CH5115 : Parameter and State Estimation (Jul-Nov 2020) Assignment 2 - Solutions

INDIAN INSTITUTE OF TECHNOLOGY MADRAS
Department of Chemical Engineering

Mark Distribution

	Question 1	Question 2	Question 3	Question 4
(a)	10	20	5	10
(b)	10	10	15	20

Question 1

Part a

Random process:

$$v[k] = Acos^2(2\pi fk + \phi)$$

$\phi =$ constant

$A =$ Random variable such that $E(A) = 0$, $var(A) = 1$

$E(A^2) = var(A) + E(A)^2 = var(A) = 1$

$$\begin{aligned}
\sigma_{vv}[l] &= E(v[k]v[k-l]) \\
\sigma_{vv}[l] &= E(A^2cos^2(2\pi fk + \phi)cos^2(2\pi f(k-l) + \phi)) \\
\sigma_{vv}[l] &= E[\frac{A^2}{4}(cos(4\pi fk + 2\phi) + 1)(cos(4\pi f(k-l) + 2\phi) + 1)] \\
\sigma_{vv}[l] &= \frac{1}{4}(cos(4\pi fk + 2\phi) + 1)(cos(4\pi f(k-l) + 2\phi) + 1)
\end{aligned}$$

Autocovariance is time (lag) dependent. Therefore, $v[k]$ is a non-stationary process.

Part b

$$\begin{aligned}
v[k] &= v[k-1] + e[k] \\
var\,(v[k]) &= var\,(v[k-1] + e[k]) \\
var\,(v[k]) &= var\,(v[k-1]) + var\,(e[k]) \\
var\,(v[k]) &= var\,(v[k-2] + e[k-1]) + \sigma_e^2 \\
var\,(v[k]) &= var\,(v[k-2]) + var\,(e[k-1]) + \sigma_e^2 \\
var\,(v[k]) &= var\,(v[k-2]) + 2\sigma_e^2 \\
&\vdots \\
var\,(v[k]) &= var\,(v[0]) + k\sigma_e^2
\end{aligned}$$

Since, variance of $v[k]$ depends on time k, the process is variance non-stationary. Using MATLAB to test this numerically

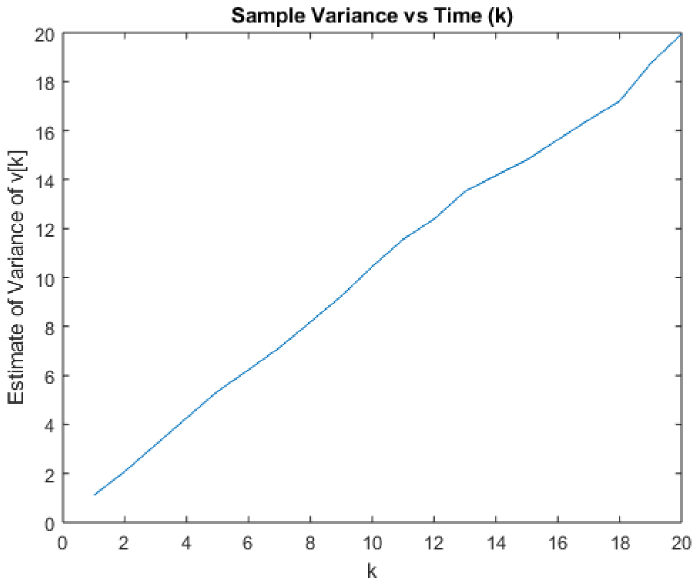


Figure 1: Variance Non-Stationarity in Random Walk

```
clc
close all
clear
var_v = [];
sigma_e = 1;         % Standard deviation of e
v = zeros(1000, 1); % 1000 realizations of v[0]

% Calculate var(v[k]) for k = 1 to 20
for k=1:20
    % Random walk v[k] = v[k-1] + e[k] (1000 realizations)
    v = v + normrnd(0, sigma_e, 1000, 1);

    % Append sample variance of v[k]
    var_v = [var_v; var(v)];
end

figure
plot(var_v)
title('Sample Variance vs Time (k)');
xlabel('k')
ylabel('Estimate of Variance of v[k]')
```

Question 2

Part a

$$
\begin{aligned}
y[k] &= \frac{b_2^0 q^{-2}}{1+f_1^0 q^{-1}} u[k] + e[k] \\
&= b_2^0 u[k-2] - f_1^0 y[k-1] + e[k] + f_1^0 e[k-1] \\
\sigma_{yy} &= E\left(y[k]y[k]\right) \\
&= E\left(\left(b_2^0 u[k-2] - f_1^0 y[k-1] + e[k] + f_1^0 e[k-1]\right)^2\right) \\
&= E[(b_2^0)^2 u[k-2]^2 + (f_1^0)^2 y[k-1]^2 + e[k]^2 + (f_1^0)^2 e[k-1]^2 - 2b_2^0 f_1^0 u[k-2]y[k-1] + 2b_2^0 u[k-2]e[k] \\
&+ 2b_2^0 f_1^0 u[k-2]e[k-1] - 2f_1^0 y[k-1]e[k] - 2(f_1^0)^2 y[k-1]e[k-1] + 2f_1^0 e[k]e[k-1]] \\
&= (b_2^0)^2 \sigma_{uu}[0] + (f_1^0)^2 \sigma_{yy}[0] + \left(1+(f_1^0)^2\right)\sigma_{ee}[0] + -2b_2^0 f_1^0 \sigma_{yu}[1] + 2b_2^0 \sigma_{eu}[2] + 2b_2^0 f_1^0 \sigma_{eu}[1] \\
&- 2f_1^0 \sigma_{ey}[1] - 2(f_1^0)^2 \sigma_{ye}[0] + 2f_1^0 \sigma_{ee}[1] \\
\sigma_{yu}[l] &= 0;\ \forall l < 2 \text{ (Two sample delay between u and y)}
\end{aligned}
$$

Therefore, $\sigma_{yu}[1] = 0$

$$
\begin{aligned}
\sigma_{eu}[l] &= 0\ \ \forall l \text{ (given)} \\
\sigma_{ey}[l] &= 0\ \ \forall l > 0 \text{ (Future } e \text{ don't affect past } y\text{)} \\
\sigma_{ee}[l]\ , \sigma_{uu}[l] &= 0\ \ \forall l \neq 0 \text{ (White noise property)} \\
E\left(y[k]e[k]\right) &= E\left(b_2^0 u[k-2]e[k] - f_1^0 y[k-1]e[k] + e[k]^2 + f_1^0 e[k-1]e[k]\right) \\
E\left(y[k]e[k]\right) &= b_2^0 \sigma_{eu}[2] - f_1^0 \sigma_{ey}[1] + \sigma_e^2 + f_1^0 \sigma_{ee}[1] \\
\sigma_{ey}[0] &= \sigma_e^2
\end{aligned}
$$

