Preface

We trust that these homework problem solutions will prove helpful in teaching a course with our text. If you find typographical errors please send us corrections via John Wiley.

Allen J. Wood
Bruce f. Wollenberg
Gerald B. Sheblé
INTRODUCTION

This text assumes that all students have had a probability course like:

Iowa State University:  **E E 322. Probabilistic Methods for Electrical Engineers.** (Cross-listed with Stat). (3-0) Cr. 3. F.S. **Prereq:** E E 224. Introduction to probability with applications to electrical engineers. Sets and events, probability space, conditional probability, total probability and Bayes' rule. Discrete and continuous random variables, cumulative distribution function, probability mass and density functions, expectation, moments, moment generating function, multiple random variables, and functions of random variables.

Based on the text: *Continuous and Discrete Signal and System Analysis* by Clare D. McGillem and George R. Cooper

MATLAB toolbox used in the following includes the basic functions. A MATLAB Toolbox for Time Series Modeling should be used for these problems or at least as a great reference.

We used the functions: **filter**, **xcorr**, and **aryule**

An alternate toolbox is the free Matlab MFE Toolbox developed by Kevin Shepherd at:
http://www.kevinsheppard.com/wiki/Main_Page

https://bitbucket.org/kevinsheppard/mfe_toolbox/src

This site include the Matlab toolbox code as well as excellent documentation.
**Solution to Problem 12.1**

The task is to find the variance of a summation of random variables. If the process is stationary, then the mean and variance is the same for the random variable sampled at difference points in time. An alternate more general question is if the sample mean as an estimate for the time series mean for a stationary time series? If the series is auto-correlated, what is the recommended way to estimate the variance?

This is just for AR(1).

The mean can be estimated from the sample mean.

Stationarity means that the expected value is constant: \( E[Y_t] = c \) for all \( t \). The sample mean should estimate \( c \).

The variance of a stationary, zero mean AR(1) series:

\[ Y_t = \phi Y_{t-1} + w_t \]

Where \( \{w_t\} \) is white noise. Then the \( \text{Var}(Y_t) \) is found:

\[ \text{Var}(Y_t) = \sigma^2 (1 - \phi^2) = \sigma^2 \]

If one has just observed \( Y_t \) and are about to observe the next tick in time \( t+1 \), then the conditional is used instead:

\[ \text{Var}(Y_{t+1}|Y_t) = E[(Y_{t+1} - E[Y_{t+1}|Y_t])^2|Y_t] \]

\[ = E[(Y_{t+1} - \phi Y_t)^2|Y_t] \]

\[ = E[w_{t+1}^2|Y_t] = \text{Var}(w_t) = \sigma^2 \]

This is the way to imagine the process, as it is constructed one step at a time as each step depends on previous values. Calculating the \( \text{Var}(Y_{t+1}) \) assumes that all the instantiations of \( \{Y_t\} \) are available at once, but are found one-at-a-time, leaving only the \( \text{Var}(Y_{t+1}|Y_t) \) to be found.

If the time series is indeed stationary, you may take the sample variance. Note that since the time series is auto-correlated, one should be more interested in the variance of the residuals.

Stationary means that the joint distribution of the time series is unaffected by time shifts. One can define a meaningful marginal distribution for a single point. Since this time series is auto-correlated, the conditional distribution of a particular value given the preceding values may be quite different from that marginal distribution. The difference between the expected mean at time \( t \), given the time series prior to \( t \), and the actual value is called the innovation. Measuring the variance of the innovation will give you a better idea of how much noise is included in the process.
Problem 12.2 Solution

Convert one polynomial to another. The polynomial function $F(z)$ is nonzero for $|z| \leq 1$ for any stationary AR($p$) model. The function

$$F(z) = \frac{1}{G(z)}$$

is analytic for $|z| \leq 1$ and has a power series representation with summable coefficients,

$$F(z) = 1 + \alpha_1 z + \alpha_2 z + \cdots$$

The AR($p$) model can be rearranged:

$$\mu + G(B) w = \frac{1}{F(B)} (c + w)$$

with

$$\mu = \frac{c}{G(B)}$$

where B is the identity applied to a constant. The coefficients of F(z) are the coefficients of the MA($\infty$). B is a bounded operator. The terms with power of B map exactly to shifts of the sequences.

MATLAB function `garchma(AR,MA,NumLags)` yields the InfiniteMA coefficients for a specified order.
Solution to Problem 12.3

MATLAB functions are given in form: \texttt{autocorr(y, numLags, numMA, numSTD)} to find the autocorrelations. The autocorrelation code is also on internet for these algorithms. MATLAB functions are given in form: \texttt{parcorr(y, numLags, numAR, numSTD)} to find the partial autocorrelations. The partial autocorrelation code is also on internet for these algorithms.

The student is to arbitrarily generate the various order of ARMA processes. The student should then find that the ACF and PACF graphs show the following patterns.
Problem 12.3 Solution continued

The Matlab filter function is used to generate AR and MA.

By definition a single variable ARMA sequence is defined as:

\[ y_t = \phi_0 + \sum_{p=1}^{P} \phi_p y_{t-p} + \sum_{q=1}^{Q} \theta_q \epsilon_{t-q} + \epsilon_t \]

where \( \epsilon = n(0,1) \)

The filter function call is

```matlab
% generate ARIMA(p,q) process for T time step samples
x = randn(1,T);
y = filter(theta, phi, x);
```

where: \( \text{theta} = [1 \ \theta_1 \ \theta_2 \ ... \ \theta_q] \)

\( \text{phi} = [1 \ \phi_1 \ \phi_2 \ ... \ \phi_p] \)

\( T = \text{number of samples, Here we used 1000 throughout} \)

ARORDER is \( p \)

MAORDER is \( q \)

The Matlab code to generate the ACF and the Partial Autocorrelation graphs below were obtained from the Mathworks help site. The code to build both is listed on the next page. Those teaching this course may simply copy the code on the next page and paste it into the Matlab editor to use.
Problem 12.3 Solution continued

Assuming that the vector of samples with 1000 entries is in vector y.

```matlab
[xc,lags] = xcorr(y,50,'coeff');
figure(01)
stem(lags(51:end),xc(51:end),'markerfacecolor',[0 0 1])
xlabel('Lag'); ylabel('ACF');
title('Sample Autocorrelation Sequence');

[arcoefs,E,K] = aryule(y,15);
pacf = -K;
lag = 1:15;
figure(02)
stem(lag,pacf,'markerfacecolor',[0 0 1]);
xlabel('Lag'); ylabel('Partial Autocorrelation');
set(gca,'xtick',1:1:15)
lconf = -1.96/sqrt(1000)*ones(length(lag),1);
uconf = 1.96/sqrt(1000)*ones(length(lag),1);
hold on;
line(lag,lconf,'color',[1 0 0]);
line(lag,uconf,'color',[1 0 0]);
```
Problem 12.3 Solution continued

ARMA(1,0) using Matlab to generate samples (NO NOISE ADDED)

\[ x(1) = 1; \]
\[ \text{for } t = 2:1000 \]
\[ \quad x(t) = 0.9 \times x(t-1); \]
\[ \text{end} \]

Sample Autocorrelation Sequence

Partial Autocorrelation
ARMA(1,0) using Matlab to generate samples (WITH NOISE ADDED)

\[
x(1) = 1;
\text{for } t = 2:1000
\quad x(t) = 0.9 \times x(t-1) + \text{randn};
\end{align*}

end
Problem 12.3 Solution continued

ARMA(1,0) using filter Function which adds noise N(0,1)

\[ x(t) = 0.9 \times x(t-1) + n(0,1); \] \( n \) is a random variable \((0,1)\)

\[ T=1000; \]
\[ \phi = [1.0, -0.9]; \]
\[ \theta = [1.0]; \]
\[ x = \text{randn}(1,T); \]
\[ y = \text{filter}(\theta, \phi, x); \]
Problem 12.3 Solution continued

ARMA(0,1) using filter function which adds noise N(0,1)

Function should generate using this

\[ x(t) = 0.9n(t-1) + n(0,1); \text{ n is a random variable (0,1)} \]

\[
T=1000; \\
phi = [1]; \\
theta=[1.0 0.9]; \\
x=randn(1,T); \\
y = filter(theta, phi, x);
\]
Problem 12.3 Solution continued

ARMA(2,0) using Matlab to generate samples

\[
x(1) = 1;
x(2) = 2;
\%
\text{for } i = 3:1000
\quad x(i) = 0.9\times x(i-1) - 0.4\times x(i-2);
\text{end}
\]
Problem 12.3 Solution continued

ARMA(2,0) using filter function

\[
T=1000; \\
phi = [1 0.9 -0.4]; \\
theta=[1]; \\
x=randn(1,T); \\
y = filter(theta, phi, x);
\]
Problem 12.4 solution

One should assume initial series coefficients and then generate time series data. MATLAB function fft(X,n,dim) provides the frequency spectrum for a given sampled input. Alternatively, the student may use MATLAB code instead of a toolbox. The Fourier algorithm source code, in MATLAB, is found in the appendices to HARTLEY_MATLAB file. The Hartley algorithm in MATLAB is found in the appendices to FFT_MATLAB file. The routine will provide the frequency spectrum just as the FFT algorithm does.

