

Introduction to Time Series and Forecasting



Time Series

Prof. Dr Taha Hussein Ali

Salahaddin University- Erbil, Iraq, College of Administration and Economics,
Department of Statistics and Informatics. Email: taha.ali@su.edu.krd

Time Series

Prof. Dr Taha Hussein Ali

Salahaddin University- Erbil, Iraq, College of Administration and Economics,
Department of Statistics and Informatics. **Email:** taha.ali@su.edu.krd

Time series analysis

Time series analysis refers to problems in which observations are collected at regular time intervals and there are correlations among successive observations. Applications cover virtually all areas of Statistics but some of the most important include economic and financial time series and many areas of environmental or ecological data.

In this course, I shall cover some of the most important methods for dealing with these problems. In the case of time series, these include the basic definitions of autocorrelations etc., then time-domain model fitting including autoregressive and moving average processes, spectral methods, and some discussion of the effect of time series correlations on other kinds of statistical inference, such as the estimation of means and regression coefficients.

Books

1. P.J. Brockwell and R.A. Davis, Time Series: Theory and Methods, Springer Series in Statistics (1986).
2. C. Chatfield, The Analysis of Time Series: Theory and Practice, Chapman and Hall (1975). Good general introduction, especially for those completely new to time series.
3. P.J. Diggle, Time Series: A Biostatistical Introduction, Oxford University Press (1990).
4. M. Kendall, Time Series, Charles Griffin (1976).

1 Models for time series

1.1 Time series data

A time series is a set of statistics, usually collected at regular intervals. Time series data occur naturally in many application areas.

- economics - e.g., monthly data for unemployment, hospital admissions, etc.
- finance - e.g., daily exchange rate, a share price, etc.
- environmental - e.g., daily rainfall, air quality readings.
- medicine - e.g., ECG brain wave activity every 2^{-8} secs.

The methods of time series analysis pre-date those for general stochastic processes and Markov Chains. The aims of time series analysis are to describe and summarise time series data, fit low-dimensional models, and make forecasts.

We write our real-valued series of observations as $\dots, X_{-2}, X_{-1}, X_0, X_1, X_2, \dots$, a doubly infinite sequence of real-valued random variables indexed by \mathbb{Z} .

1.2 Trend, seasonality, cycles and residuals

One simple method of describing a series is that of **classical decomposition**. The notion is that the series can be decomposed into four elements:

Trend (T_t) — long term movements in the mean;

Seasonal effects (I_t) — cyclical fluctuations related to the calendar;

Cycles (C_t) — other cyclical fluctuations (such as a business cycles);

Residuals (E_t) — other random or systematic fluctuations.

The idea is to create separate models for these four elements and then combine them, either additively

$$X_t = T_t + I_t + C_t + E_t$$

or multiplicatively

$$X_t = T_t \cdot I_t \cdot C_t \cdot E_t.$$

1.3 Stationary processes

1. A sequence $\{X_t, t \in \mathbb{Z}\}$ is **strongly stationary** or **strictly stationary** if

$$(X_{t_1}, \dots, X_{t_k}) \stackrel{D}{=} (X_{t_1+h}, \dots, X_{t_k+h})$$

for all sets of time points t_1, \dots, t_k and integer h .

2. A sequence is **weakly stationary**, or **second order stationary** if

- (a) $\mathbb{E}(X_t) = \mu$, and
- (b) $\text{cov}(X_t, X_{t+k}) = \gamma_k$,

where μ is constant and γ_k is independent of t .

- 3. The sequence $\{\gamma_k, k \in \mathbb{Z}\}$ is called the **autocovariance function**.
- 4. We also define

$$\rho_k = \gamma_k / \gamma_0 = \text{corr}(X_t, X_{t+k})$$

and call $\{\rho_k, k \in \mathbb{Z}\}$ the **autocorrelation function** (ACF).

Remarks.