Substituting in the expression for σ_{yy}

$$
\begin{aligned}
\sigma_y^2 &= (b_2^0)^2 \sigma_u^2 + (f_1^0)^2 \sigma_y^2 + \left(1+(f_1^0)^2\right)\sigma_e^2 - 2(f_1^0)^2 \sigma_e^2 \\
\sigma_y^2 &= \frac{(b_2^0)^2}{1-(f_1^0)^2}\sigma_u^2 + \sigma_e^2
\end{aligned}
$$

To compute $\sigma_{yu}[2]$

$$
\begin{aligned}
\sigma_{yu}[l] &= E(y[k]u[k-l]) \\
y[k]u[k-2] &= b_2^o u[k-2]^2 - f_1^o y[k-1]u[k-2] + e[k]u[k-2] \\
E(y[k]u[k-2]) &= b_2^o E\left(u[k-2]^2\right) - f_1^o E(y[k-1]u[k-2]) + E(e[k]u[k-2]) \\
\sigma_{yu}[2] &= b_2^0 \sigma_u^2 - f_1^0 \sigma_{yu}[1] + \sigma_{eu}[2] \\
\sigma_{yu}[2] &= b_2^0 \sigma_u^2
\end{aligned}
$$

To compute $\sigma_{yy}[1]$

$$
\begin{aligned}
\sigma_{yy}[1] &= E\left(y[k]y[k-1]\right) \\
y[k]y[k-1] &= b_2^0 u[k-2]y[k-1] - f_1^0 y[k-1]^2 + e[k]y[k-1] + f_1^0 e[k-1]y[k-1] \\
\sigma_{yy}[1] &= b_2^0 \sigma_{yu}[1] - f_1^0 \sigma_y^2 + \sigma_{ey}[1] + f_1^0 \sigma_{ey}[0] \\
&= -f_1^0 \frac{(b_2^0)^2}{1-(f_1^0)^2}\sigma_u^2
\end{aligned}
$$

Part b

The MATLAB code to numerically verify the above:

```matlab
clc;
clear;
close all;

N = 500;
sigma2_u = 2;
SNR = 10;
b20 = 2;
f10 = 0.5;

% Generating y_star using filter on u
b = [0 0 b20];
a = [1 f10];
u = normrnd(0, sigma2_u^0.5, N, 1);
y_star = filter(b, a, u);

% Calculating sigma2_e based on SNR
sigma2_y_star = var(y_star);
sigma2_e = sigma2_y_star/SNR;
disp(['Sigma^2 e = ', num2str(sigma2_e)]);

% Simulating y from y_star and e
e = normrnd(0, sigma2_e^0.5, N, 1);
y = y_star + e;

% Numerical Estimates
sigma2_y = var(y);
acf = xcov(y, 10, 'unbiased');
sigma_yy1 = acf(12);
ccf = xcov(y, u, 10, 'unbiased');
sigma_yu1 = ccf(12);
sigma_yu2 = ccf(13);

% Theoretical Values
Tsigma2_y = b20^2/(1-f10^2)*sigma2_u + sigma2_e;
Tsigma_yy1 = -f10*b20^2/(1-f10^2)*sigma2_u;
Tsigma_yu1 = 0;
Tsigma_yu2 = b20*sigma2_u;

disp(['Sigma^2 y: Theoretical = ', num2str(Tsigma2_y), ' | Numerical = ', num2str(
    sigma2_y)]);
disp(['Sigma_yy[1]: Theoretical = ', num2str(Tsigma_yy1), ' | Numerical = ',
    num2str(sigma_yy1)]);
disp(['Sigma_yu[1]: Theoretical = ', num2str(Tsigma_yu1), ' | Numerical = ',
    num2str(sigma_yu1)]);
disp(['Sigma_yu[2]: Theoretical = ', num2str(Tsigma_yu2), ' | Numerical = ',
    num2str(sigma_yu2)]);
```

The output produced for a sample run was

```
Sigma^2 e = 1.0076
Sigma^2 y  : Theoretical = 11.6743 | Numerical = 11.7059
Sigma_yy[1]: Theoretical = -5.3333 | Numerical = -5.2591
Sigma_yu[1]: Theoretical = 0       | Numerical = -0.049482
Sigma_yu[2]: Theoretical = 4       | Numerical = 3.9251
```