Comparisons of the two algorithms should include the following salient points:

The replacement of complex multiplications in a DFT by real multiplications in a DHT, two real multiplications and one real addition.

Conversion to the DFT coefficient requires on a real addition and division by 2.

The next higher order FHT can be found by combining two identical preceding lower order FHTs. If N = 2, the Hartley transform can be represented in a 2 x 2 matrix form which is composed of sub-matrices of the matrix form for N = 2. This can be extended to any order which is a power of 2.

The Hartley transform is a bilateral transform where the same functional form can be used for both the forward and inverse transforms.

The DFHT is the most efficient, as of this date, in terms of static memory use (executable size).

As new algorithms are brought forward for the DFFT and the DFHT, the above comments will change.

Continued next page
% testing fft
clc

%SET UP AND PLOT SERIES AS TIME FUNCTION
Fs = 1000;            % Sampling frequency
T = 1/Fs;             % Sample time
L = 1000;            % Length of signal
t = (0:L-1)*T;        % Time vector
% Sum of a 100 Hz sinusoid and a 200 Hz sinusoid
y = 0.5*sin(2*pi*100*t) + sin(2*pi*240*t);
y = y + 2*randn(size(t)); % Sinusoids plus noise
figure(01)
plot(Fs*t(1:100),y(1:100))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')

%CALC FFT AND PLOT FREQ SPECTRUM
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
figure(02)
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')

Plot results are on the next page.
Problem 12.4 solution continued

Here are the signal plot and the fft result

The above Matlab code and plots are given as a demonstration of the fft. Next we demonstrate how a time series can represent a sine wave.
Problem 12.4 solution continued

Representing a sine wave with a time series

In representing the sine function using a time series of the type:

\[ y_t = \phi_0 + \sum_{p=1}^{P} \phi_p y_{t-p} \]

If we sample at \( t = t_n \) then this sample will be the negative of a sample taken at \( t = t_n - T / 2 \) for our sample above with parameters given below, for 10 samples any sample at 5 time steps previous will be its negative.

\[
\begin{align*}
N &= 10 \\
T &= N \left(\frac{1}{F_s}\right) \\
T &= \frac{1}{f} \\
\frac{1}{f} &= 10 \left(\frac{1}{F_s}\right) \\
F_s &= 10 f
\end{align*}
\]

Then we conclude that \( ARorder = N / 2 = 5 \)

\[
\phi = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \end{bmatrix}
\]

Resulting time series is:

\[ y_t = -y_{t-5} \]

The resulting time series is generated using the Matlab code shown on the next page and the plots follow. \( Fs=500, f=50 \)
Problem 12.4 solution continued

Resulting autocorrelations and partial autocorrelation plots.

ARorder = 5;
phi = [ 0 0 0 0 -1];
Fs = 500
Problem 12.4 solution continued

Resulting time series and fft plots.

NOTE: we have shown how a simple sampled sine function has been transformed into a time series. Also note that the function $y_t = -y_{t-5}$ also applies to odd harmonics of the same magnitude as the base frequency.
Problem 12.4 solution continued

We now generate a time series as an ARMA process and plot its autocorrelation function, its partial autocorrelation and then plot it as a time series. Lastly we run the time series through the fft function and plot the frequency spectrum. The time series to be generated is below code is given next:

\[ y(k) + 0.4y(k-1) = w(k) - 0.6w(k-1) \]

```matlab
%GENERATE ARMA TIME SERIES USING FILTER FUNCTION
T=1000;
constant = 0;
ARorder=1;
phi = [1.0 0.4];
MAorder=1;
theta=[1.0 -0.6];
y = armaxfilter_simulate(T, constant, ARorder, phi , MAorder, theta);

% generate ARIMA(p,q) process for T time step samples
x=randn(1,T);
y = filter(theta, phi, x);

%PLOT AUTOCORRELATION AND PARTIAL AUTOCORRELATION FUNCTIONS
[xc,lags] = xcorr(y,50, 'coeff');
figure(01)
stem(lags(51:end),xc(51:end), 'markerfacecolor', [0 0 1])
xlabel('Lag'); ylabel('ACF');
title('Sample Autocorrelation Sequence');

% generate ARIMA(p,q) process for T time step samples

%PLOT AUTOCORRELATION AND PARTIAL AUTOCORRELATION FUNCTIONS
[arcoefs,E,K] = aryule(y,15);
pacf = -K;
lag = 1:15;
figure(02)
stem(lag,pacf, 'markerfacecolor', [0 0 1]);
xlabel('Lag'); ylabel('Partial Autocorrelation');
title('Sample Partial Autocorrelation Sequence');
set(gca, 'xtick',1:1:15)
lconf = -1.96/sqrt(1000)*ones(length(lag),1);
uconf = 1.96/sqrt(1000)*ones(length(lag),1);
hold on;
line(lag,lconf, 'color', [1 0 0]);
line(lag,uconf, 'color', [1 0 0]);

%SET UP AND PLOT SERIES AS TIME FUNCTION
Fs = 1000; % Sampling frequency
T = 1/Fs;
% Sample frequency
L = 1000; % Length of signal
% Length of signal
t = (0:L-1)*T; % Time vector
% Time vector
figure(03)
p = plot(Fs*t(1:100),y(1:100))
title('ARMA time series as a signal')
xlabel('Time (milliseconds)')

%CALC FFT AND PLOT FREQUENCY SPECTRUM
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(y,NFFT)/L;
%Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
figure(04)
p = plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')

[Pxx,F] = pyulear(y,4,1024,1);
figure(05);
p = plot(F,10*log10(Pxx), 'r'); hold on;
legend('True Power Spectral Density', 'PSD Estimate')
```

The plots are given on the following pages
Problem 12.4 solution continued

Sample Autocorrelation Sequence

Sample Partial Autocorrelation Sequence

Problem 12.4 solution continued
ARMA time series as a signal

Single-Sided Amplitude Spectrum of $y(t)$
Problem 12.4 solution continued

Additional Matlab logic plots the power spectral density

```matlab
[Pxx,F] = pyulear(y,4,1024,1);
figure(05);
plot(F,10*log10(Pxx), 'r'); hold on;
legend('True Power Spectral Density', 'PSD Estimate')
```

![Graph showing the true power spectral density and its estimate](image-url)
Problem 12.5 Solution

Yule-Walker proof: PARTE-6-cavazos-isais and others available in the University of Minnesota CUSP web site (http://www.cusp.umn.edu)

Yule-Walker proof follows. Others abound on the internet.

See yule-walker pdf file in the University of Minnesota CUSP web site (http://www.cusp.umn.edu) for extra information on ch 12 solutions.

Run the MATLAB DSP Toolbox function: Yule-Walker AR Estimator to compute estimate of autoregressive (AR) model parameters using Yule-Walker method

Alternatively use the Burg method as a comparison to compute estimate of autoregressive (AR) model parameters using Burg method. Reference comparison of Yule-Walker and Burg algorithm in the following.
Problem 12.6 Solution

Use the process in problem 12.2 or 12.4 to find the parameters given using weighted least squares (WLS) method. State estimation is the same algorithm and that code should be used here. Note that the unknowns are now the parameters with the measurements being the known values. The objective function is still the minimization of the error squared.

We are going to use some data generated using this formula and random numbers generated by a random number generator to get the $y(k)$ values on the next page.

\[ y(k) = -0.4y(k - 1) + w(k) - 0.6w(k - 1) \]

We will generalize this equation to use variables as shown below.

\[ y(k) = \mu y(k - 1) + \theta_t w(k) + \theta_{t-1} w(k - 1) \]

Where we see that
\[ \mu = -0.4 \]
\[ \theta_t = 1 \]
\[ \theta_{t-1} = -0.6 \]

The data used was first generated in a spreadsheet as shown on the next page.
Problem 12.6 Solution continued

