

$$P_2^{-1} = P_1^{-1} + H_2^T W_2 H_2, \quad (1)$$

$$\hat{\mathbf{x}}_1 = P_1 H_1^T W_1 \tilde{\mathbf{y}}_1 \quad (2)$$

$$\hat{\mathbf{x}}_2 = P_2(H_1^T W_1 \tilde{\mathbf{y}}_1 + H_2^T W_2 \tilde{\mathbf{y}}_2) \quad (3)$$

Sequential Estimation (v)

- Pre-multiply Eq. (2) by the inverse of P_1

$$P_1^{-1}\hat{\mathbf{x}}_1 = H_1^T W_1 \tilde{\mathbf{y}}_1$$

- From Eq. (1) we have $P_1^{-1} = P_2^{-1} - H_2^T W_2 H_2$
- Substituting this into the previous equation gives

$$H_1^T W_1 \tilde{\mathbf{y}}_1 = P_2^{-1}\hat{\mathbf{x}}_1 - H_2^T W_2 H_2 \hat{\mathbf{x}}_1$$

- Substituting this expression into Eq. (3) yields

$$\hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_1 + K_2(\tilde{\mathbf{y}}_2 - H_2\hat{\mathbf{x}}_1)$$

Current measure

Previous estimate

where

$$K_2 \equiv P_2 H_2^T W_2$$

- Note that we have now achieved our goal of having the estimate at the second "time" be only a function of the previous "time" estimate and current "time" measurement

Sequential Estimation (vi)

- We now have a mechanism to sequentially provide an updated estimate based upon the previous estimate and associated side calculations
- We can easily generalize these equations to use the k^{th} estimate to determine the estimate at $k+1$ from the $k+1$ subset of measurements
- General form is given by

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + K_{k+1}(\tilde{\mathbf{y}}_{k+1} - H_{k+1}\hat{\mathbf{x}}_k)$$

where

$$K_{k+1} = P_{k+1}H_{k+1}^T W_{k+1}$$

"Kalman Gain Matrix"

$$P_{k+1}^{-1} = P_k^{-1} + H_{k+1}^T W_{k+1} H_{k+1}$$

- Update equation is known as Kalman update equation
  - We will see this update form again later!
- Specific form for $P^{-1}_{n\times n}$ is known as the information matrix recursion
  - Must invert an $n \times n$ matrix at each time step

Sequential Estimation (vii)

- The current approach for computing P_{k+1} involves computing its inverse, which offers no advantage over inverting the normal equations in their original batch processing
  - This is due to the fact that an $n \times n$ inverse must still be performed
  - Is there a more judicious form? The answer is yes
- Using the matrix inversion lemma

$$[A + B\,C\,D]^{-1} = A^{-1} - A^{-1}B\left(D\,A^{-1}B + C^{-1}\right)^{-1} D\,A^{-1}$$

with

$$A = P_k^{-1}, \quad B = H_{k+1}^T, \quad C = W_{k+1}, \quad D = H_{k+1}$$

leads to

$$P_{k+1} = P_k - P_k H_{k+1}^T \left(\overbrace{H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1}}^{\text{Same size of } \tilde{y}_{k+1}}\right)^{-1} H_{k+1} P_k \qquad (4)$$

- The matrix inverse is now the size of the dimension of the measurement subset, which is often much less than n
  - This usually significantly saves on the computations even though two inverses are required

Sequential Estimation (viii)

- The Kalman gain can also be rearranged in several alternate forms
- One of the more common is obtained by substituting Eq. (4) into the Kalman gain equation $K_{k+1} = P_{k+1} H_{k+1}^T W_{k+1}$ to give

$$\begin{aligned} K_{k+1} &= \left[P_k - P_k H_{k+1}^T \left(H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right)^{-1} H_{k+1} P_k \right] H_{k+1}^T W_{k+1} \\ &= P_k H_{k+1}^T \left[I - \left(H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right)^{-1} H_{k+1} P_k H_{k+1}^T \right] W_{k+1} \end{aligned}$$

- Now, factoring $\left(H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right)^{-1}$ outside of the square brackets leads directly to

$$K_{k+1} = P_k H_{k+1}^T \left(H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right)^{-1} \left[W_{k+1}^{-1} + \cancel{H_{k+1} P_k} H_{k+1}^T - \cancel{H_{k+1} P_k} H_{k+1}^T \right] W_{k+1}$$

- Leads simply to

$$K_{k+1} = P_k H_{k+1}^T \left[H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1} \right]^{-1}$$

Sequential Estimation (ix)

- Now have the covariance recursion form

$$\boxed{\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + K_{k+1}(\tilde{\mathbf{y}}_{k+1} - H_{k+1}\hat{\mathbf{x}}_k)}$$

$$\boxed{K_{k+1} = P_k H_{k+1}^T \left[H_{k+1} P_k H_{k+1}^T + W_{k+1}^{-1}\right]^{-1}}$$

$$\boxed{P_{k+1} = \left[I - K_{k+1} H_{k+1}\right] P_k}$$

- Requires inverse of the size of current measurement
  - Usually much smaller than n
- Must initialize both the estimate and P
  - Can use a small batch of n measurements, or can use the following