Question 3

```
close all
clear
load('a2_q3.mat')
```

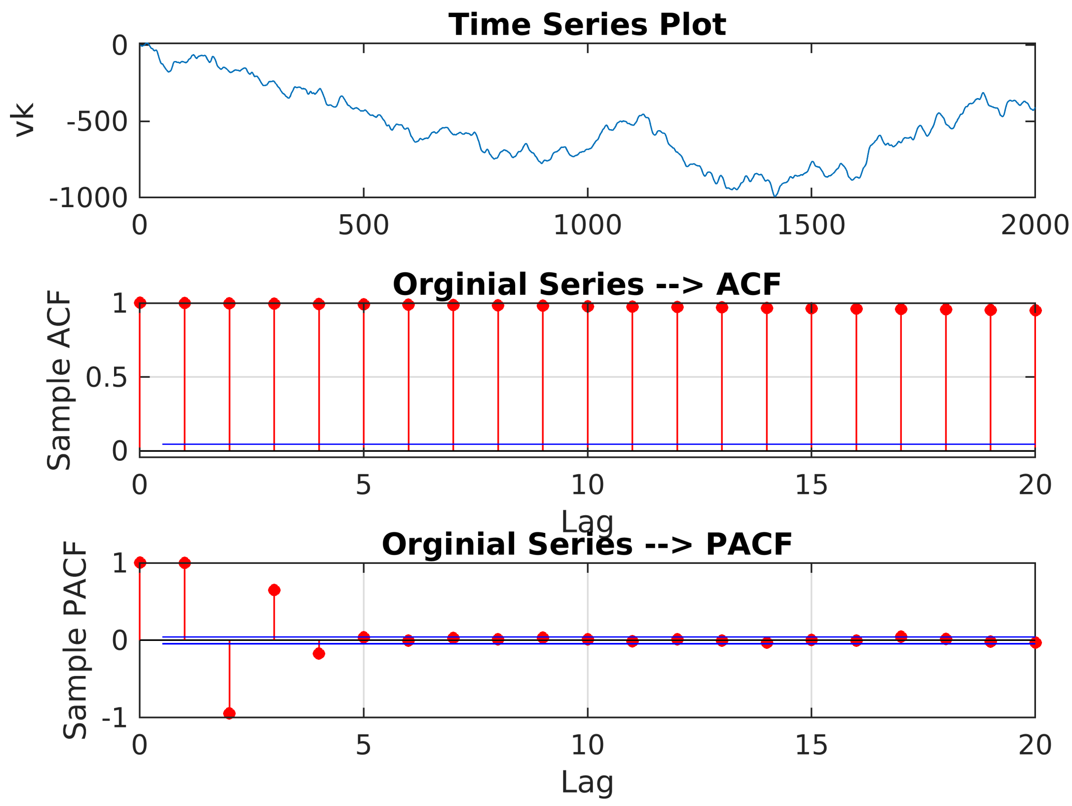
3.a) Check for integrating effects

Visual Inference

```
figure
subplot(3, 1, 1)
plot(vk)
title('Time Series Plot')
ylabel('vk')

subplot(3, 1, 2)
autocorr(vk)
title('Orginial Series --> ACF')
ylabel('Sample ACF')

subplot(3, 1, 3)
parcorr(vk)
title('Orginial Series --> PACF')
ylabel('Sample PACF')
```



Very slowly decaying ACF indicates the presence of integrating effect.

PACF at lag 1 is close to unity indicating the presence of a unit root.

Perform ADF Test to verify unit root presence

```
[~, pValue] = adftest(vk);
alpha = 0.05;

disp('ADF Test on series:')
```

```
ADF Test on series:
```

```
if pValue < alpha
    disp('Null Hypothesis rejected: Unit root not present')
else
    disp('Failed to reject Null Hypothesis: Unit root might be present')
end
```

```
Failed to reject Null Hypothesis: Unit root might be present
```

3.b) Fit a model

Determine order of integrating effect

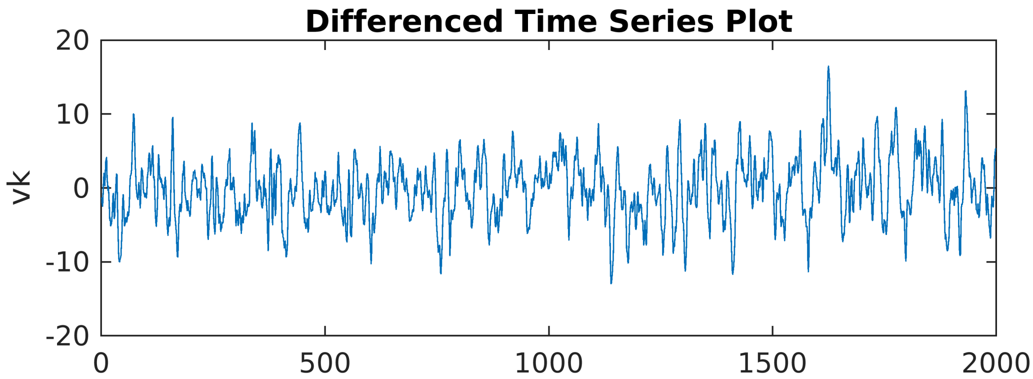
Difference the time series once

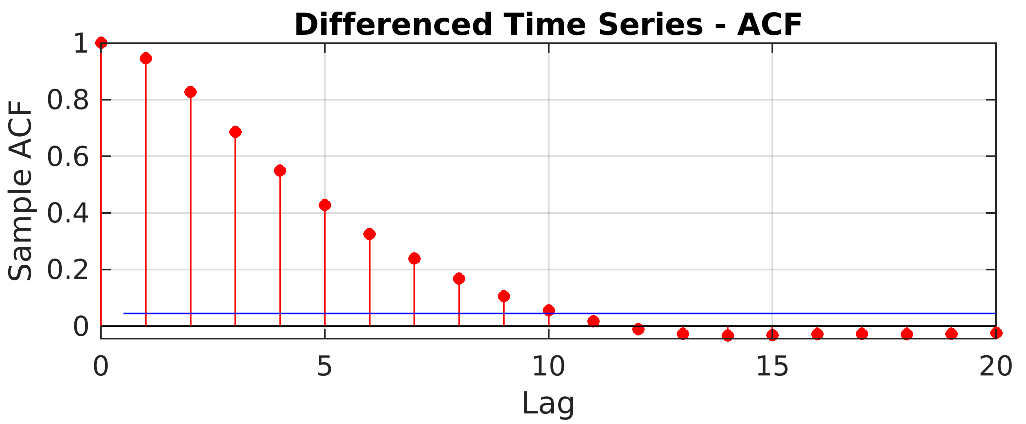
```
vk_diff = diff(vk);
```

Visual Inference

```
figure
subplot(2, 1, 1)
plot(vk_diff)
title('Differenced Time Series Plot')
ylabel('vk')

subplot(2, 1, 2)
autocorr(vk_diff)
title('Differenced Time Series - ACF')
ylabel('Sample ACF')
```





ACF dies down. Integrating effects might be removed.

Confirm using ADF test

```
[~, pValue] = adftest(vk_diff);
alpha = 0.05;

disp('ADF Test on differenced series')
```

```
ADF Test on differenced series
```

```
if pValue < alpha
    disp('Null Hypothesis rejected: Unit root not present')
else
    disp('Failed to reject Null Hypothesis. Unit root might be present')
end
```