Data for Problem 12.6 from spreadsheet

<table>
<thead>
<tr>
<th>k</th>
<th>y(k)</th>
<th>w(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0.653503797</td>
</tr>
<tr>
<td>0</td>
<td>0.497923871</td>
<td>0.890026149</td>
</tr>
<tr>
<td>1</td>
<td>-0.471329106</td>
<td>0.261856132</td>
</tr>
<tr>
<td>2</td>
<td>0.15505843</td>
<td>0.123640467</td>
</tr>
<tr>
<td>3</td>
<td>0.117255364</td>
<td>0.253463016</td>
</tr>
<tr>
<td>4</td>
<td>0.349013638</td>
<td>0.547993593</td>
</tr>
<tr>
<td>5</td>
<td>0.476156763</td>
<td>0.944567284</td>
</tr>
<tr>
<td>6</td>
<td>-0.258284779</td>
<td>0.498921861</td>
</tr>
<tr>
<td>7</td>
<td>0.586446897</td>
<td>0.782486102</td>
</tr>
<tr>
<td>8</td>
<td>0.003650517</td>
<td>0.707720937</td>
</tr>
<tr>
<td>9</td>
<td>-0.203120208</td>
<td>0.222780741</td>
</tr>
<tr>
<td>10</td>
<td>0.312074244</td>
<td>0.364417877</td>
</tr>
<tr>
<td>11</td>
<td>0.05916459</td>
<td>0.402645013</td>
</tr>
<tr>
<td>12</td>
<td>-0.140667135</td>
<td>0.12458569</td>
</tr>
<tr>
<td>13</td>
<td>0.542195514</td>
<td>0.560680067</td>
</tr>
<tr>
<td>14</td>
<td>0.373257186</td>
<td>0.926543432</td>
</tr>
<tr>
<td>15</td>
<td>0.083722371</td>
<td>0.788951304</td>
</tr>
<tr>
<td>16</td>
<td>-0.461004718</td>
<td>0.045855013</td>
</tr>
<tr>
<td>17</td>
<td>0.420540468</td>
<td>0.263651589</td>
</tr>
<tr>
<td>18</td>
<td>0.670539361</td>
<td>0.996946502</td>
</tr>
<tr>
<td>19</td>
<td>-0.335013949</td>
<td>0.531369697</td>
</tr>
<tr>
<td>20</td>
<td>0.467029166</td>
<td>0.651845405</td>
</tr>
<tr>
<td>21</td>
<td>0.310880326</td>
<td>0.888799236</td>
</tr>
<tr>
<td>22</td>
<td>-0.383850654</td>
<td>0.273781018</td>
</tr>
<tr>
<td>23</td>
<td>0.651222858</td>
<td>0.661951207</td>
</tr>
<tr>
<td>24</td>
<td>0.081068263</td>
<td>0.73872813</td>
</tr>
<tr>
<td>25</td>
<td>-0.422765747</td>
<td>0.052898437</td>
</tr>
<tr>
<td>26</td>
<td>0.226813686</td>
<td>0.08944645</td>
</tr>
<tr>
<td>27</td>
<td>0.547811636</td>
<td>0.69220498</td>
</tr>
<tr>
<td>28</td>
<td>-0.106294339</td>
<td>0.528153303</td>
</tr>
<tr>
<td>29</td>
<td>0.01999416</td>
<td>0.294368406</td>
</tr>
<tr>
<td>30</td>
<td>-0.036334532</td>
<td>0.148284176</td>
</tr>
<tr>
<td>31</td>
<td>0.823324784</td>
<td>0.897761477</td>
</tr>
<tr>
<td>32</td>
<td>-0.155462881</td>
<td>0.712523919</td>
</tr>
<tr>
<td>33</td>
<td>0.229936381</td>
<td>0.59526558</td>
</tr>
<tr>
<td>34</td>
<td>-0.366570004</td>
<td>0.082546897</td>
</tr>
<tr>
<td>35</td>
<td>0.509796877</td>
<td>0.412690214</td>
</tr>
<tr>
<td>36</td>
<td>0.384646236</td>
<td>0.836179115</td>
</tr>
<tr>
<td>37</td>
<td>-0.056839598</td>
<td>0.598936366</td>
</tr>
<tr>
<td>38</td>
<td>0.197821391</td>
<td>0.534531372</td>
</tr>
<tr>
<td>39</td>
<td>0.333871328</td>
<td>0.733718707</td>
</tr>
<tr>
<td>40</td>
<td>0.087823987</td>
<td>0.661603743</td>
</tr>
<tr>
<td>41</td>
<td>0.409588129</td>
<td>0.84167997</td>
</tr>
<tr>
<td>42</td>
<td>-0.137568408</td>
<td>0.531274826</td>
</tr>
<tr>
<td>43</td>
<td>0.414765961</td>
<td>0.678503493</td>
</tr>
<tr>
<td>44</td>
<td>-0.184646535</td>
<td>0.388361945</td>
</tr>
<tr>
<td>45</td>
<td>0.366700047</td>
<td>0.5258586</td>
</tr>
<tr>
<td>46</td>
<td>0.182314742</td>
<td>0.64450992</td>
</tr>
<tr>
<td>47</td>
<td>-0.338784724</td>
<td>0.120847125</td>
</tr>
<tr>
<td>48</td>
<td>0.54054031</td>
<td>0.477534695</td>
</tr>
<tr>
<td>49</td>
<td>-0.031068613</td>
<td>0.471668328</td>
</tr>
<tr>
<td>50</td>
<td>-0.013160491</td>
<td>0.25741306</td>
</tr>
</tbody>
</table>
Problem 12.6 Solution continued

Next a Matlab code was written to solve for the unknown parameters $\mu$, $\theta_t$, and $\theta_{t-1}$.

This is the matlab code that brings the data to the problem:

```matlab
% enter problem 6 data

k y(k) w(k)
data = [  
-1 0  0.653503797  
0 0.497923871 0.890026149  
1 -0.471329106 0.261856132  
2 0.15505843 0.123640467  
3 0.11725364 0.253463016  
4 0.349013638 0.547993593  
5 0.47615673 0.945672824  
6 -0.259284779 0.498921861  
7 0.586446897 0.782486102  
8 0.003650517 0.707720937  
9 -0.203312028 0.222780741  
10 0.312074244 0.364417877  
11 0.05916459 0.402645013  
12 -0.140667153 0.12459569  
13 0.542195514 0.560680067  
14 0.373257186 0.926543432  
15 0.083722371 0.788951304  
16 -0.461004718 0.04585013  
17 0.420540468 0.263651589  
18 0.670539651 0.996946502  
19 -0.335013949 0.531369697  
20 0.467029166 0.651845405  
21 0.310880326 0.88799236  
22 -0.383850654 0.273781018  
23 0.651228258 0.661951207  
24 0.08106263 0.73872813  
25 -0.42765747 0.052894373  
26 0.226813686 0.08944645  
27 0.547811636 0.69220498  
28 -0.106293439 0.528153303  
29 0.01999416 0.294368406  
30 -0.036334532 0.148284176  
31 0.823324784 0.897761477  
32 -0.155462881 0.71253919  
33 0.229936381 0.59526558  
34 -0.366587004 0.082546897  
35 0.509796877 0.412690214  
36 0.384646236 0.836179115  
37 -0.056629598 0.59836366  
38 0.197821391 0.534531372  
39 0.33871328 0.733718077  
40 0.087823987 0.661603743  
41 0.409588129 0.84517997  
42 -0.137568408 0.531274826  
43 0.414765961 0.678503493  
44 -0.184646535 0.388361945  
45 0.366700047 0.5258586  
46 0.182314742 0.64450992  
47 -0.338784724 0.120847125  
48 0.54054031 0.477534695  
49 -0.031068613 0.471668328  
50 -0.03160491 0.25741306];

for k=1:52
  y(k)=data(k,2);
  w(k)=data(k,3);
end
```
Problem 12.6 Solution continued

This is the code for the least squares fit.

```matlab
% calculate the least squares fit to the data of prob 6
% enter the data for problem 6
enterdata6

% remember that y(3) is actually the first entry y(1)

% Allocate space for A matrix and b vector
A = zeros(50,3);
b = zeros(50);

for i = 1:50
    k = i+2;
    b(i) = y(k);
    A(i,1) = y(k-1);
    A(i,2) = w(k);
    A(i,3) = w(k-1);
end

% use the pseudo inverse of A to solve for the least squares solution
x = inv(A'*A)*A'*b;

mu = x(1)
theta_t = x(2)
theta_tminus1 = x(3)

The results are:

mu =
    -0.4000

theta_t =
    1.0000

theta_tminus1 =
    -0.6000

The Least Squares fit is done as follows:

We take the equation \( y(k) = \mu y(k-1) + \theta_t w(k) + \theta_{t-1} w(k-1) \)

And we write the sample values using a matrix like this:

\[
\begin{bmatrix}
y(1) \\
y(2) \\
... \\
y(50)
\end{bmatrix} =
\begin{bmatrix}
y(0) & w(1) & w(0) \\
y(1) & w(2) & w(1) \\
... & ... & ... \\
y(49) & w(50) & w(49)
\end{bmatrix}
\begin{bmatrix}
\mu \\
\theta_t \\
\theta_{t-1}
\end{bmatrix}
\]

The matrix has 50 rows and three columns. We solve for the three unknowns by the pseudo inverse formula.
Problem 12.6 Solution continued

If we write the equation on the previous page as:

\[ b = Ax \]

then the solution is

\[ x = [A^T A]^{-1} A^T b \]

This solution is called the “least squares” solution because it minimizes the following sum

\[
\min \sum_{k=1}^{50} (y(k) - \mu y(k-1) - \theta_t w(k) - \theta_{t-1} w(k-1))^2
\]

That is, the minimum of the sum of the squares of the errors gives us the values of \( \mu, \theta_t, \) and \( \theta_{t-1} \)

These are often called the “least squares estimates” of the three unknowns.
Solution to Problem 12.7

Find the AR parameters for the process of Problem 12.1. First, simulate the AR process to generate the data without adding noise. Then add noise to determine how quickly the estimated parameters change. Many sites on the internet discuss this algorithm. See http://paulbourke.net/miscellaneous/ar/

Process in problem 12.1 is shown below:

\[ y(k) = -\frac{1}{3}y(k-1) - \frac{1}{3}y(k-2) + w(k) \]

We simulate the no noise case using:

```matlab
x(1) = 1;
x(2) = 2;

for i = 3:1000
    x(i) = -0.3333*x(i-1) + 0.3333*x(i-2);
end
```

and for the noice case:

```matlab
T = 1000;
phi = [1.0, -0.333, +0.333];
theta=[1.0];
x = randn(1,T);
y = filter(theta, phi, x);
```
Problem 12.7 solution continued

No Noise Case

```matlab
% GENERATE ARMA TIME SERIES USING MATLAB
x(1) = 1;
x(2) = 2;
%
for i = 3:1000
    x(i) = -0.3333*x(i-1) + 0.3333*x(i-2);
end

% PLOT AUTOCORRELATION AND PARTIAL AUTOCORRELATION FUNCTIONS
[xc, lags] = xcorr(y, 50, 'coeff');
figure(01)
stem(lags(51:end), xc(51:end), 'markerfacecolor', [0 0 1])
xlabel('Lag'); ylabel('ACF');
title('Sample Autocorrelation Sequence');

[arcoefs, E, K] = aryule(y, 15);
pacf = -K;
lag = 1:15;
figure(02)
stem(lag, pacf, 'markerfacecolor', [0 0 1]);
xlabel('Lag'); ylabel('Partial Autocorrelation');
title('Sample Partial Autocorrelation Sequence');
set(gca, 'xtick', 1:1:15)
lconf = -1.96/sqrt(1000)*ones(length(lag), 1);
uconf = 1.96/sqrt(1000)*ones(length(lag), 1);
hold on;
line(lag, lconf, 'color', [1 0 0]);
line(lag, uconf, 'color', [1 0 0]);
```
Problem 12.7 solution continued

No Noise Case

Sample Autocorrelation Sequence

ACF

Partial Autocorrelation

Lag
Problem 12.7 solution continued

\[ T = 1000; \]
\[ \phi = [1.0 \ -0.333 \ +0.333]; \]
\[ \theta = [1.0]; \]
\[ x = \text{randn}(1, T); \]
\[ y = \text{filter}(\theta, \phi, x); \]

**With Noise Case**

NOTE: It takes until lag = 12 for the partial autocorrelation to come within the bands shown, in the no noise case this is achieved by lag = 3.
Solution to Problem 12.8

Use the Innovations Algorithm in MATLAB: see file:

Introduction to Time Series Analysis Lecture 11 Peter Bartlett.pdf available in the University of Minnesota CUSP web site (http://www.cusp.umn.edu)

Also, see http://www.math.chalmers.se/Stat/Grundutb/CTH/tms086/1314/ and download lecture 2 file FTS_14_2.pdf,

These lecture notes are from Mattias Sundén, of Chalmers University.

Where: \( \hat{\nu} \) is the estimate of the variance

\( \hat{\gamma} \) is from the covariance matrix of the time series (which can be had directly from the Matlab cov function.

The thetas are the estimates of the time series parameters.

Sundén’s algorithm for the innovations method is as follows:

Start with \( \hat{\nu}_0 = \hat{\gamma}(0) \)

For \( m = 1,\ldots,n-1 \) and \( k = 0,\ldots,m-1 \), proceed using

\[
\hat{\theta}_{m,m-k} = \frac{1}{\hat{\nu}_k} \left[ \hat{\gamma}(m-k) - \sum_{j=0}^{k-1} \hat{\theta}_{m,m-j} \hat{\theta}_{k,k-j} \hat{\nu}_j \right] 
\]

\[
\hat{\nu}_m = \hat{\gamma}(0) - \sum_{j=0}^{m-1} \hat{\theta}_{m,m-j}^2 \hat{\nu}_j 
\]

A note on the Matlab code. Matlab can have an index of zero in a for statement, but Matlab cannot have zero or negative indices for a vector or matrix. To overcome this limitation, we used an anonymous function, for example:

\[
gamma = @(i) gamma_stored(i+1); \quad \% \text{this is a Matlab "anonymous function" to} \]
\[
\% \text{allow us to address the variable gamma} \]
\[
\% \text{with an index of zero} \]

Thus we use the anonymous function “gamma” to actually pull a cell value from the vector gamma_stored which has indices 1,2,3,... and this simulates pulling values from the cells of a vector gamma with indices 0,1,2,..., The same trick is used to obtain elements of the vector v with indices 0,1,2,...

The code on the next page allows us to simulate the MA process with \( \theta_1 = 0.5 \) and \( \theta_2 = 0.2 \) we give the autocorrelation and partial autocorrelation plots as in previous problem solutions, then the results of the innovations algorithm are given for steps 10, 20, ..., 100.
Problem 12.8 solution continued

%GENERATE MA TIME SERIES USING MFE FUNCTION
T=1000;
constant = 0;
ARorder=0;
phi = [0];
MAorder=2;
theta=[0.5 0.2];
y = armaxfilter_simulate(T, constant, ARorder, phi , MAorder, theta);

%PLOT AUTOCORRELATION AND PARTIAL AUTOCORRELATION FUNCTIONS
[xc,lags] = xcorr(y,50, 'coeff');
figure(01)
stem(lags(51:end),xc(51:end), 'markerfacecolor', [0 0 1])
xlabel('Lag'); ylabel('ACF');
title('Sample Autocorrelation Sequence');

[arcoefs,E,K] = aryule(y,15);
pacf = -K;
figure(02)
stem(lag,pacf, 'markerfacecolor', [0 0 1]);
xlabel('Lag'); ylabel('Partial Autocorrelation');
title('Sample Partial Autocorrelation Sequence');
set(gca, 'xtick', 1:1:15)
lconf = -1.96/sqrt(1000)*ones(length(lag),1);
uconf = 1.96/sqrt(1000)*ones(length(lag),1);
hold on;
line(lag,lconf, 'color', [1 0 0]);
line(lag,uconf, 'color', [1 0 0]);

z=y; % use z as the time series name from here on
numsteps = 101
v_stored = zeros(1,numsteps); % set up storage for v
theta = zeros(numsteps, numsteps);
maxlags = numsteps;
[c1,lags] = xcov(z,maxlags, 'coeff'); % use cov ftn to calculate autocorrelations
% note: the xcov ftn stores results from - maxlags to *maxlags

% this logic moves the gamma values into a table with of values
% for lag going from 0 to maxlags
i = 0;
for j = maxlags+1:2*maxlags+1
  i = i + 1;
  gamma_stored(i) = c1(j);
end

gamma = @(i) gamma_stored(i+1); % this is a Matlab "anonymous function" to
% allow us to address the variable gamma
% with an index of zero
v_stored(1) = gamma(0); % first value in v is gamma(0)

v = @(i) v_stored(i+1); % note: every time we add to v_stored we need to
% redefine the anonymous ftn to get v

% This is the logic from the Mattias Sundén lecture series (see solution % documentation)
for m = 1:(numsteps-1)
  for k = 0:(m-1)
    theta(m,m-k) = (1/v(k))*(gamma(m-k));
    for j = 0:k-1
      theta(m,m-k) = theta(m,m-k) - (1/v(k))*(theta(m,m-j)*theta(k,k-j)*v(j));
    end
  end
  v_stored(m+1) = gamma(0);
  for j = 0:m-1
    v = @(i) v_stored(i+1);
    v_stored(m+1) = v_stored(m+1) - (theta(m,m-j)^2)*v(j);
  end
  %v_stored
end
% Print out the final ten steps to show convergence
for m = 10:10:numsteps
    k = 1;
    fprintf(' %s %2d %s %2d %s %10.5f
', 'theta(', m, ',', k, ')', theta(m, k) );
    k = 2;
    fprintf(' %s %2d %s %2d %s %10.5f
', 'theta(', m, ',', k, ')', theta(m, k) );
    fprintf(' %s %2d %s %10.5f
', 'v(', m, ')', v(m) );
end

Sample Autocorrelation Sequence

Sample Partial Autocorrelation Sequence
Problem 12.8 solution continued

10 steps
\theta(10, 1) = 0.47532 \\
\theta(10, 2) = 0.15473 \\
v(10) = 0.78849

20 steps
\theta(20, 1) = 0.47349 \\
\theta(20, 2) = 0.15400 \\
v(20) = 0.78294

30 steps
\theta(30, 1) = 0.47328 \\
\theta(30, 2) = 0.15355 \\
v(30) = 0.77499

40 steps
\theta(40, 1) = 0.46786 \\
\theta(40, 2) = 0.14418 \\
v(40) = 0.76685

50 steps
\theta(50, 1) = 0.46882 \\
\theta(50, 2) = 0.14047 \\
v(50) = 0.75850

60 steps
\theta(60, 1) = 0.46709 \\
\theta(60, 2) = 0.14191 \\
v(60) = 0.75249

70 steps
\theta(70, 1) = 0.46981 \\
\theta(70, 2) = 0.14246 \\
v(70) = 0.74753

80 steps
\theta(80, 1) = 0.47610 \\
\theta(80, 2) = 0.14210 \\
v(80) = 0.74136

90 steps
\theta(90, 1) = 0.47823 \\
\theta(90, 2) = 0.14077 \\
v(90) = 0.73367

100 steps
\theta(100, 1) = 0.47905 \\
\theta(100, 2) = 0.14201 \\
v(100) = 0.89505
Solution to Problem 12.9

Based on paper by Richter and Sheble (IEEE)

ANN Toolbox in MATLAB:


Also see PDF files: CAINE2003-stabilitypaper and v18-32 available in the University of Minnesota CUSP website (http://www.cusp.umn.edu)

The solution to this problem that we worked out was done using the Mathworks ANN Toolbox (see http://www.mathworks.com/products/neural-network/). Specifically we used the Neural Network Fitting tool as described in Mathworks documentation.

In the case of the problem of identifying the ARIMA process given its time series we followed these steps:

A sample is generated with 1000 time steps. The Matlab function “filter” is used to generate a specific ARIMA process. The values of p and q are varied between 0,1, and 2 for both p and q. Once the time series is generated, we use the xcorr function to calculate the Autocorrelation series and the function aryule to generate the partial Autocorrelation series. The data from these calculations goes to make up the sample that the neural network will use. The first 15 terms from the autocorrelation series make up the top 15 entries in the sample and then first 15 values from the autocorrelation series make up the bottom 15 entries so that the sample has 30 numbers. Then a “target” for the sample is built as a 2 x 1 vector with the p value in the top position and the q value in the second position. We then repeated this process for 200 samples and 200 corresponding targets. The samples are held in a matrix of 30 rows by 200 columns and the targets in a matrix of 2 rows and 200 columns.

To start, we generated time series with a very “clean” input of all zeros and a 1 in the t=1 position and ran it in the neural net program to see if it would build a neural net that could use the sample information (autocorrelation and partial autocorrelation values) to recognize the p and q values. The results of the “no noise” starting values tests are on the next page.
The Neural Network program used part of the sample to train the neural network, part to test it and part to verify that it is working properly. The first graph shows the convergence of the algorithm.

Next we display the error histogram for the results.
Last of all we show the “performance” plots giving the training, validation and test results. The closer the color lines are to the diagonal labeled $Y = T$ on the graphs, the better fit.

As can be seen here the fit is quite close for the no noise case. Next we returned to the time series generation that started with a vector of 1000 normal distributed random values and then produced the final ARIMA time series with the filter function.
The neural network is not able to do as good a job as seen in the following graphs of convergence, error histogram and performance.
Training: $R=0.82919$

Validation: $R=0.57054$

Test: $R=0.39219$

All: $R=0.69823$
The neural network training tool user interface at the end of training looks like this.
% Artificial Neural Network training set builder
clear;
close all
clc

numtimesteps = 1000;
numsets = 200;

p_q_values = zeros(2,numsets);
samples = zeros(30,numsets);
targets = zeros(2,numsets);

for setnum = 1:numsets
    % GENERATE ARMA TIME SERIES USING FILTER FUNCTION
    T=numtimesteps;
    constant = 0;

    % each pass uses random numbers of p = 1, 2, or 3
    pran = rand;
    p=0;
    if pran>0 & pran<= 0.3333
        p = 1;
    end
    if pran>0.3333 & pran<= 0.6666
        p = 2;
    end
    if pran>0.6666 & pran<= 1.0
        p = 3;
    end
    phi = rand([1,p]); % random values of phi
    phi(1) = 1;
    p = p-1;

    % each pass uses random numbers of q = 1, 2, or 3
    qran = rand;
    q=0;
    if qran>0 & qran<= 0.3333
        q = 1;
    end
    if qran>0.3333 & qran<= 0.6666
        q = 2;
    end
    if qran>0.6666 & qran<= 1.0
        q = 3;
    end
    theta = rand([1,q]); % random values of theta
    theta(1) = 1;
    q = q-1;
p_q_values(1,setnum) = p;
p_q_values(2,setnum) = q;

% generate ARIMA(p,q) process for numtimesteps samples
x = randn(1,numtimesteps);
  % x = zeros(1,numtimesteps);
  % x(1) = 1;
y = filter(theta, phi, x);

[xc,lags] = xcorr(y,50,'coeff');
% figure(01)
% stem(lags(51:end),xc(51:end),'markerfacecolor',[0 0 1])
% xlabel('Lag'); ylabel('ACF');
% title('Sample Autocorrelation Sequence');
%
[arcoefs,E,K] = aryule(y,15);
pacf = -K;
% lag = 1:15;
% figure(02)
% stem(lag,pacf,'markerfacecolor',[0 0 1]);
% xlabel('Lag'); ylabel('Partial Autocorrelation');
% set(gca,'xtick',1:1:15)
% lconf = -1.96/sqrt(1000)*ones(length(lag),1);
% uconf = 1.96/sqrt(1000)*ones(length(lag),1);
% hold on;
% line(lag,lconf,'color',[1 0 0]);
% line(lag,uconf,'color',[1 0 0]);

% select first 15 entries from xc and first 15 from pacf
isample_row = 1;
for i = 51:65
  samples(isample_row,setnum) = xc(i);
  isample_row = isample_row + 1;
end
for i = 1:15
  samples(isample_row,setnum) = pacf(i);
  isample_row = isample_row + 1;
end

% ***** end main loop

targets = p_q_values;

% p_q_values
% samples
% targets

nnstart
Problem 12.10 solution

See fast transform algorithms above in problem 4.

*Applied Numerical Methods Using MATLAB* By Won Y. Yang, Wenwu Cao, Tae-Sang Chung, John Morris


Students should look up references to “Least Squares Spectrum Analysis” or LSSA, where they will find mention of the Lomb-Scargle Periodogram. The LSSA method fits sinusoidal functions to data samples. The Lomb-Scargle Periodogram is available directly within Mathworks. In the solution which follows we used the “fastlomb” function available from Mathworks.

Results:

1) We use pure sinusoids as in problem 12.4 to demonstrate that the Lomb-Scargle Periodogram gets us the same result as the FFT.

2) Next we use the same time series modeled in problem 12.4 and estimate its frequency spectrum using FFT

and the Lomb-Scargle Periodogram which are quite similar
Problem 12.10 solution continued

Using pure sinusoids as in problem 12.4