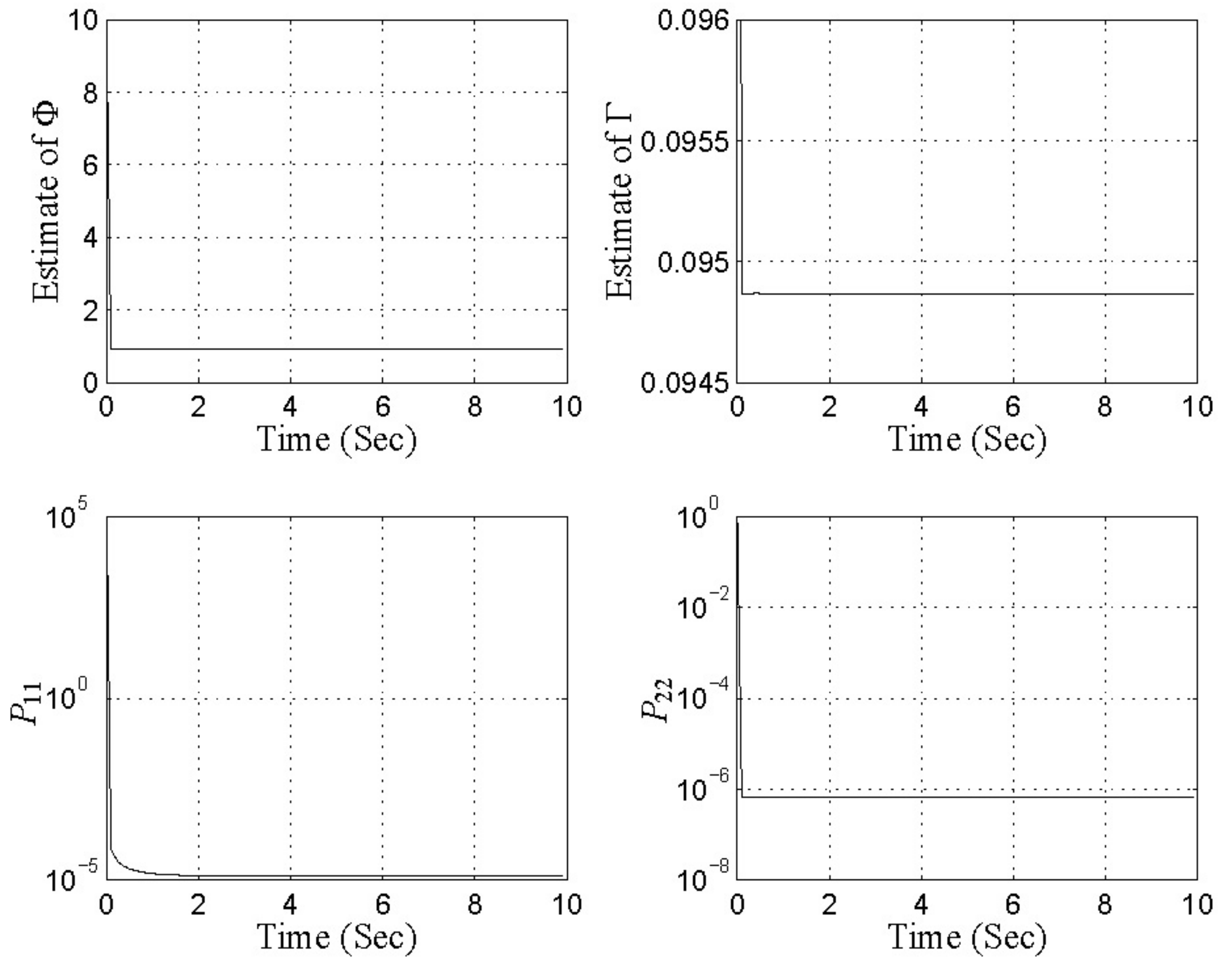
$$P_1 = \left[\frac{1}{\alpha^2} I + H_1^T W_1 H_1\right]^{-1}, \quad \hat{\mathbf{x}}_1 = P_1 \left[\frac{1}{\alpha}\boldsymbol{\beta} + H_1^T W_1 \tilde{\mathbf{y}}_1\right]$$

where α is a very "large" number and β is a vector of "small" numbers

- It can be shown that the estimate at the last point of the sequential process is equal to the batch solution

Sequential Example (i)

- Use previous state-space model example – Do Batch & Sequential. Then do same with polynomial example.