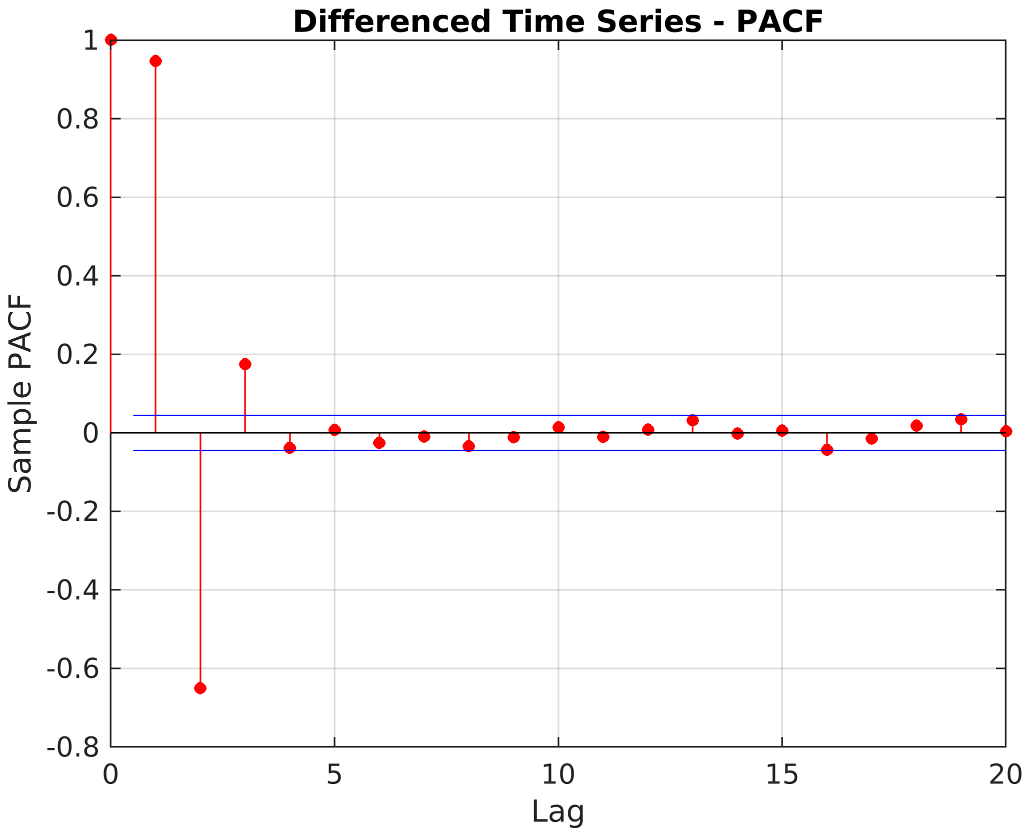
```
Null Hypothesis rejected: Unit root not present
```

Therefore, integrating effect order = 1

Finding model orders for the difference series

Plot PACF to check AR order

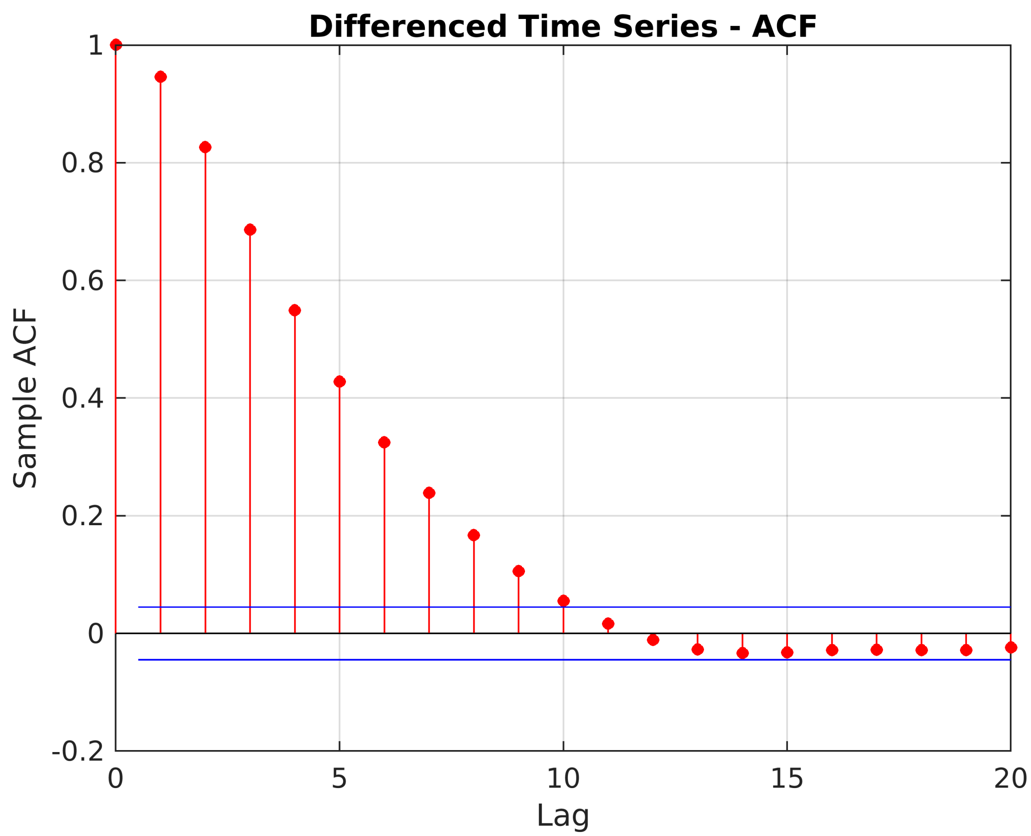
```
figure;
parcorr(vk_diff)
title('Differenced Time Series - PACF')
ylabel('Sample PACF')
```



AR order 3 observed

Plot ACF to check MA order

```
figure;
autocorr(vk_diff)
title('Differenced Time Series - ACF')
ylabel('Sample ACF')
```



MA order 9 observed

Testing out different models

ARIMA(3, 1, 0)

Since AR(3) has a better parsimony than MA(9), we start with AR(3) on the differenced data

```
mod_arma30 = arima(3,0,0);
mod_arma30est = estimate(mod_arma30, vk_diff);
```

```
ARIMA(3,0,0) Model (Gaussian Distribution):
```

	Value	StandardError	TStatistic	PValue
Constant	-0.01367	0.022482	-0.60805	0.54316
AR{1}	1.6748	0.022404	74.754	0
AR{2}	-0.92222	0.038405	-24.013	2.0388e-127
AR{3}	0.17419	0.02181	7.9867	1.3865e-15
Variance	1.0076	0.030858	32.652	7.4083e-234

p-value for constant is high --> Set to 0 and fit

```
mod_arma30.Constant = 0;
mod_arma30est = estimate(mod_arma30, vk_diff);
```

```
ARIMA(3,0,0) Model (Gaussian Distribution):
```

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
AR{1}	1.6751	0.022411	74.741	0
AR{2}	-0.9225	0.038407	-24.019	1.7467e-127
AR{3}	0.17443	0.021801	8.0011	1.2329e-15
Variance	1.0078	0.03086	32.656	6.6054e-234