```matlab
% testing fft
clc

%SET UP AND PLOT SERIES AS TIME FUNCTION
Fs = 1000;             % Sampling frequency
T = 1/Fs;              % Sample time
L = 1000;             % Length of signal
t = (0:L-1)*T;        % Time vector
% Sum of a 100 Hz sinusoid and a 200 Hz sinusoid
y = 0.5*sin(2*pi*100*t) + sin(2*pi*240*t);    % Sinusoids plus noise
t = t + 2*randn(size(t));
figure(01)
pplot(Fs*t(1:100),y(1:100))
title('Signal without Noise')
xlabel('time (milliseconds)')

%CALC FFT AND PLOT FREQ SPECTRUM
NFFT = 2^nextpow2(L);   % Next power of 2 from length of y
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);
figure(02)
pplot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')

for i = 1:1000
    t(i) = i/1000; % each sample is every 0.01 second
end

[P,f,alpha] = fastlomb(y,t,1,0.5);
```
Problem 12.10 solution continued

Using FFT

Using Lomb-Scargle
Problem 12.10 solution continued

Now using the main example of problem 12.4 with time series

\[ y(k) + 0.4y(k-1) = w(k) - 0.6w(k-1) \]

We compare the FFT and the Lomb-Scargle spectrums:

![Single-Sided Amplitude Spectrum of y(t)](image1)

![Lomb-Scargle normalized periodogram](image2)
Solution to Problem 12.11

Use generalized weighted least squares with each function. See thesis:

DEMAND_FORECASTING_ARIMA

Also see MATLAB toolbox on curve fitting:


Process for problems is to first assume parameters for each function, generate time series, then curve fit to find the original function parameters.

Execute curve fitting, using MATLAB curve fitting or State Estimation code, for the time series to find the assumed parameters. If assumed parameters are not found, there is a programming problem to be fixed. If the parameters are found, the add noise to the original time series to see parameter drift as more noise is added.

MATLAB code for least squares curve fitting follows for three basic functions.

X is demand for the period, Y is the period number (first period, second period, etc.).

X is input vector corresponding to Y input vector for each value of X.

Length of vector X can be found by `size(X,2)`.

Array size can be found by `size(A,1)` for number of rows, and `size(A,2)` for number of columns.

Remember that all vectors are rows in MATLAB.

Curve fitting the form \( y = ax \), a linear function were the objective is to minimize:

\[
J = \sum_{i=1}^{n} (y_i - ax_i)^2
\]

The first derivative is:

\[
\frac{dJ}{da} = \sum_{i=1}^{n} 2(y_i - ax_i)(-x_i)
\]

Setting the first derivative to zero

\[
a = \frac{\sum_{i=1}^{n} (y_i x_i)}{\sum_{i=1}^{n} x_i^2}
\]
Solution to Problem 12.11 continued

The MATLAB functions are:

```matlab
function a = proportion(X,Y)
XSumSquares = 0.0; %variable for sum of squares of x values.
XYSumProduct = 0.0; %variable for sum of products of x and y values.
n = size (X,2);
for i =1:n
    XSumSquares = XSumSquares + X(i)^2;
    XYSumProduct = XYSumProduct + X(i)*Y(i);
end
a = XYSumProduct/ XSumSquares;
```

Linear fit of y = ax + b follows in a similar fashion to find the slope (a) and the intercept (b):

```matlab
function [a,b] = linear_fit(X,Y)
XSum = 0.0; % variable for the sum of X values
YSum = 0.0; % variable for sum of Y values
XYSumProduct = 0.0; % variable for the sum of products of X and Y values
XSumSquares = 0.0; % variable for the sum of the squares of X
YSumSquares = 0.0; % variable for the sum of the squares of Y
n = size(X,2); % variable for the number of x values
for i = 1,n; 
    XSum = XSum + X(i); 
    YSum = YSum +Y(i); 
    XSumSquares = XSumSquares + X(i)^2; 
    YSumSquares = YSumSquares + Y(i)^2; 
    XYSumProduct = XYSumProduct + X(i)*Y(i); 
end
a = (n* XYSumProduct - Xsum * YSum)/(n*XSumSquares*XSum^2)
b = (XSumSquares*YSum)/(n*XSumSquares - XSum^2)
```

Power curve fit of y = a x^2 starts with the log natural (ln) of both sides, find the new objective function and the new derivative. Again, set the derivative to zero and solve for the unknowns.