- Estimates converge to batch solution at last point

Sequential Example (ii)

```
% Get Truth and Measurements
dt=0.1;tf=10;
t=[0:dt:tf];
m=length(t);
a=-1;b=1;c=1;d=0;u=[100;zeros(m-1,1)];
[ad,bd]=c2d(a,b,dt);
y=dlsim(ad,bd,c,d,u);
ym=y+0.08*randn(m,1);

% Weight and H Matrix
w=inv(0.08^2);
h=[ym(1:m-1) u(1:m-1)];
```

Sequential Example (iii)

```
% Initial Conditions for Sequential Algorithm
alpha=1e3;
beta=[1e-2;1e-2];
p0=inv(1/alpha/alpha*eye(2)+h(1,:)'*w*h(1,:));
x0=p0*(1/alpha*beta+h(1,:)'*w*ym(2));

% Sequential Least Squares
xr=zeros(m-1,2);xr(1,:)=x0';
p=zeros(m-1,2);p(1,:)=diag(p0)';pp=p0;
for i=1:m-2;
 k=pp*h(i+1,:)'*inv(h(i+1,:)*pp*h(i+1,:)'+inv(w));
 pp=(eye(2)-k*h(i+1,:))*pp;
 xr(i+1,:)=xr(i,:)+(k*(ym(i+2)-h(i+1,:)*xr(i,:)'))'; % need y_{k+2} measurement
 p(i+1,:)=diag(pp)';
end
```

Nonlinear Least Squares (i) – BATCH ONLY, NO Sequential

- Many systems involve nonlinearities of the form

$$\tilde{\mathbf{y}} = \mathbf{f}(\mathbf{x}) + \mathbf{v}$$

- Estimate and residual are given by

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{f}(\hat{\mathbf{x}}) \\ \mathbf{e} &= \tilde{\mathbf{y}} - \hat{\mathbf{y}} \equiv \Delta\mathbf{y} \end{aligned} \quad , \qquad \tilde{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{e}$$

- Seek to find $\hat{\mathbf{x}}$ that minimizes

Linearize f(x)

$$J = \frac{1}{2}\mathbf{e}^T W \mathbf{e} = \frac{1}{2}[\tilde{\mathbf{y}} - \mathbf{f}(\hat{\mathbf{x}})]^T W [\tilde{\mathbf{y}} - \mathbf{f}(\hat{\mathbf{x}})]$$

- Simple solution only given in special cases
- Must use an iterative approach in general
  - Nonlinear least squares is such an approach
  - Others too, like gradient method

Nonlinear Least Squares (ii)

- Let's say that the estimate is given by some current value plus a correction

$$\hat{\mathbf{x}} = \mathbf{x}_c + \Delta\mathbf{x}$$

  - Say that the correction is small, and now we linearize the nonlinear function about the current value using a first-order Taylor series expansion

$$\mathbf{f}(\hat{\mathbf{x}}) \approx \mathbf{f}(\mathbf{x}_c) + H\Delta\mathbf{x}, \quad \underset{n \times n}{H} \equiv \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\mathbf{x}_c}$$

  - Measurement residual "after the correction" can be linearly approximated now

$$\Delta\mathbf{y} \equiv \tilde{\mathbf{y}} - \mathbf{f}(\hat{\mathbf{x}}) \approx \tilde{\mathbf{y}} - \mathbf{f}(\mathbf{x}_c) - H\Delta\mathbf{x} = \Delta\mathbf{y}_c - H\Delta\mathbf{x}$$

    where the residual "before the correction" is

$$\Delta\mathbf{y}_c \equiv \tilde{\mathbf{y}} - \mathbf{f}(\mathbf{x}_c)$$

Nonlinear Least Squares (iii)

- Now suppose we wish to determine the correction term, which can be done using linear least squares now
- We now minimize

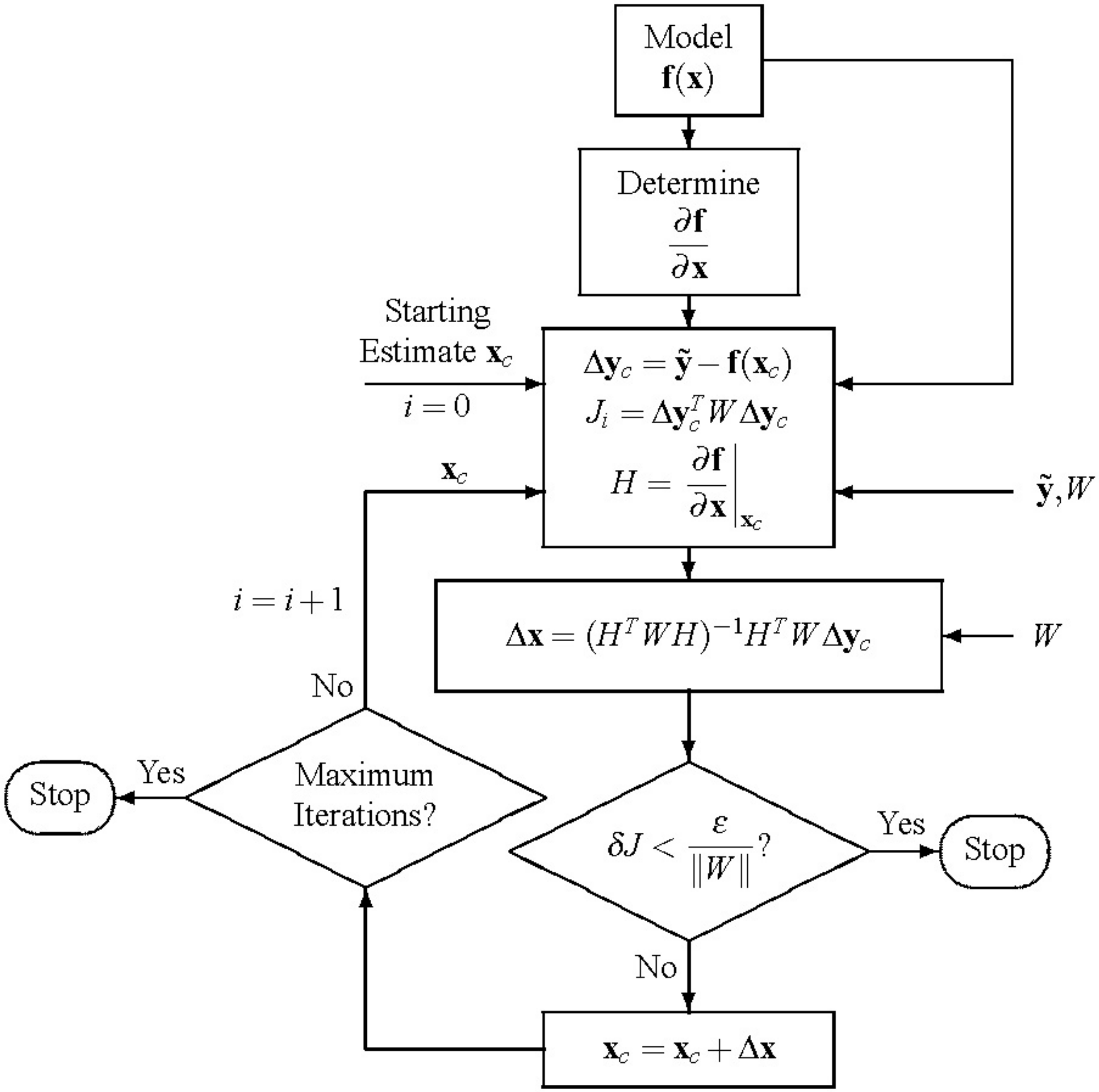
$$J = \frac{1}{2}\Delta\mathbf{y}^T W \Delta\mathbf{y} \equiv \frac{1}{2}(\Delta\mathbf{y}_c - H\Delta\mathbf{x})^T W (\Delta\mathbf{y}_c - H\Delta\mathbf{x})$$