- 1. A strictly stationary process is weakly stationary.
- 2. If the process is Gaussian, that is $(X_{t_1}, \dots, X_{t_k})$ is multivariate normal, for all t_1, \dots, t_k , then weak stationarity implies strong stationarity.
- 3. $\gamma_0 = \text{var}(X_t) > 0$, assuming X_t is genuinely random.
- 4. By symmetry, $\gamma_k = \gamma_{-k}$, for all k .

1.4 Autoregressive processes

The **autoregressive process** of order p is denoted $\text{AR}(p)$, and defined by

$$X_t = \sum_{r=1}^p \phi_r X_{t-r} + \epsilon_t \tag{1.1}$$

where ϕ_1, \dots, ϕ_r are fixed constants and $\{\epsilon_t\}$ is a sequence of independent (or uncorrelated) random variables with mean 0 and variance σ^2 .

The $\text{AR}(1)$ process is defined by

$$X_t = \phi_1 X_{t-1} + \epsilon_t. \tag{1.2}$$

To find its autocovariance function we make successive substitutions, to get

$$X_t = \epsilon_t + \phi_1(\epsilon_{t-1} + \phi_1(\epsilon_{t-2} + \dots)) = \epsilon_t + \phi_1\epsilon_{t-1} + \phi_1^2\epsilon_{t-2} + \dots$$

The fact that $\{X_t\}$ is second order stationary follows from the observation that $\mathbb{E}(X_t) = 0$ and that the autocovariance function can be calculated as follows:

$$\begin{aligned} \gamma_0 &= \mathbb{E} \left(\epsilon_t + \phi_1\epsilon_{t-1} + \phi_1^2\epsilon_{t-2} + \dots \right)^2 = (1 + \phi_1^2 + \phi_1^4 + \dots) \sigma^2 = \frac{\sigma^2}{1 - \phi_1^2} \\ \gamma_k &= \mathbb{E} \left(\sum_{r=0}^{\infty} \phi_1^r \epsilon_{t-r} \sum_{s=0}^{\infty} \phi_1^s \epsilon_{t+k-s} \right) = \frac{\sigma^2 \phi_1^k}{1 - \phi_1^2}. \end{aligned}$$

There is an easier way to obtain these results. Multiply equation (1.2) by X_{t-k} and take the expected value, to give

$$\mathbb{E}(X_t X_{t-k}) = \mathbb{E}(\phi_1 X_{t-1} X_{t-k}) + \mathbb{E}(\epsilon_t X_{t-k}).$$

Thus $\gamma_k = \phi_1 \gamma_{k-1}$, $k = 1, 2, \dots$

Similarly, squaring (1.2) and taking the expected value gives

$$\mathbb{E}(X_t^2) = \phi_1 \mathbb{E}(X_{t-1}^2) + 2\phi_1 \mathbb{E}(X_{t-1} \epsilon_t) + \mathbb{E}(\epsilon_t^2) = \phi_1^2 \mathbb{E}(X_{t-1}^2) + 0 + \sigma^2$$

and so $\gamma_0 = \sigma^2 / (1 - \phi_1^2)$.

More generally, the AR(p) process is defined as

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + \epsilon_t. \quad (1.3)$$

Again, the autocorrelation function can be found by multiplying (1.3) by X_{t-k} , taking the expected value and dividing by γ_0 , thus producing the **Yule-Walker equations**

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \dots + \phi_p \rho_{k-p}, \quad k = 1, 2, \dots$$

These are linear recurrence relations, with general solution of the form

$$\rho_k = C_1 \omega_1^{|k|} + \dots + C_p \omega_p^{|k|},$$

where $\omega_1, \dots, \omega_p$ are the roots of

$$\omega^p - \phi_1 \omega^{p-1} - \phi_2 \omega^{p-2} - \dots - \phi_p = 0$$

and C_1, \dots, C_p are determined by $\rho_0 = 1$ and the equations for $k = 1, \dots, p-1$. It is natural to require $\gamma_k \rightarrow 0$ as $k \rightarrow \infty$, in which case the roots must lie inside the unit circle, that is, $|\omega_i| < 1$. Thus there is a restriction on the values of ϕ_1, \dots, ϕ_p that can be chosen.