```matlab
function a = transformed_squares(X, Y)
n = size (X, 2); % number of x and y values
A = 0.0; % variable for the sum of ln(Y) - 2*ln(X) values
for I = 1:n; 
    A = A + log(Y(iI)-2*log(X(i));
end
A = A / n
a = exp(A);
```

It is assumed that student will use the MATLAB toolbox on curve fitting for these nonlinear functions:  

Alternatively, the derivatives and mismatch could be found and used to alter the State Estimation code.
Solution for Problem 12.11a

\[ y = ae^x \]

<table>
<thead>
<tr>
<th>(a)</th>
<th>0.5 actual</th>
<th>(a_{est})</th>
<th>0.5 estimated solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y)</td>
<td>(x)</td>
<td>noise</td>
<td>(y_{est})</td>
</tr>
<tr>
<td>2.27E-05</td>
<td>-10</td>
<td>0.139571</td>
<td>2.27E-05</td>
</tr>
<tr>
<td>6.17E-05</td>
<td>-9</td>
<td>0.02793</td>
<td>6.17E-05</td>
</tr>
<tr>
<td>0.000168</td>
<td>-8</td>
<td>0.47085</td>
<td>0.000168</td>
</tr>
<tr>
<td>0.000456</td>
<td>-7</td>
<td>0.215954</td>
<td>0.000456</td>
</tr>
<tr>
<td>0.001239</td>
<td>-6</td>
<td>0.362807</td>
<td>0.001239</td>
</tr>
<tr>
<td>0.003369</td>
<td>-5</td>
<td>0.958192</td>
<td>0.003369</td>
</tr>
<tr>
<td>0.009158</td>
<td>-4</td>
<td>0.629592</td>
<td>0.009158</td>
</tr>
<tr>
<td>0.024894</td>
<td>-3</td>
<td>0.56611</td>
<td>0.024894</td>
</tr>
<tr>
<td>0.067668</td>
<td>-2</td>
<td>0.582783</td>
<td>0.067668</td>
</tr>
<tr>
<td>0.18394</td>
<td>-1</td>
<td>0.189995</td>
<td>0.18394</td>
</tr>
<tr>
<td>0.5</td>
<td>0</td>
<td>0.904339</td>
<td>0.5</td>
</tr>
<tr>
<td>1.359141</td>
<td>1</td>
<td>0.583485</td>
<td>1.359141</td>
</tr>
<tr>
<td>3.694528</td>
<td>2</td>
<td>0.388673</td>
<td>3.694528</td>
</tr>
<tr>
<td>10.04277</td>
<td>3</td>
<td>0.868544</td>
<td>10.04277</td>
</tr>
<tr>
<td>27.29908</td>
<td>4</td>
<td>0.190508</td>
<td>27.29908</td>
</tr>
<tr>
<td>74.20658</td>
<td>5</td>
<td>0.460668</td>
<td>74.20658</td>
</tr>
<tr>
<td>201.7144</td>
<td>6</td>
<td>0.587749</td>
<td>201.7144</td>
</tr>
<tr>
<td>548.3166</td>
<td>7</td>
<td>0.052241</td>
<td>548.3166</td>
</tr>
<tr>
<td>1490.479</td>
<td>8</td>
<td>0.141522</td>
<td>1490.479</td>
</tr>
<tr>
<td>4051.542</td>
<td>9</td>
<td>0.831213</td>
<td>4051.542</td>
</tr>
<tr>
<td>11013.23</td>
<td>10</td>
<td>0.670714</td>
<td>11013.23</td>
</tr>
</tbody>
</table>

Objective 0
Solution for Problem 12.11b

\[ y = \sum_{i=1}^{n} a_i e^{-\left(\frac{x-b_i}{c_i}\right)^2} \]

<table>
<thead>
<tr>
<th></th>
<th>a(1)</th>
<th>b(1)</th>
<th>c(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual</td>
<td>0.6</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>estimated</td>
<td>0.5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>solved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>x</th>
<th>noise</th>
<th>yest</th>
<th>error^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.23402E-25</td>
<td>-10</td>
<td>0.449129</td>
<td>2.23402E-25</td>
<td>0</td>
</tr>
<tr>
<td>3.14573E-22</td>
<td>-9</td>
<td>0.448044</td>
<td>3.14573E-22</td>
<td>0</td>
</tr>
<tr>
<td>2.68664E-19</td>
<td>-8</td>
<td>0.559555</td>
<td>2.68664E-19</td>
<td>0</td>
</tr>
<tr>
<td>1.39171E-16</td>
<td>-7</td>
<td>0.151342</td>
<td>1.39171E-16</td>
<td>0</td>
</tr>
<tr>
<td>4.37263E-14</td>
<td>-6</td>
<td>0.110498</td>
<td>4.37263E-14</td>
<td>0</td>
</tr>
<tr>
<td>8.33277E-12</td>
<td>-5</td>
<td>0.737065</td>
<td>8.33277E-12</td>
<td>0</td>
</tr>
<tr>
<td>9.63137E-10</td>
<td>-4</td>
<td>0.279692</td>
<td>9.63137E-10</td>
<td>0</td>
</tr>
<tr>
<td>6.75211E-08</td>
<td>-3</td>
<td>0.519281</td>
<td>6.75211E-08</td>
<td>0</td>
</tr>
<tr>
<td>2.87107E-06</td>
<td>-2</td>
<td>0.714732</td>
<td>2.87107E-06</td>
<td>0</td>
</tr>
<tr>
<td>7.40459E-05</td>
<td>-1</td>
<td>0.809573</td>
<td>7.40459E-05</td>
<td>0</td>
</tr>
<tr>
<td>0.001158272</td>
<td>0</td>
<td>0.604946</td>
<td>0.001158272</td>
<td>0</td>
</tr>
<tr>
<td>0.010989383</td>
<td>1</td>
<td>0.702256</td>
<td>0.010989383</td>
<td>0</td>
</tr>
<tr>
<td>0.063239535</td>
<td>2</td>
<td>0.969914</td>
<td>0.063239535</td>
<td>0</td>
</tr>
<tr>
<td>0.220727665</td>
<td>3</td>
<td>0.132633</td>
<td>0.220727665</td>
<td>0</td>
</tr>
<tr>
<td>0.46728047</td>
<td>4</td>
<td>0.858985</td>
<td>0.46728047</td>
<td>0</td>
</tr>
<tr>
<td>0.220727665</td>
<td>5</td>
<td>0.506867</td>
<td>0.220727665</td>
<td>0</td>
</tr>
<tr>
<td>0.46728047</td>
<td>6</td>
<td>0.716551</td>
<td>0.46728047</td>
<td>0</td>
</tr>
<tr>
<td>0.220727665</td>
<td>7</td>
<td>0.02851</td>
<td>0.220727665</td>
<td>0</td>
</tr>
<tr>
<td>0.063239535</td>
<td>8</td>
<td>0.282027</td>
<td>0.063239535</td>
<td>0</td>
</tr>
<tr>
<td>0.010989383</td>
<td>9</td>
<td>0.253251</td>
<td>0.010989383</td>
<td>0</td>
</tr>
<tr>
<td>0.001158272</td>
<td>10</td>
<td>0.902835</td>
<td>0.001158272</td>
<td>0</td>
</tr>
</tbody>
</table>

objective 0
Solution for Problem 12.11c

\[ y = \left( p_1 x^1 \right) + \left( p_2 x^2 \right) \]

<table>
<thead>
<tr>
<th></th>
<th>p(1)</th>
<th>p(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>estimate</td>
<td>0.500000</td>
<td>0.399998</td>
</tr>
<tr>
<td>solved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>x</th>
<th>noise</th>
<th>yest</th>
<th>error^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>-11</td>
<td>35</td>
<td>-10.0</td>
<td>0.958654</td>
<td>34.999753</td>
</tr>
<tr>
<td>27.9</td>
<td>-9</td>
<td>0.406492</td>
<td>27.899800</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>21.6</td>
<td>-8</td>
<td>0.901766</td>
<td>21.599842</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>16.1</td>
<td>-7</td>
<td>0.395009</td>
<td>16.099879</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>11.4</td>
<td>-6</td>
<td>0.931889</td>
<td>11.399911</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>7.5</td>
<td>-5</td>
<td>0.779743</td>
<td>7.499938</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>4.4</td>
<td>-4</td>
<td>0.372673</td>
<td>4.399961</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>2.1</td>
<td>-3</td>
<td>0.192833</td>
<td>2.099978</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>0.6</td>
<td>-2</td>
<td>0.288384</td>
<td>0.599990</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>-0.1</td>
<td>-1</td>
<td>0.472206</td>
<td>-0.100002</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.048610</td>
<td>0.000000</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>0.818629</td>
<td>0.899997</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>2.6</td>
<td>2</td>
<td>0.881319</td>
<td>2.599990</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>5.1</td>
<td>3</td>
<td>0.918140</td>
<td>5.099977</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>8.4</td>
<td>4</td>
<td>0.849063</td>