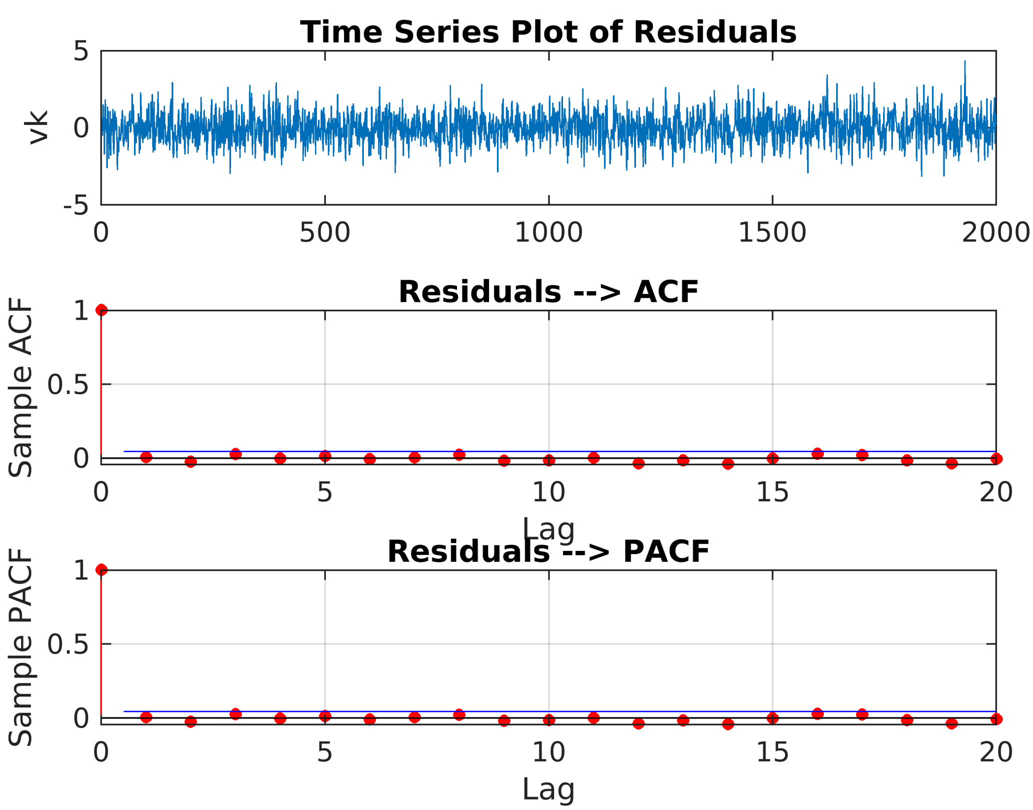
Null hypothesis (parameter=0) rejected for all parameters. Therefore no overfit present.

Residual Analysis (Checking underfit)

```
[res_arma30,~,logL] = infer(mod_arma30est, vk_diff);
figure
subplot(3, 1, 1)
plot(res_arma30)
title('Time Series Plot of Residuals')
ylabel('vk')

subplot(3, 1, 2)
autocorr(res_arma30)
title('Residuals --> ACF')
ylabel('Sample ACF')

subplot(3, 1, 3)
parcorr(res_arma30)
title('Residuals --> PACF')
ylabel('Sample PACF')
```



ACF and PACF indicate whiteness (no significant correlations at any lag). We confirm whiteness of residuals using the Ljung-Box test

```
[ht_resarma30, pval] = lbqtest(res_arma30)
```

```
ht_resarma30 = logical
   0
pval = 0.6090
```

```
if pval < alpha
    disp('Null Hypothesis rejected: Some correlations are significant (Series not white
else
    disp('Failed to reject Null Hypothesis: Series might be white')
end
```

```
Failed to reject Null Hypothesis: Series might be white
```

Thus, the model does not underfit the data.

To verify goodness we also try ARIMA(1,1,1) and ARMA(2,1,1)

ARIMA(1, 1, 1)

This has a lower parsimony than ARIMA(3,1,0)

```
mod_arma11 = arima(1,0,1);
mod_arma11est = estimate(mod_arma11, vk_diff);
```

ARIMA(1,0,1) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	-0.014795	0.038653	-0.38275	0.7019
AR{1}	0.91346	0.0090158	101.32	0
MA{1}	0.62324	0.018035	34.557	1.1034e-261
Variance	1.1305	0.033626	33.62	8.5902e-248

p-value for constant is high --> Set to 0 and fit

```
mod_arma11.Constant = 0;
mod_arma11est = estimate(mod_arma11, vk_diff);
```

ARIMA(1,0,1) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
AR{1}	0.91364	0.0090131	101.37	0
MA{1}	0.6232	0.018022	34.579	5.2203e-262
Variance	1.1306	0.033631	33.617	9.3518e-248

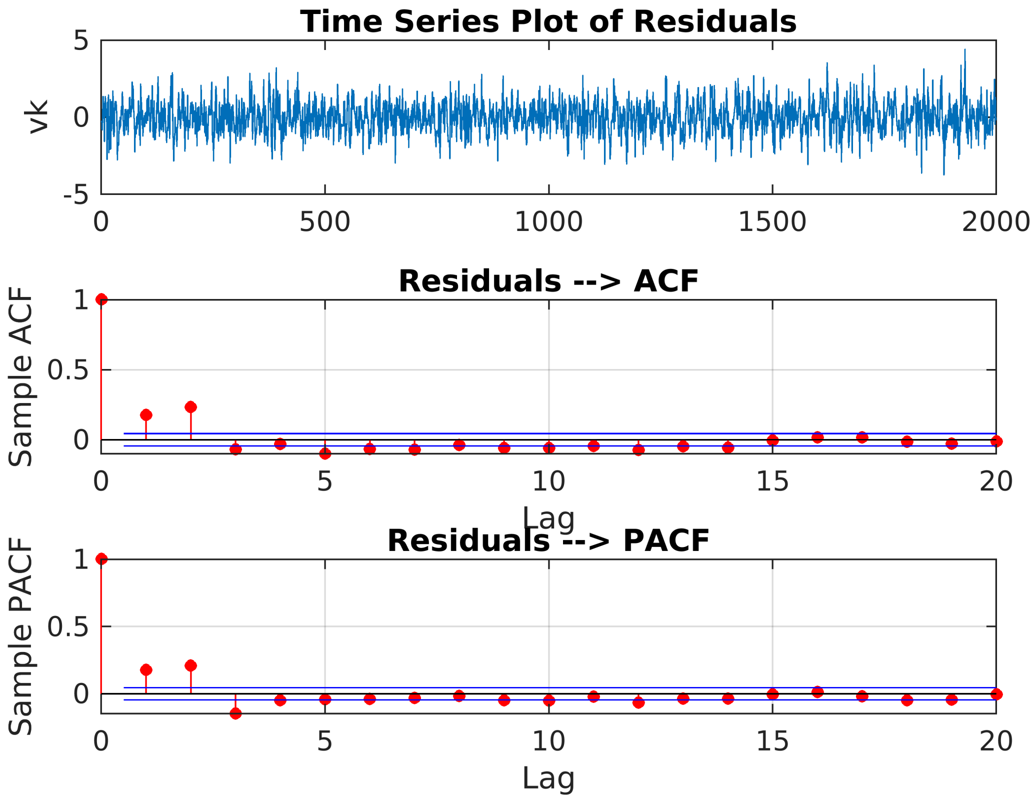
Null hypothesis (parameter=0) rejected for all parameters. Therefore no overfit present.