- Solution is given by

$$\boxed{\Delta\mathbf{x} = (H^T W H)^{-1} H^T W \Delta\mathbf{y}_c}$$

- Notes
  - Same conditions apply for matrix inverse to exist
  - Solution converges rapidly if initial guess is good
  - May diverge quickly if initial guess is not good
  - Gradient solution has “opposite” effect
  - Levenberg-Marquardt algorithm combines good features of both
    - Start with LM and transition to NLS as solution converges

Nonlinear Least Squares (iv)



$$\delta J \equiv \frac{|J_i - J_{i-1}|}{J_i} < \frac{\varepsilon}{\|W\|}$$

NLS Example I

- Simple case

$$y = x^3 + 6x^2 + 11x + 6 = 0,$$

$$\mathbf{y} = y = 0$$
$$\mathbf{f}(\mathbf{x}) = f(x) = x^3 + 6x^2 + 11x + 6$$

- Correction becomes (Newton root solver)

$$x = x_c - \left[\left. \frac{\partial f}{\partial x} \right|_{x_c} \right]^{-1} f(x_c)$$

iteration	x	x	x
0	0.0000	-1.6000	-5.0000
1	-0.5455	-2.2462	-4.0769
2	-0.8490	-1.9635	-3.5006
3	-0.9747	-2.0001	-3.1742
4	-0.9991	-2.0000	-3.0324
5	-1.0000	-2.0000	-3.0015
6	-1.0000	-2.0000	-3.0000
7	-1.0000	-2.0000	-3.0000

Started iterations using three distinct initial guesses

Converges to the three roots of equation

NLS Example II (i)

- Consider previous discrete-time state model, but now wish to determine continuous-time parameters

$$y_{k+1} = \left[e^{a\Delta t}\right] y_k + \left[\frac{b}{a}(e^{a\Delta t} - 1)\right] u_k$$

- Must use nonlinear least squares approach now

$$\mathbf{x} = \begin{bmatrix} a & b \end{bmatrix}^T$$

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{y}_2 & \tilde{y}_3 & \cdots & \tilde{y}_{101} \end{bmatrix}^T$$

$$f_k = \left[e^{a\Delta t}\right] y_k + \left[\frac{b}{a}(e^{a\Delta t} - 1)\right] u_k$$

- Partials

$$\frac{\partial f_k}{\partial a} = \Delta t \left[e^{a\Delta t}\right] y_k + \left[\frac{b}{a^2}(1 - e^{a\Delta t}) + \frac{b}{a}\Delta t e^{a\Delta t}\right] u_k$$

$$\frac{\partial f_k}{\partial b} = \frac{1}{a}(e^{a\Delta t} - 1) u_k$$

NLS Example II (ii)