<td>8.399960</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>12.5</td>
<td>5</td>
<td>0.867152</td>
<td>12.499938</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>17.4</td>
<td>6</td>
<td>0.872816</td>
<td>17.399910</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>23.1</td>
<td>7</td>
<td>0.733098</td>
<td>23.099878</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>29.6</td>
<td>8</td>
<td>0.850420</td>
<td>29.599841</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>36.9</td>
<td>9</td>
<td>0.326666</td>
<td>36.899799</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>45</td>
<td>10</td>
<td>0.965109</td>
<td>44.999751</td>
<td>0.000000 error^2</td>
</tr>
</tbody>
</table>

0.000000 objective
Solution for Problem 12.11d

\[ y = a + bx^c \]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>input</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual</td>
<td>3</td>
<td>0.9</td>
<td>2</td>
<td>solved</td>
</tr>
<tr>
<td>estimated</td>
<td>2.999999105</td>
<td>0.89996872</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>x</th>
<th>noise</th>
<th>yest</th>
<th>error^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>-10</td>
<td>0.224329</td>
<td>92.999686</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>75.9</td>
<td>-9</td>
<td>0.758325</td>
<td>75.899746</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>60.6</td>
<td>-8</td>
<td>0.266314</td>
<td>60.599799</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>47.1</td>
<td>-7</td>
<td>0.630058</td>
<td>47.099846</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>35.4</td>
<td>-6</td>
<td>0.377891</td>
<td>35.399886</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>25.5</td>
<td>-5</td>
<td>0.986692</td>
<td>25.499921</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>17.4</td>
<td>-4</td>
<td>0.720524</td>
<td>17.399949</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>11.1</td>
<td>-3</td>
<td>0.188339</td>
<td>11.099971</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>6.6</td>
<td>-2</td>
<td>0.523004</td>
<td>6.599987</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>3.9</td>
<td>-1</td>
<td>0.671346</td>
<td>3.899996</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.972055</td>
<td>2.999999</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>3.9</td>
<td>1</td>
<td>0.703574</td>
<td>3.899996</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>6.6</td>
<td>2</td>
<td>0.661743</td>
<td>6.599987</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>11.1</td>
<td>3</td>
<td>0.121933</td>
<td>11.099971</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>17.4</td>
<td>4</td>
<td>0.981325</td>
<td>17.399949</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>25.5</td>
<td>5</td>
<td>0.563668</td>
<td>25.499921</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>35.4</td>
<td>6</td>
<td>0.896415</td>
<td>35.399886</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>47.1</td>
<td>7</td>
<td>0.367010</td>
<td>47.099846</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>60.6</td>
<td>8</td>
<td>0.043336</td>
<td>60.599799</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>75.9</td>
<td>9</td>
<td>0.335745</td>
<td>75.899746</td>
<td>0.000000 error^2</td>
</tr>
<tr>
<td>93</td>
<td>10</td>
<td>0.789912</td>
<td>92.999686</td>
<td>0.000000 error^2</td>
</tr>
</tbody>
</table>

0.00001 objective
### Solution for Problem 12.11e

\[
y = \frac{\sum_{i=1}^{n+1} (p_i x^{n+1-i})}{x^n + \sum_{i=1}^{m} (q_i x^{m-i})} = \frac{p_1 x^2 + p_2 x^1 + p_3 x^0}{x^2 + q_2 x^1 + q_3 x^0}
\]

<table>
<thead>
<tr>
<th></th>
<th>(n=2)</th>
<th>(p(1))</th>
<th>(p(2))</th>
<th>(p(3))</th>
<th>(m=2)</th>
<th>(q(1))</th>
<th>(q(2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual</td>
<td></td>
<td>0.5</td>
<td>0.3</td>
<td>0.4</td>
<td></td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>estimated</td>
<td>0.50007667</td>
<td>0.200028501</td>
<td>0.500093033</td>
<td>0.600021961</td>
<td>0.400842148</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(y)</th>
<th>(x)</th>
<th>noise</th>
<th>yest</th>
<th>error^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.45508475</td>
<td>-10</td>
<td>0.363874286</td>
<td>0.455573814</td>
<td>4.26926E-09</td>
</tr>
<tr>
<td>0.45</td>
<td>-9</td>
<td>0.51757403</td>
<td>0.450063505</td>
<td>4.03293E-09</td>
</tr>
<tr>
<td>0.44295302</td>
<td>-8</td>
<td>0.778879083</td>
<td>0.443014083</td>
<td>3.72871E-09</td>
</tr>
<tr>
<td>0.433628319</td>
<td>-7</td>
<td>0.622012259</td>
<td>0.433686007</td>
<td>3.32801E-09</td>
</tr>
<tr>
<td>0.420731707</td>
<td>-6</td>
<td>0.5371512</td>
<td>0.420784512</td>
<td>2.78836E-09</td>
</tr>
<tr>
<td>0.401785714</td>
<td>-5</td>
<td>0.596801376</td>
<td>0.401831018</td>
<td>2.05243E-09</td>
</tr>
<tr>
<td>0.371428571</td>
<td>-4</td>
<td>0.804356778</td>
<td>0.371461456</td>
<td>1.08143E-09</td>
</tr>
<tr>
<td>0.315789474</td>
<td>-3</td>
<td>0.42959621</td>
<td>0.315800037</td>
<td>1.11592E-10</td>
</tr>
<tr>
<td>0.1875</td>
<td>-2</td>
<td>0.223982569</td>
<td>0.187473114</td>
<td>7.22832E-10</td>
</tr>
<tr>
<td>-0.25</td>
<td>-1</td>
<td>0.074695636</td>
<td>-0.249799977</td>
<td>4.00093E-08</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.691845287</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.6</td>
<td>1</td>
<td>0.484135129</td>
<td>0.599839939</td>
<td>2.56196E-08</td>
</tr>
<tr>
<td>0.607142857</td>
<td>2</td>
<td>0.014178992</td>
<td>0.60714496</td>
<td>4.42244E-12</td>
</tr>
<tr>
<td>0.589285714</td>
<td>3</td>
<td>0.603577643</td>
<td>0.589332098</td>
<td>2.15149E-09</td>
</tr>
<tr>
<td>0.574468085</td>
<td>4</td>
<td>0.29171364</td>
<td>0.574530774</td>
<td>3.9299E-09</td>
</tr>
<tr>
<td>0.563380282</td>
<td>5</td>
<td>0.793243388</td>
<td>0.563450283</td>
<td>4.90023E-09</td>
</tr>
<tr>
<td>0.555</td>
<td>6</td>
<td>0.633755923</td>
<td>0.555073718</td>
<td>5.43441E-09</td>
</tr>
<tr>
<td>0.548507463</td>
<td>7</td>
<td>0.372130781</td>
<td>0.548583232</td>
<td>5.74105E-09</td>
</tr>
<tr>
<td>0.543352601</td>
<td>8</td>
<td>0.570979431</td>
<td>0.543429567</td>
<td>5.92375E-09</td>
</tr>
<tr>
<td>0.539170507</td>
<td>9</td>
<td>0.542946033</td>
<td>0.539248196</td>
<td>6.03554E-09</td>
</tr>
<tr>
<td>0.535714286</td>
<td>10</td>
<td>0.979146306</td>
<td>0.53579242</td>
<td>6.10495E-09</td>
</tr>
</tbody>
</table>

1.2797E-07 objective
Solution for Problem 12.11f

\[ y = \sum_{i=1}^{n} \left( a_i \sin(b_i x + c_i) \right) \]

\[
\begin{array}{ccccccc}
\text{n=2} & \text{a(1)} & \text{b(1)} & \text{c(1)} & \text{a(2)} & \text{b(2)} & \text{c(2)} \\
\text{actual} & 0.5 & 0.6 & 0.3 & 0.4 & 0.2 & 0.1 \\