Residual Analysis (Checking underfit)

```
[res_arma11,~,logL] = infer(mod_arma11est, vk_diff);
figure
subplot(3, 1, 1)
plot(res_arma11)
title('Time Series Plot of Residuals')
ylabel('vk')

subplot(3, 1, 2)
autocorr(res_arma11)
title('Residuals --> ACF')
ylabel('Sample ACF')

subplot(3, 1, 3)
parcorr(res_arma11)
title('Residuals --> PACF')
ylabel('Sample PACF')
```



ACF and PACF do not indicate whiteness (significant correlations are present at small lags). We also test this using the Ljung-Box test.

```
[ht_resarma11, pval] = lbqtest(res_arma11)
```

```
ht_resarma11 = logical
   1
pval = 0
```

```
if pval < alpha
    disp('Null Hypothesis rejected: Some correlations are significant (Series not white
```

```
else
    disp('Failed to reject Null Hypothesis: Series might be white')
end
```

```
Null Hypothesis rejected: Some correlations are significant (Series not white)
```

Thus, the model underfits the data. The model is not complex enough.

ARIMA(2, 1, 1)

This model has the same number of parameters as ARIMA(3,1,0)

```
mod_arma21 = arima(2,0,1);
mod_arma21est = estimate(mod_arma21, vk_diff);
```

ARIMA(2,0,1) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	-0.018296	0.028552	-0.64077	0.52167
AR{1}	1.4104	0.028476	49.531	0
AR{2}	-0.50723	0.028004	-18.113	2.5372e-73
MA{1}	0.27084	0.031775	8.5239	1.543e-17
Variance	1.006	0.030744	32.723	7.3548e-235

p-value for constant is high --> Set to 0 and fit

```
mod_arma21.Constant = 0;
mod_arma21est = estimate(mod_arma21, vk_diff);
```

ARIMA(2,0,1) Model (Gaussian Distribution):

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
AR{1}	1.4104	0.028474	49.532	0
AR{2}	-0.50695	0.028009	-18.099	3.2211e-73
MA{1}	0.27108	0.031765	8.5337	1.4173e-17
Variance	1.0062	0.03075	32.723	7.4176e-235

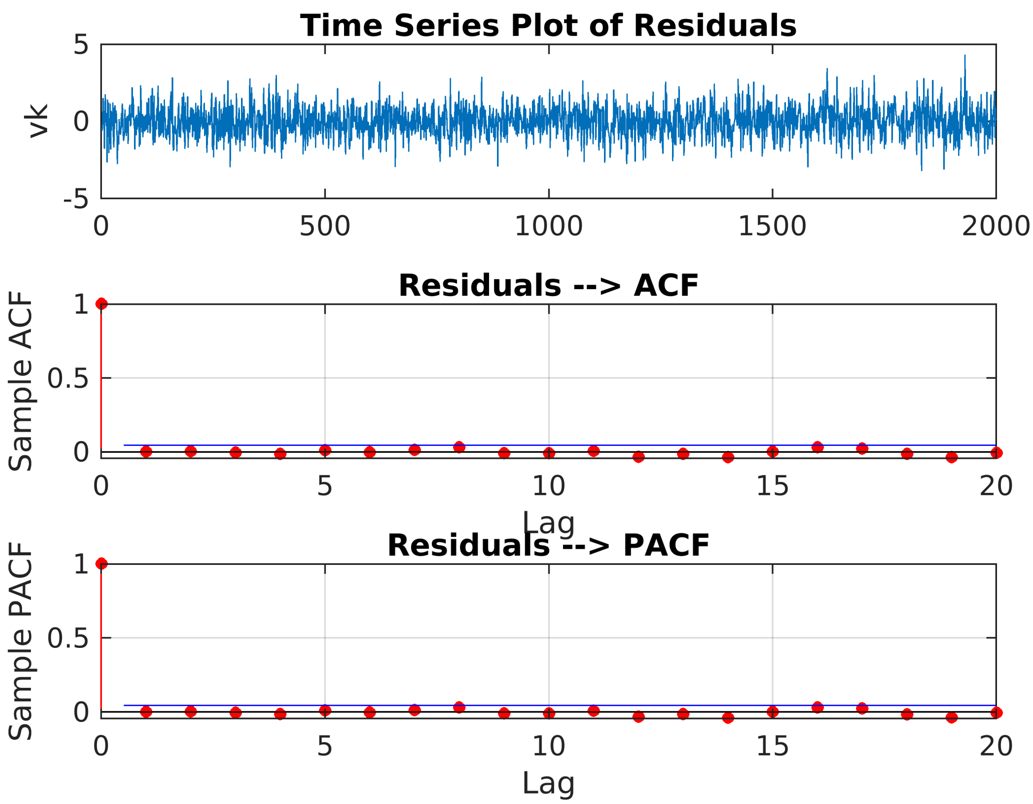
Null hypothesis (parameter=0) rejected for all parameters. Therefore no overfit present.