1.5 Moving average processes

The **moving average process** of order q is denoted MA(q) and defined by

$$X_t = \sum_{s=0}^q \theta_s \epsilon_{t-s} \quad (1.4)$$

where $\theta_1, \dots, \theta_q$ are fixed constants, $\theta_0 = 1$, and $\{\epsilon_t\}$ is a sequence of independent (or uncorrelated) random variables with mean 0 and variance σ^2 .

It is clear from the definition that this is second order stationary and that

$$\gamma_k = \begin{cases} 0, & |k| > q \\ \sigma^2 \sum_{s=0}^{q-|k|} \theta_s \theta_{s+k}, & |k| \leq q \end{cases}$$

We remark that two moving average processes can have the same autocorrelation function. For example,

$$X_t = \epsilon_t + \theta\epsilon_{t-1} \quad \text{and} \quad X_t = \epsilon_t + (1/\theta)\epsilon_{t-1}$$

both have $\rho_1 = \theta/(1 + \theta^2)$, $\rho_k = 0$, $|k| > 1$. However, the first gives

$$\epsilon_t = X_t - \theta\epsilon_{t-1} = X_t - \theta(X_{t-1} - \theta\epsilon_{t-2}) = X_t - \theta X_{t-1} + \theta^2 X_{t-2} - \dots$$

This is only valid for $|\theta| < 1$, a so-called **invertible process**. No two invertible processes have the same autocorrelation function.

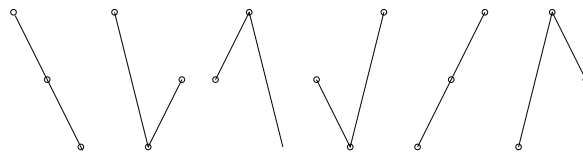
1.6 White noise

The sequence $\{\epsilon_t\}$, consisting of independent (or uncorrelated) random variables with mean 0 and variance σ^2 is called **white noise** (for reasons that will become clear later.) It is a second order stationary series with $\gamma_0 = \sigma^2$ and $\gamma_k = 0$, $k \neq 0$.

1.7 The turning point test

We may wish to test whether a series can be considered to be white noise, or whether a more complicated model is required. In later chapters we shall consider various ways to do this, for example, we might estimate the autocovariance function, say $\{\hat{\gamma}_k\}$, and observe whether or not $\hat{\gamma}_k$ is near zero for all $k > 0$.

However, a very simple diagnostic is the **turning point test**, which examines a series $\{X_t\}$ to test whether it is purely random. The idea is that if $\{X_t\}$ is purely random then three successive values are equally likely to occur in any of the six possible orders.



In four cases there is a turning point in the middle. Thus in a series of n points we might expect $(2/3)(n - 2)$ turning points.

In fact, it can be shown that for large n , the number of turning points should be distributed as about $N(2n/3, 8n/45)$. We reject (at the 5% level) the hypothesis that the series is unsystematic if the number of turning points lies outside the range $2n/3 \pm 1.96\sqrt{8n/45}$.

2 Models of stationary processes

2.1 Purely indeterministic processes

Suppose $\{X_t\}$ is a second order stationary process, with mean 0. Its **autocovariance function** is

$$\gamma_k = \mathbb{E}(X_t X_{t+k}) = \text{cov}(X_t, X_{t+k}), \quad k \in \mathbb{Z}.$$

1. As $\{X_t\}$ is stationary, γ_k does not depend on t .
2. A process is said to be **purely-indeterministic** if the regression of X_t on $X_{t-q}, X_{t-q-1}, \dots$ has explanatory power tending to 0 as $q \rightarrow \infty$. That is, the residual variance tends to $\text{var}(X_t)$.