\text{estimated} & 0.500074 & 0.599922 & 0.299868 & 0.399956 & 0.199831 & 0.099698 \\
\text{solved} & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccc}
\text{y} & \text{x} & \text{noise} & \text{yest} & \text{error}^2 \\
-0.10318 & -10 & 0.381266 & -0.103 & 3.07E-08 \\
0.066241 & -9 & 0.451856 & 0.0664 & 2.51E-08 \\
0.089767 & -8 & 0.38888 & 0.089862 & 8.98E-09 \\
-0.04154 & -7 & 0.783088 & -0.0415 & 1.39E-09 \\
-0.27761 & -6 & 0.831278 & -0.2776 & 1.92E-10 \\
-0.52702 & -5 & 0.552304 & -0.527 & 4.39E-10 \\
-0.68929 & -4 & 0.297772 & -0.6892 & 1.12E-09 \\
-0.69052 & -3 & 0.674304 & -0.6904 & 5.39E-10 \\
-0.50987 & -2 & 0.850913 & -0.5098 & 5.37E-10 \\
-0.18769 & -1 & 0.922309 & -0.1877 & 9.14E-09 \\
0.187693 & 0 & 0.041162 & 0.187528 & 2.73E-08 \\
0.509872 & 1 & 0.593157 & 0.509671 & 4E-08 \\
0.690518 & 2 & 0.111615 & 0.690336 & 3.31E-08 \\
0.689292 & 3 & 0.953504 & 0.689172 & 1.43E-08 \\
0.527021 & 4 & 0.257232 & 0.526975 & 2.05E-09 \\
0.27761 & 5 & 0.364648 & 0.277609 & 1.76E-12 \\
0.04154 & 6 & 0.233946 & 0.041523 & 2.79E-10 \\
-0.08977 & 7 & 0.684254 & -0.08985 & 7.69E-09 \\
-0.06624 & 8 & 0.384404 & -0.06641 & 2.93E-08 \\
0.103177 & 9 & 0.397023 & 0.102981 & 3.87E-08 \\
0.353691 & 10 & 0.108407 & 0.353598 & 8.54E-09 \\
& & & & & 2.8E-07 objective \\
\end{array}
\]
Solution for Problem 12.11g

\[ F(x, \mu, \sigma, p) = 1 - \exp \left[ - \left( \frac{x - \mu}{\sigma} \right)^p \right] \]

\[ f(x, \mu, \sigma, p) = \frac{p}{\sigma^p} (x - \mu)^{p-1} \exp \left[ - \left( \frac{x - \mu}{\sigma} \right)^p \right] \]

\[
\sum_{i=1}^{n} \left( 1 - \exp \left[ - \left( \frac{x - \mu}{\sigma} \right)^p \right] - \frac{i}{n+1} \right)^2
\]

\[ \mu = 0 \]

\[
\ln \left( \ln \left( \frac{1}{1 - F(x)} \right) \right) = p \ln(\sigma) - p \ln(x)
\]

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual</td>
<td>3</td>
<td>4.6</td>
</tr>
<tr>
<td>estimated</td>
<td>2.941</td>
<td>4.552</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>noise</th>
<th>yest</th>
<th>error^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.578169</td>
<td>0.509193</td>
<td>4.457282</td>
<td>0.014614</td>
</tr>
<tr>
<td>2</td>
<td>2.498727</td>
<td>0.468802</td>
<td>2.418736</td>
<td>0.006399</td>
</tr>
<tr>
<td>3</td>
<td>1.282332</td>
<td>0.172013</td>
<td>1.226263</td>
<td>0.003144</td>
</tr>
<tr>
<td>4</td>
<td>0.419286</td>
<td>0.094928</td>
<td>0.38019</td>
<td>0.001528</td>
</tr>
<tr>
<td>5</td>
<td>-0.25014</td>
<td>0.128877</td>
<td>-0.27608</td>
<td>0.000672</td>
</tr>
<tr>
<td>6</td>
<td>-0.79711</td>
<td>0.947664</td>
<td>-0.81228</td>
<td>0.00023</td>
</tr>
<tr>
<td>7</td>
<td>-1.25956</td>
<td>0.043785</td>
<td>-1.26564</td>
<td>3.69E-05</td>
</tr>
<tr>
<td>8</td>
<td>-1.66016</td>
<td>0.969334</td>
<td>-1.65836</td>
<td>3.24E-06</td>
</tr>
<tr>
<td>9</td>
<td>-2.0135</td>
<td>0.467871</td>
<td>-2.00476</td>
<td>7.65E-05</td>
</tr>
<tr>
<td>10</td>
<td>-2.32959</td>
<td>0.679038</td>
<td>-2.31462</td>
<td>0.000224</td>
</tr>
<tr>
<td>11</td>
<td>-2.61552</td>
<td>0.647041</td>
<td>-2.59493</td>
<td>0.000424</td>
</tr>
<tr>
<td>12</td>
<td>-2.87655</td>
<td>0.306954</td>
<td>-2.85083</td>
<td>0.000662</td>
</tr>
<tr>
<td>13</td>
<td>-3.11668</td>
<td>0.448394</td>
<td>-3.08623</td>
<td>0.000927</td>
</tr>
<tr>
<td>14</td>
<td>-3.339</td>
<td>0.363421</td>
<td>-3.30419</td>
<td>0.001212</td>
</tr>
<tr>
<td>15</td>
<td>-3.54598</td>
<td>0.715081</td>
<td>-3.50709</td>
<td>0.001512</td>
</tr>
</tbody>
</table>

0.031665
Solution to Problem 12.12

This is the Matrix Inversion Lemma that was so popular with the IBM 360: Matrixcookbook

And several others from the internet.


Problem 12. This is the Matrix Inversion Lemma that was so popular for updating the impedance matrix.

The Matrix Inversion Lemma is defined by the equation

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

Proof: We construct an augmented matrix \(A, B, C,\) and \(D\) and its inverse:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}^{-1} =
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix}
\]  \[(2)\]

We then construct the two products:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix} =
\begin{bmatrix}
AE + BG & AF + BH \\
CE + DG & CF + DH
\end{bmatrix} =
\begin{bmatrix}
I & 0 \\
0 & I
\end{bmatrix}
\]  \[(3)\]

And

\[
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix}
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix} =
\begin{bmatrix}
EA + FC & EB + FD \\
GA + HC & GB + HD
\end{bmatrix} =
\begin{bmatrix}
I & 0 \\
0 & I
\end{bmatrix}
\]  \[(4)\]

Sub-matrices in (3) and (4) are broken out to form eight matrix equations:

\[
AE + BG = I \]  \[(5)\]

\[
AF + BH = 0 \]  \[(6)\]

\[
CE + DG = 0 \]  \[(7)\]

\[
CF + DH = I \]  \[(8)\]

\[
EA + FC = I \]  \[(9)\]

\[
EB + FD = 0 \]  \[(10)\]

\[
GA + HC = 0 \]  \[(11)\]

\[
GB + HD = I \]  \[(12)\]

Combining (5) through (12) in various orders gives two sets of equations for \(E, F, G\) and \(H\) from \(A, B, C,\) and \(D,\) assuming that all of the involved inverse matrices exist.:

\[
E = (A - BD^{-1} C)^{-1} \]  \[(13)\]

\[
F = -(A - BD^{-1} C)^{-1} BD^{-1} \]  \[(14)\]

\[
G = -D^{-1} C (A - BD^{-1} C)^{-1} \]  \[(15)\]

\[
H = D^{-1} + D^{-1} C (A - BD^{-1} C)^{-1} BD^{-1} \]  \[(16)\]

And

\[
E = A^{-1} + A^{-1} B (D - CA^{-1} B)^{-1} CA^{-1} \]  \[(17)\]

\[
F = -A^{-1} B (D - CA^{-1} B)^{-1} \]  \[(18)\]

\[
G = -(D - CA^{-1} B)^{-1} DA^{-1} \]  \[(19)\]

\[
H = (D - CA^{-1} B)^{-1} \]  \[(20)\]
Solution to Problem 12.12 continued

The proof is completed by combining either (13) and (17) or (16) and (20).

Alternatively, the following shows the IEEE PES traditional approach.

Lemma: Given conformable matrices P & Q:

\[(I + PQ)^{-1}P = P(I + QP)^{-1}\]

Derivation:

\[(I + P)^{-1}(I + P) = I\]

Provided \((I+P)\) is non-singular.

\[(I + P)^{-1} = I - (I + P)^{-1}P\]

Consider \((A+UBV)\), where:

Nonsingular \(A\), original matrix for which we have: \(A=LU\), \(A^{-1}b\)

Want: \((A+UBV)^{-1}b\)

\((A+UBV) = A(I + A^{-1}UBV)\)

Provided \(A\) is invertible, then by the lemma:

\[(A+UBV)^{-1} = (I + A^{-1}UBV)^{-1} A^{-1} = A^{-1} (I + UBVA^{-1})^{-1}\]

Setting:

\(P = A^{-1}UBV\)

Substituting:

\[(I + P)^{-1} = I - (I + P)^{-1}P\]

\[(A + UBV)^{-1} = (I + A^{-1}UBV)^{-1} A^{-1} = [I - (I + A^{-1}UBV)^{-1} A^{-1}UBV] A^{-1} = A^{-1} - (I + A^{-1}UBV)^{-1} A^{-1}UBVA^{-1} = A^{-1} - A^{-1}U(I+BV^{-1}U)^{-1}BVA^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1} *\]

\(B\) is assumed nonsingular.

Inverse must exist. Then

\[(A + UV)^{-1}b = A^{-1}b - A^{-1}U (B^{-1} + VA^{-1}U)^{-1}VA^{-1}b\]

Reference: The Matrix cookbook. Several others are found on the internet.

There is a Matrix Inversion Lemma website in connexions website: http://cnx.org/.