- The matrix H is given by

$$H = \begin{bmatrix} \Delta t\left[e^{a\Delta t}\right]\tilde{y}_1 + \left[\frac{b}{a^2}(1-e^{a\Delta t}) + \frac{b}{a}\Delta t e^{a\Delta t}\right]u_1 & \frac{1}{a}(e^{a\Delta t}-1)u_1 \\ \Delta t\left[e^{a\Delta t}\right]\tilde{y}_2 + \left[\frac{b}{a^2}(1-e^{a\Delta t}) + \frac{b}{a}\Delta t e^{a\Delta t}\right]u_2 & \frac{1}{a}(e^{a\Delta t}-1)u_2 \\ \vdots & \vdots \\ \Delta t\left[e^{a\Delta t}\right]\tilde{y}_{100} + \left[\frac{b}{a^2}(1-e^{a\Delta t}) + \frac{b}{a}\Delta t e^{a\Delta t}\right]u_{100} & \frac{1}{a}(e^{a\Delta t}-1)u_{100} \end{bmatrix}$$

- Iterations, starting at 5 and 5

iteration	$\hat{a}$	$\hat{b}$
0	5.0000	5.0000
1	0.4876	1.9540
2	−0.8954	1.0634
3	−1.0003	0.9988
4	−1.0009	0.9985
5	−1.0009	0.9985
6	−1.0009	0.9985

True values are given by -1 and 1, so good performance is given

NLS Example II (iii)

```
% Get Truth and Measurements
dt=0.1;tf=10;
t=[0:dt:tf];
m=length(t);
a=-1;b=1;c=1;d=0;u=[100;zeros(m-1,1)];
[ad,bd]=c2d(a,b,dt);
y=dlsim(ad,bd,c,d,u);
ym=y+0.08*randn(m,1);
w=inv(0.08^2);

% Initialize Variables
h=[ym(1:m-1) u(1:m-1)];
clear xe
xe(1,:)=[5 5];
xx=100000;
```

NLS Example II (iv)

```
% Nonlinear Least Squares
i=1;
while norm(xx) > 1e-8
if i > 50, break; end
aa=xe(i,1);bb=xe(i,2);
ea=exp(aa*dt);
h=[dt*ym(1:m-1)*ea+(bb/aa^2*(1-ea)+bb/aa*dt*ea)*u(1:m-1) 1/aa*(ea-1)*u(1:m-1)];
xx=inv(h'*w*h)*h'*w*(ym(2:m)-ym(1:m-1)*ea-bb/aa*(ea-1)*u(1:m-1));
xe(i+1,:)=xe(i,:)+xx';
i=i+1;
end

iteration_results=xe
disp(' ')
[phie,game]=c2d(xe(i,1),xe(i,2),dt)
```

NLS Example III (i)

- Under certain approximations, the pitch θ and yaw ψ attitude dynamics of an inertially and aerodynamically symmetric projectile can be modeled via

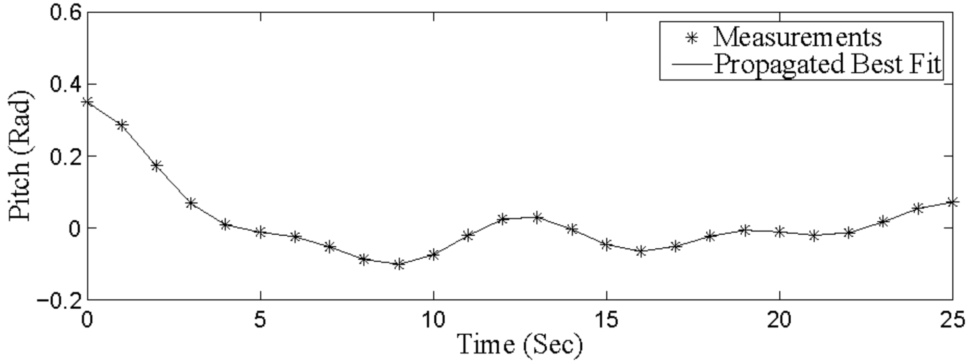
$$\theta(t) = k_1 e^{\lambda_1 t}\cos(\omega_1 t + \delta_1) + k_2 e^{\lambda_2 t}\cos(\omega_2 t + \delta_2) + k_3 e^{\lambda_3 t}\cos(\omega_3 t + \delta_3) + k_4$$

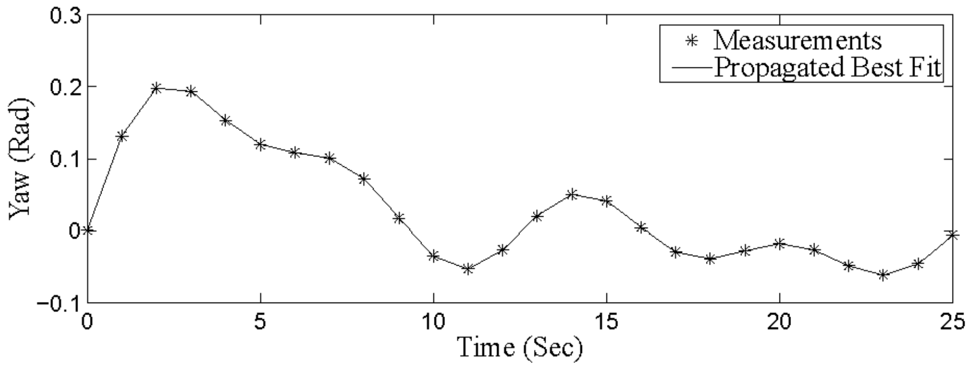
$$\psi(t) = k_1 e^{\lambda_1 t}\sin(\omega_1 t + \delta_1) + k_2 e^{\lambda_2 t}\sin(\omega_2 t + \delta_2) + k_3 e^{\lambda_3 t}\sin(\omega_3 t + \delta_3) + k_5$$

  where $k_1, k_2, k_3, k_4, k_5, \lambda_1, \lambda_2, \lambda_3, \omega_1, \omega_2, \omega_3, \delta_1, \delta_2, \delta_3$ are 14 constants which can be related to the aerodynamic and mass characteristics of the projectile and to the initial motion conditions