An important theorem due to Wold (1938) states that every purely-indeterministic second order stationary process $\{X_t\}$ can be written in the form

$$X_t = \mu + \theta_0 Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + \dots$$

where $\{Z_t\}$ is a sequence of uncorrelated random variables.

3. A **Gaussian process** is one for which X_{t_1}, \dots, X_{t_n} has a joint normal distribution for all t_1, \dots, t_n . No two distinct Gaussian processes have the same autocovariance function.

2.2 ARMA processes

The **autoregressive moving average process**, ARMA(p, q), is defined by

$$X_t - \sum_{r=1}^p \phi_r X_{t-r} = \sum_{s=0}^q \theta_s \epsilon_{t-s}$$

where again $\{\epsilon_t\}$ is white noise. This process is stationary for appropriate ϕ, θ .

EXAMPLE 2.1

Consider the **state space model**

$$\begin{aligned} X_t &= \phi X_{t-1} + \epsilon_t, \\ Y_t &= X_t + \eta_t. \end{aligned}$$

Suppose $\{X_t\}$ is unobserved, $\{Y_t\}$ is observed and $\{\epsilon_t\}$ and $\{\eta_t\}$ are independent white noise sequences. Note that $\{X_t\}$ is AR(1). We can write

$$\begin{aligned} \xi_t &= Y_t - \phi Y_{t-1} \\ &= (X_t + \eta_t) - \phi(X_{t-1} + \eta_{t-1}) \\ &= (X_t - \phi X_{t-1}) + (\eta_t - \phi \eta_{t-1}) \\ &= \epsilon_t + \eta_t - \phi \eta_{t-1} \end{aligned}$$

Now ξ_t is stationary and $\text{cov}(\xi_t, \xi_{t+k}) = 0$, $k \geq 2$. As such, ξ_t can be modelled as a MA(1) process and $\{Y_t\}$ as ARMA(1, 1).

2.3 ARIMA processes

If the original process $\{Y_t\}$ is not stationary, we can look at the first order difference process

$$X_t = \nabla Y_t = Y_t - Y_{t-1}$$

or the second order differences

$$X_t = \nabla^2 Y_t = \nabla(\nabla Y)_t = Y_t - 2Y_{t-1} + Y_{t-2}$$

and so on. If we ever find that the differenced process is a stationary process we can look for a ARMA model of that.

The process $\{Y_t\}$ is said to be an **autoregressive integrated moving average process**, ARIMA(p, d, q), if $X_t = \nabla^d Y_t$ is an ARMA(p, q) process.

AR, MA, ARMA and ARIMA processes can be used to model many time series. A key tool in identifying a model is an estimate of the autocovariance function.

2.4 Estimation of the autocovariance function

Suppose we have data (X_1, \dots, X_T) from a stationary time series. We can estimate

- the mean by $\bar{X} = (1/T) \sum_1^T X_t$,
- the autocovariance by $c_k = \hat{\gamma}_k = (1/T) \sum_{t=k+1}^T (X_t - \bar{X})(X_{t-k} - \bar{X})$, and
- the autocorrelation by $r_k = \hat{\rho}_k = \hat{\gamma}_k / \hat{\gamma}_0$.

The plot of r_k against k is known as the **correlogram**. If it is known that μ is 0 there is no need to correct for the mean and γ_k can be estimated by

$$\hat{\gamma}_k = (1/T) \sum_{t=k+1}^T X_t X_{t-k}.$$

Notice that in defining $\hat{\gamma}_k$ we divide by T rather than by $(T - k)$. When T is large relative to k it does not much matter which divisor we use. However, for mathematical simplicity and other reasons there are advantages in dividing by T .