Residual Analysis (Checking underfit)

```
[res_arma21,~,logL] = infer(mod_arma21est, vk_diff);
figure
subplot(3, 1, 1)
plot(res_arma21)
title('Time Series Plot of Residuals')
ylabel('vk')

subplot(3, 1, 2)
autocorr(res_arma21)
title('Residuals --> ACF')
ylabel('Sample ACF')
```

```
subplot(3, 1, 3)
parcorr(res_arma21)
title('Residuals --> PACF')
ylabel('Sample PACF')
```



ACF and PACF indicate whiteness (no significant correlations at any lag). We confirm whiteness of residuals using the Ljung-Box test

```
[ht_resarma21, pval] = lbqtest(res_arma21)
```

```
ht_resarma21 = logical
   0
pval = 0.7695
```

```
if pval < alpha
    disp('Null Hypothesis rejected: Some correlations are significant (Series not white
else
    disp('Failed to reject Null Hypothesis: Series might be white')
end
```

```
Failed to reject Null Hypothesis: Series might be white
```

Thus, the model does not underfit the data.

Choosing the final model using Information Criteria

```
summarize(mod_arma30est)
```

```
   ARIMA(3,0,0) Model (Gaussian Distribution)
```

```
Effective Sample Size: 1999
Number of Estimated Parameters: 4
LogLikelihood: -2844.2
AIC: 5696.4
BIC: 5718.8
```

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
AR{1}	1.6751	0.022411	74.741	0
AR{2}	-0.9225	0.038407	-24.019	1.7467e-127
AR{3}	0.17443	0.021801	8.0011	1.2329e-15
Variance	1.0078	0.03086	32.656	6.6054e-234

```
summarize(mod_arma21est)
```

ARIMA(2,0,1) Model (Gaussian Distribution)

```
Effective Sample Size: 1999
Number of Estimated Parameters: 4
LogLikelihood: -2842.66
AIC: 5693.32
BIC: 5715.73
```

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
AR{1}	1.4104	0.028474	49.532	0
AR{2}	-0.50695	0.028009	-18.099	3.2211e-73
MA{1}	0.27108	0.031765	8.5337	1.4173e-17
Variance	1.0062	0.03075	32.723	7.4176e-235

ARIMA(2,1,1) has a smaller AIC and BIC than ARIMA(3,1,0). Thus the final model used is

$$(1 + 0.2711q^{-1})e[k] = (1 - q^{-1})(1 - 1.4104q^{-1} + 0.50695q^{-2})v[k]$$

Question 3

Part a

Finding the log likelihood for N observations ($\ln f(\mathbf{y}_N \mid \mu)$):

$$L(\mu; \mathbf{y}_N) = \ln f(\mathbf{y}_N \mid \mu) = \ln \Pi_{k=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y[k]-\mu)^2}{2\sigma^2}\right)$$
$$= c - \frac{N}{2}\ln\sigma^2 - \frac{1}{2}\sum_{k=1}^{N}\frac{(y[k]-\mu)^2}{\sigma^2}$$

Since μ is constrained to be a non-negative value, we add the constraint to the likelihood equation:

$$L(\mu; \mathbf{y}_N) = \ln f(\mathbf{y}_N \mid \mu) = \ln \Pi_{k=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y[k]-\mu)^2}{2\sigma^2}\right) + \lambda(\mu - 0)$$

such that $\lambda(\mu - 0) = 0$

Solving the above constrained equation, we get:

$$\hat{\mu}_{\text{MLE}} = max(0, \frac{1}{N}\sum_{k=1}^{N} y[k])$$

Finding Score (by ignoring constraints):

$$S(\mu; \mathbf{y}_N) = \frac{\partial}{\partial\mu} L(\mu; \mathbf{y}_N) = \sum_{k=1}^{N}\frac{(y[k]-\mu)}{\sigma^2}$$

Finding Fisher's Information from Score:

$$I(\mu) = -E\left(\frac{\partial S}{\partial \mu}\right) = \frac{N}{\sigma^2}$$

Note: Fisher's information assumes that the parameters are unconstrained. Therefore, any knowledge of constraints cannot be incorporated while calculating the Score or FI.

Part b

Given that $Y = aX + b + \varepsilon$, we can write $y[k] = ax[k] + b + \varepsilon$.
Since $\varepsilon \sim \mathcal{N}\left(0, \sigma_e^2\right)$ and X is free of randomness, $y[k] \sim \mathcal{N}\left(ax[k] + b, \sigma_e^2\right)$

$$L(a, b; \mathbf{y}_N) = \ln f(\mathbf{y}_N \mid a, b) = \ln \Pi_{k=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_e} \exp\left(-\frac{(y[k]-(ax[k]+b))^2}{2\sigma_e^2}\right)$$
$$= c - \frac{N}{2}\ln\sigma_e^2 - \frac{1}{2}\sum_{k=1}^{N}\frac{(y[k]-(ax[k]+b))^2}{\sigma_e^2}$$

Finding Score for a, b:

$$S\left(a;\mathbf{y}_N\right) = \frac{\partial}{\partial a}L\left(a,b;\mathbf{y}_N\right) = \sum_{k=1}^{N}\frac{(y[k]-(ax[k]+b))x[k]}{\sigma_e^2}$$

$$S\left(b;\mathbf{y}_N\right) = \frac{\partial}{\partial b}L\left(a,b;\mathbf{y}_N\right) = \sum_{k=1}^{N}\frac{(y[k]-(ax[k]+b))}{\sigma_e^2}$$

Equating both scores to zero and solving the system of equations for a and b gives:

$$\hat{a}_{MLE} = \frac{N\sum_{k=1}^{N}y[k]x[k]-\sum_{k=1}^{N}x[k]\sum_{k=1}^{N}y[k]}{N\sum_{k=1}^{N}x[k]^2-(\sum_{k=1}^{N}x[k])^2}$$

$$\hat{b}_{MLE} = \frac{\sum_{k=1}^{N}y[k]-\hat{a}_{MLE}\sum_{k=1}^{N}x[k]}{N}$$

Finding Fisher's Information from Score:

$$I(a) = -E\left(\frac{\partial S}{\partial a}\right) = \frac{\sum_{k=1}^{N}x[k]^2}{\sigma_e^2}$$