  - These constants are often estimated by nonlinear least squares to "best fit" measured pitch and yaw histories modeled by the above equations
  - Consider simulated measurements of $\theta(t)$ and $\psi(t)$ with the measurement error generated by using a zero-mean Gaussian noise process with a standard deviation given by $\sigma = 0.0002$
  - 52 measurements are sampled at 1 second intervals

NLS Example III (ii)





Constant Parameter	Start Value	True Value
k_1	0.5000	0.2000
k_2	0.2500	0.1000
k_3	0.1250	0.0500
k_4	0.0000	0.0001
k_5	0.0000	0.0001
λ_1	-0.1500	-0.1000
λ_2	-0.0600	-0.0500
λ_3	-0.0300	-0.0250
ω_1	0.2600	0.2500
ω_2	0.5500	0.5000
ω_3	0.9500	1.0000
δ_1	0.0100	0.0000
δ_2	0.0100	0.0000
δ_3	0.0100	0.0000

Note: starting values are not that far off the true values

NLS Example III (iii)

– Code this up

- The nonlinear least squares variables are given by

$$\overset{(14\times 1)}{\mathbf{x}} = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & \lambda_1 & \lambda_2 & \lambda_3 & \omega_1 & \omega_2 & \omega_3 & \delta_1 & \delta_2 & \delta_3 \end{bmatrix}^T$$

$$\overset{(52\times 1)}{\tilde{\mathbf{y}}} = \begin{bmatrix} \tilde{\theta}(0) & \tilde{\psi}(0) & \tilde{\theta}(1) & \tilde{\psi}(1) & \cdots & \tilde{\theta}(25) & \tilde{\psi}(25) \end{bmatrix}^T$$

$$\overset{(52\times 14)}{H} = \begin{bmatrix} \left.\dfrac{\partial\theta(0)}{\partial x_1}\right|_{\mathbf{x}_c} & \cdots & \left.\dfrac{\partial\theta(0)}{\partial x_{14}}\right|_{\mathbf{x}_c} \\ \left.\dfrac{\partial\psi(0)}{\partial x_1}\right|_{\mathbf{x}_c} & \cdots & \left.\dfrac{\partial\psi(0)}{\partial x_{14}}\right|_{\mathbf{x}_c} \\ \vdots & & \vdots \\ \left.\dfrac{\partial\theta(25)}{\partial x_1}\right|_{\mathbf{x}_c} & \cdots & \left.\dfrac{\partial\theta(25)}{\partial x_{14}}\right|_{\mathbf{x}_c} \\ \left.\dfrac{\partial\psi(25)}{\partial x_1}\right|_{\mathbf{x}_c} & \cdots & \left.\dfrac{\partial\psi(25)}{\partial x_{14}}\right|_{\mathbf{x}_c} \end{bmatrix}, \quad \overset{(52\times 52)}{W} = 10^8 \begin{bmatrix} \overset{= (1/0.0002)^2}{0.25} & & & 0 \\ & 0.25 & & \\ & & \ddots & \\ 0 & & & 0.25 \end{bmatrix}$$

NLS Example III (iv)

- Convergence history

Parameter	Iteration Number					σ
	0	1	2	$\cdots$	5	
k_1	0.5000	0.1852	0.1975		0.1999	0.0006
k_2	0.2500	0.1075	0.1012		0.0997	0.0005
k_3	0.1250	0.0567	0.0505		0.0500	0.0001
k_4	0.0000	-0.0006	0.0001		0.0002	0.0001
k_5	0.0000	-0.0018	-0.0005		0.0001	0.0001
λ_1	-0.1500	-0.1234	-0.0954		-0.0998	0.0004
λ_2	-0.0600	-0.0661	-0.0585		-0.0497	0.0004
λ_3	-0.0300	-0.0398	-0.0338		-0.0250	0.0002
ω_1	0.2600	0.2490	0.2471		0.2500	0.0004
ω_2	0.5500	0.5300	0.4955		0.4999	0.0004
ω_3	0.9500	0.9697	1.0068		0.9998	0.0002
δ_1	0.0100	0.0344	0.0143		0.0010	0.0031
δ_2	0.0100	-0.0447	0.0051		0.0001	0.0048
δ_3	0.0100	0.0024	-0.0570		-0.0001	0.0024

NLS Example III (v)

- Observe the rather dramatic convergence progress shown in the results
- The rightmost column is obtained by taking the square root of the 14 diagonal elements of $(H^T W H)^{-1}$ on the final iteration
  - We prove this interpretation later
  - Note that the convergence errors are comparable in size to the corresponding σ
  - Weighted sum square of residuals at each iteration is given by