Suppose the stationary process $\{X_t\}$ has autocovariance function $\{\gamma_k\}$. Then

$$\text{var} \left(\sum_{t=1}^T a_t X_t \right) = \sum_{t=1}^T \sum_{s=1}^T a_t a_s \text{cov}(X_t, X_s) = \sum_{t=1}^T \sum_{s=1}^T a_t a_s \gamma_{|t-s|} \geq 0.$$

A sequence $\{\gamma_k\}$ for which this holds for every $T \geq 1$ and set of constants (a_1, \dots, a_T) is called a **nonnegative definite sequence**. The following theorem states that $\{\gamma_k\}$ is a valid autocovariance function if and only if it is nonnegative definite.

THEOREM 2.2 (Blochner) The following are equivalent.

1. There exists a stationary sequence with autocovariance function $\{\gamma_k\}$.
2. $\{\gamma_k\}$ is nonnegative definite.
3. The spectral density function,

$$f(\omega) = \frac{1}{\pi} \sum_{k=-\infty}^{\infty} \gamma_k e^{ik\omega} = \frac{1}{\pi} \gamma_0 + \frac{2}{\pi} \sum_{k=1}^{\infty} \gamma_k \cos(\omega k),$$

is positive if it exists.

Dividing by T rather than by $(T - k)$ in the definition of $\hat{\gamma}_k$

- ensures that $\{\hat{\gamma}_k\}$ is nonnegative definite (and thus that it could be the autocovariance function of a stationary process), and
- can reduce the L^2 -error of r_k .

2.5 Identifying a MA(q) process

In a later lecture we consider the problem of identifying an ARMA or ARIMA model for a given time series. A key tool in doing this is the correlogram.

The MA(q) process X_t has $\rho_k = 0$ for all k , $|k| > q$. So a diagnostic for MA(q) is that $|r_k|$ drops to near zero beyond some threshold.

2.6 Identifying an AR(p) process

The AR(p) process has ρ_k decaying exponentially. This can be difficult to recognise in the correlogram. Suppose we have a process X_t which we believe is AR(k) with

$$X_t = \sum_{j=1}^k \phi_{j,k} X_{t-j} + \epsilon_t$$

with ϵ_t independent of X_1, \dots, X_{t-1} .

Given the data X_1, \dots, X_T , the least squares estimates of $(\phi_{1,k}, \dots, \phi_{k,k})$ are obtained by minimizing

$$\frac{1}{T} \sum_{t=k+1}^T \left(X_t - \sum_{j=1}^k \phi_{j,k} X_{t-j} \right)^2.$$

This is approximately equivalent to solving equations similar to the Yule-Walker equations,

$$\hat{\gamma}_j = \sum_{\ell=1}^k \hat{\phi}_{\ell,k} \hat{\gamma}_{|j-\ell|}, \quad j = 1, \dots, k$$

These can be solved by the **Levinson-Durbin recursion**:

Step 0. $\sigma_0^2 := \hat{\gamma}_0$, $\hat{\phi}_{1,1} = \hat{\gamma}_1/\hat{\gamma}_0$, $k := 0$

Step 1. Repeat until $\hat{\phi}_{k,k}$ near 0:

$$\begin{aligned}
 & k := k + 1 \\
 \hat{\phi}_{k,k} & := \left(\hat{\gamma}_k - \sum_{j=1}^{k-1} \hat{\phi}_{j,k-1} \hat{\gamma}_{k-j} \right) / \sigma_{k-1}^2 \\
 \hat{\phi}_{j,k} & := \hat{\phi}_{j,k-1} - \hat{\phi}_{k,k} \hat{\phi}_{k-j,k-1}, \text{ for } j = 1, \dots, k-1 \\
 \sigma_k^2 & := \sigma_{k-1}^2 (1 - \hat{\phi}_{k,k}^2)
 \end{aligned}$$

We test whether the order k fit is an improvement over the order $k-1$ fit by looking to see if $\hat{\phi}_{k,k}$ is far from zero.