$$I(b) = -E\left(\frac{\partial S}{\partial b}\right) = \frac{N}{\sigma_e^2}$$

The MATLAB code to numerically verify the MLE estimates:

```
%Generating Data
clc;
clear all;
N = 100;
a = 2;
b = 3;
e = randn(N,1);
xk = 2 + randn(N,1);
yk = a*xk + b + e;

% Analytical estimates of a and b(a_MLE, b_MLE) based on the derived
% expressions
a_MLE = (N*sum(yk.*xk) - sum(xk)*sum(yk))/(N*sum(xk.*xk) - sum(xk)^2);
b_MLE = (sum(yk)-a_MLE*sum(xk))/N;

llh = [];
arange = 0:0.1:5;
brange = 0:0.1:5;

for ai=arange
    l = [];
    for bi=brange
        l = [l, -0.5*sum((yk - (ai*xk + bi)).^2)];
    end
    llh = [llh; l];
end
% Finding a_hat, b_hat from LLH plot
llhmax = max(llh(:));
```

```
[a_hat_ind, b_hat_ind] = find(llh == llhmax);
a_hat = arange(a_hat_ind);
b_hat = brange(b_hat_ind);

[A,B] = meshgrid(arange, brange);
figure; surf(A, B, llh)
hold on; line([a, a], [0, 5], [-3000, 2000],'Color','red');
hold on; line([a_hat, a_hat], [0, 5], [-3000, 2000], 'Color','green');
hold on; line([0, 0.5], [b, b], [-3000, 2000], 'Color','blue');
hold on; line([0, 0.5], [b_hat, b_hat], [-3000, 2000], 'Color','black');
legend('Log likelihood', 'a (true value)', 'a (predicted)', 'b (true value)', 'b (
    predicted)')
xlabel('Parameter a')
ylabel('Parameter b')
zlabel('Log likelihood function')

disp(['a_hat: Theoretical = ', num2str(a_MLE), '| Numerical = ', num2str(a_hat)])
disp(['b_hat: Theoretical = ', num2str(b_MLE), '| Numerical = ', num2str(b_hat)])
```

Output:
a_hat: Theoretical = 2.0339— Numerical = 2
b_hat: Theoretical = 2.9178— Numerical = 3

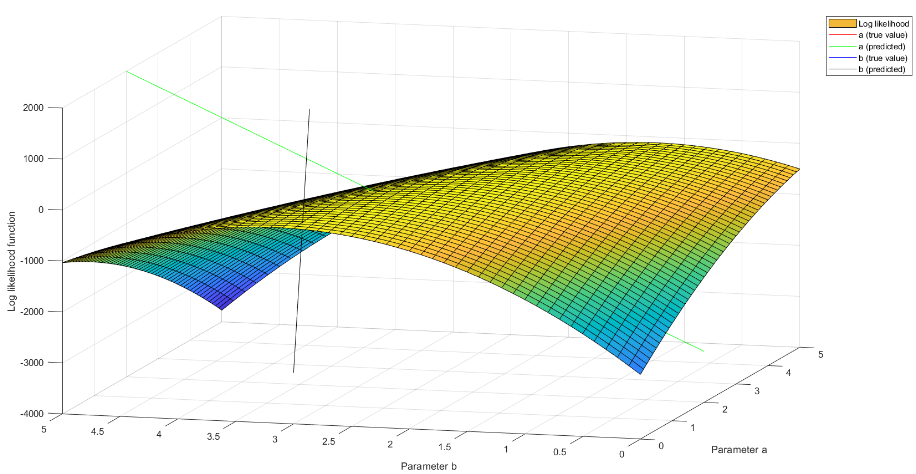


Figure 2: 3D plot of Log likelihood as a function of $\boldsymbol{a}$ and $\boldsymbol{b}$

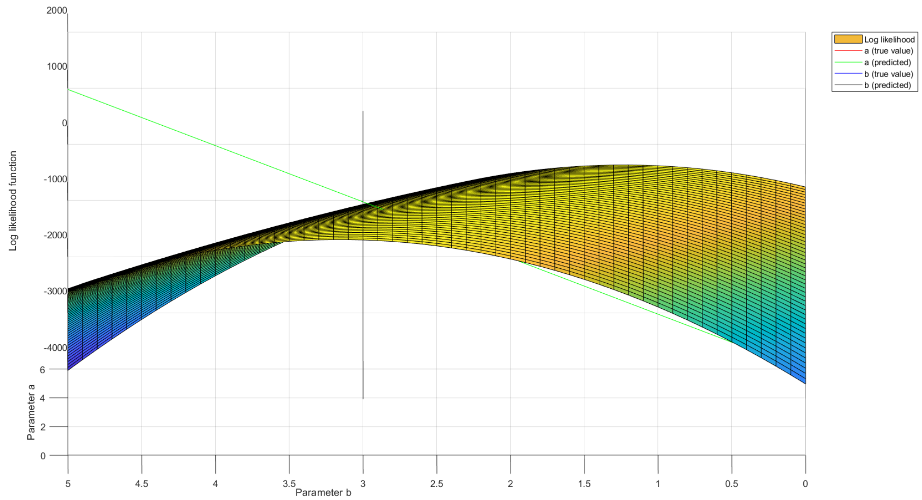


Figure 3: 3D plot of Log likelihood as a function of $\boldsymbol{a}$

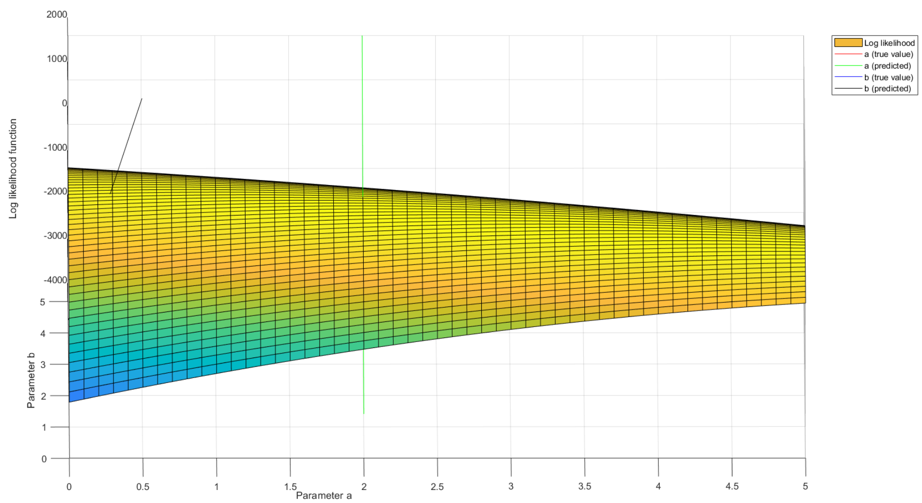


Figure 4: 3D plot of Log likelihood as a function of $\boldsymbol{b}$