Cost	Iteration Number				
	0	1	2	$\cdots$	5
J	1.08×10^7	2.51×10^5	1.17×10^4		1.93×10^1

- Dramatic convergence is evidenced by the decrease of the weighted sum square of the residuals by six orders of magnitude in five iterations

LM Algorithm (i)

- Levenberg-Marquardt (LM) algorithm used to overcome the deficiencies of both NLS and the gradient algorithms

↖ linearize f ↖ linearization of loss function

$$J = \frac{1}{2}\Delta\mathbf{y}^T W \Delta\mathbf{y} \equiv \frac{1}{2}[\tilde{\mathbf{y}} - \mathbf{f}(\hat{\mathbf{x}})]^T W [\tilde{\mathbf{y}} - \mathbf{f}(\hat{\mathbf{x}})] \qquad (1)$$

  - Take the gradient with respect to the estimate, evaluated at x_c

$$\nabla_{\hat{\mathbf{x}}} J\big|_{\mathbf{x}_c} = -H^T W [\tilde{\mathbf{y}} - \mathbf{f}(\mathbf{x}_c)] \equiv -H^T W \Delta\mathbf{y}_c$$

where

$$H \equiv \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\mathbf{x}_c}$$

  - The method of gradients seeks corrections down the gradient

$$\Delta\mathbf{x} = -\frac{1}{\eta}\nabla_{\hat{\mathbf{x}}} J = \frac{1}{\eta} H^T W \Delta\mathbf{y}_c$$

where $1/\eta$ is a scalar which controls the step size

LM Algorithm (ii)

- The LM correction is then given by

$$\Delta\mathbf{x} = (H^T W H + \eta\mathcal{H})^{-1} H^T W \Delta\mathbf{y}_c \tag{2}$$

$$\hat{\mathbf{x}} = \mathbf{x}_c + \Delta\mathbf{x} \tag{3}$$

  where $\mathcal{H}$ is a diagonal matrix with entries given by the diagonal elements of $H^T W H$ or in some cases simply the identity matrix
  - The search direction is an intermediate between the steepest descent and the differential correction direction
- As $\eta \to 0$ the correction is equivalent to the differential correction method (Non linear least squares)
- As $\eta \to \infty$ the correction reduces to a steepest descent search along the negative gradient of J (Gradient method)
- Controlling η is a heuristic art form that can be tuned by the user
  - Generally η is large in early iterations and should definitely be reduced toward zero in the region near the minimum

- Typical recipe for implementing the LM algorithm

1. Compute Eq. (1) using an initial estimate for $\hat{\mathbf{x}}$, denoted by $\mathbf{x}_c$.

2. Use Eqs. (2) and (3) to update the current estimate with a large value for η (usually much larger than the norm of $H^T W H$, typically 10 to 100 times the norm).

3. Recompute Eq. (1) with the new estimate. If the new value for Eq. (1) is $\geq$ the value computed in step 1, then the new estimate is disregarded and η is replaced by $f\eta$, where f is a fixed positive constant, usually between 1 and 10 (we suggest a default of 5). Otherwise, retain the estimate, and replace η with η/f.

4. After each subsequent iteration, compare the new value of Eq. (1) with its value using the previous estimate and replace η with $f\eta$ or η/f as in step 3. The estimate $\hat{\mathbf{x}}$ is retained if J in Eq. (1) continues to decrease and discarded if Eq. (1) increases.

LM Example (i)

- Same as last example
  - Starting value for λ_1 is set to -0.8500 instead of -0.1500
  - For this initial value, the standard least squares solution diverges rapidly with each iteration
  - Use the LM algorithm with $\eta = 1 \times 10^6$
  - Convergence now occurs

Parameter	Iteration Number				
	0	10	15	$\cdots$	20
k_1	0.5000	0.3601	0.0844		0.1999
k_2	0.2500	0.1946	0.2099		0.0997
k_3	0.1250	0.0905	0.0620		0.0500
k_4	0.0000	−0.0062	0.0111		0.0002
k_5	0.0000	−0.0047	−0.0004		0.0001
λ_1	−0.8500	−0.7977	−0.0436		−0.0998
λ_2	−0.0600	−0.0760	−0.1270		−0.0497
λ_3	−0.0300	−0.0418	−0.0436		−0.0250
ω_1	0.2600	0.1094	0.1621		0.2500
ω_2	0.5500	0.5505	0.4950		0.4999
ω_3	0.9500	0.9582	0.9874		0.9998
δ_1	0.0100	0.0060	0.5068		0.0010
δ_2	0.0100	−0.1234	−0.3482		0.0001
δ_3	0.0100	0.1225	0.1918		−0.0001
η	10^6	0.5120	0.0041		10^{-6}

LM Example (ii)

```
% True Values
k1=0.2;k2=0.1;k3=0.05;k4=0.0001;k5=0.0001;
lam1=-0.1;lam2=-0.05;lam3=-0.025;
w1=0.25;w2=0.5;w3=1.0;
d1=0;d2=0;d3=0;

% Measurements
t=[0:1:25]';m=length(t);
theta=k1*exp(lam1*t).*cos(w1*t+d1)+k2*exp(lam2*t).*cos(w2*t+d2) ...
    +k3*exp(lam3*t).*cos(w3*t+d3)+k4;

psi=k1*exp(lam1*t).*sin(w1*t+d1)+k2*exp(lam2*t).*sin(w2*t+d2) ...
  +k3*exp(lam3*t).*sin(w3*t+d3)+k5;

thetam=theta+0.0002*randn(m,1);
psim=psi+0.0002*randn(m,1);
ym=[thetam;psim];

% Factor for Levenberg-Marquardt Algorithm
fac=1e6;facc(1)=fac;
```

LM Example (iii)

```
% Initial Conditions
xc=[0.5;0.25;0.125;0;0;-0.85;-0.06;-0.03;0.26;0.55;0.95;0.01;0.01;0.01];
dx=ones(14,1);i=1;clear xe;xe(1,:)=xc';

%Levenberg-Marquardt Algorithm
while norm(dx)>1e-6,

i=i+1;if (i > 50), break, end

k1e=xc(1);k2e=xc(2);k3e=xc(3);k4e=xc(4);k5e=xc(5);
lam1e=xc(6);lam2e=xc(7);lam3e=xc(8);
w1e=xc(9);w2e=xc(10);w3e=xc(11);
d1e=xc(12);d2e=xc(13);d3e=xc(14);

thetae=k1e*exp(lam1e*t).*cos(w1e*t+d1e)+k2e*exp(lam2e*t).*cos(w2e*t+d2e) ...
    +k3e*exp(lam3e*t).*cos(w3e*t+d3e)+k4e;

psie=k1e*exp(lam1e*t).*sin(w1e*t+d1e)+k2e*exp(lam2e*t).*sin(w2e*t+d2e) ...
   +k3e*exp(lam3e*t).*sin(w3e*t+d3e)+k5e;
```

LM Example (iv)

```
h=[exp(lam1e*t).*cos(w1e*t+d1e) exp(lam2e*t).*cos(w2e*t+d2e) ...
  exp(lam3e*t).*cos(w3e*t+d3e) ones(m,1) zeros(m,1) ...
  k1e*t.*exp(lam1e*t).*cos(w1e*t+d1e) k2e*t.*exp(lam2e*t).*cos(w2e*t+d2e) ...
  k3e*t.*exp(lam3e*t).*cos(w3e*t+d3e) -k1e*t.*exp(lam1e*t).*sin(w1e*t+d1e) ...
  -k2e*t.*exp(lam2e*t).*sin(w2e*t+d2e) -k3e*t.*exp(lam3e*t).*sin(w3e*t+d3e) ...
  -k1e*exp(lam1e*t).*sin(w1e*t+d1e) -k2e*exp(lam2e*t).*sin(w2e*t+d2e) ...
  -k3e*exp(lam3e*t).*sin(w3e*t+d3e)
  exp(lam1e*t).*sin(w1e*t+d1e) exp(lam2e*t).*sin(w2e*t+d2e) ...
  exp(lam3e*t).*sin(w3e*t+d3e) zeros(m,1)  ones(m,1) ...
  k1e*t.*exp(lam1e*t).*sin(w1e*t+d1e) ...
  k2e*t.*exp(lam2e*t).*sin(w2e*t+d2e) ...
  k3e*t.*exp(lam3e*t).*sin(w3e*t+d3e) ...
  k1e*t.*exp(lam1e*t).*cos(w1e*t+d1e) ...
  k2e*t.*exp(lam2e*t).*cos(w2e*t+d2e) ...
  k3e*t.*exp(lam3e*t).*cos(w3e*t+d3e) ...
  k1e*exp(lam1e*t).*cos(w1e*t+d1e) ...
  k2e*exp(lam2e*t).*cos(w2e*t+d2e) ...
  k3e*exp(lam3e*t).*cos(w3e*t+d3e)];
```

LM Example (v)

```
dy=ym-[thetae;psie];
jold=sum(dy'*dy);

dx=inv(h'*h + fac*diag(diag(h'*h)))*h'*(ym-[thetae;psie]);
xc=xc+dx;
xe(i,:)=xc';

k1e=xc(1);k2e=xc(2);k3e=xc(3);k4e=xc(4);k5e=xc(5);
lam1e=xc(6);lam2e=xc(7);lam3e=xc(8);
w1e=xc(9);w2e=xc(10);w3e=xc(11);
d1e=xc(12);d2e=xc(13);d3e=xc(14);

thetae=k1e*exp(lam1e*t).*cos(w1e*t+d1e)+k2e*exp(lam2e*t).*cos(w2e*t+d2e) ...
      +k3e*exp(lam3e*t).*cos(w3e*t+d3e)+k4e;

psie=k1e*exp(lam1e*t).*sin(w1e*t+d1e)+k2e*exp(lam2e*t).*sin(w2e*t+d2e) ...
    +k3e*exp(lam3e*t).*sin(w3e*t+d3e)+k5e;
dy=ym-[thetae;psie];
jnew=sum(dy'*dy);
```

LM Example (vi)

```
% Update Factor
if jnew < jold
 fac=fac/5;
else
 fac=fac*5;
end

facc(i)=fac;

cost(i)=.25e8*dy'*dy*0.5;

end
```