The statistic $\hat{\phi}_{k,k}$ is called the k th **sample partial autocorrelation coefficient** (PACF). If the process X_t is genuinely $\text{AR}(p)$ then the population PACF, $\phi_{k,k}$, is exactly zero for all $k > p$. Thus a diagnostic for $\text{AR}(p)$ is that the sample PACFs are close to zero for $k > p$.

2.7 Distributions of the ACF and PACF

Both the sample ACF and PACF are approximately normally distributed about their population values, and have standard deviation of about $1/\sqrt{T}$, where T is the length of the series. A rule of thumb is that ρ_k is negligible (and similarly $\phi_{k,k}$) if r_k (similarly $\hat{\phi}_{k,k}$) lies between $\pm 2/\sqrt{T}$. (2 is an approximation to 1.96. Recall that if $Z_1, \dots, Z_n \sim N(\mu, 1)$, a test of size 0.05 of the hypothesis $H_0 : \mu = 0$ against $H_1 : \mu \neq 0$ rejects H_0 if and only if \bar{Z} lies outside $\pm 1.96/\sqrt{n}$).

Care is needed in applying this rule of thumb. It is important to realize that the sample autocorrelations, r_1, r_2, \dots , (and sample partial autocorrelations, $\hat{\phi}_{1,1}, \hat{\phi}_{2,2}, \dots$) are not independently distributed. The probability that any one r_k should lie outside $\pm 2/\sqrt{T}$ depends on the values of the other r_k .

A ‘portmanteau’ test of white noise (due to Box & Pierce and Ljung & Box) can be based on the fact that approximately

$$Q'_m = T(T+2) \sum_{k=1}^m (T-k)^{-1} r_k^2 \sim \chi_m^2.$$

The sensitivity of the test to departure from white noise depends on the choice of m . If the true model is $\text{ARMA}(p, q)$ then greatest power is obtained (rejection of the white noise hypothesis is most probable) when m is about $p+q$.

4 Estimation of the spectrum

4.1 The periodogram

Suppose we have $T = 2m + 1$ observations of a time series, y_1, \dots, y_T . Define the **Fourier frequencies**, $\omega_j = 2\pi j/T$, $j = 1, \dots, m$, and consider the regression model

$$y_t = \alpha_0 + \sum_{j=1}^m \alpha_j \cos(\omega_j t) + \sum_{j=1}^m \beta_j \sin(\omega_j t),$$

which can be written as a general linear model, $Y = X\theta + \epsilon$, where

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_T \end{pmatrix}, \quad X = \begin{pmatrix} 1 & c_{11} & s_{11} & \cdots & c_{m1} & s_{m1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & c_{1T} & s_{1T} & \cdots & c_{mT} & s_{mT} \end{pmatrix}, \quad \theta = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \beta_1 \\ \vdots \\ \alpha_m \\ \beta_m \end{pmatrix}, \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_T \end{pmatrix},$$

$$c_{jt} = \cos(\omega_j t), \quad s_{jt} = \sin(\omega_j t).$$

The least squares estimates in this model are given by

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y.$$

Note that

$$\sum_{t=1}^T e^{i\omega_j t} = \frac{e^{i\omega_j}(1 - e^{i\omega_j T})}{1 - e^{i\omega_j}} = 0$$

$$\implies \sum_{t=1}^T c_{jt} + i \sum_{t=1}^T s_{jt} = 0 \implies \sum_{t=1}^T c_{jt} = \sum_{t=1}^T s_{jt} = 0$$

and

$$\sum_{t=1}^T c_{jt} s_{jt} = \frac{1}{2} \sum_{t=1}^T \sin(2\omega_j t) = 0,$$

$$\sum_{t=1}^T c_{jt}^2 = \frac{1}{2} \sum_{t=1}^T \{1 + \cos(2\omega_j t)\} = T/2,$$

$$\sum_{t=1}^T s_{jt}^2 = \frac{1}{2} \sum_{t=1}^T \{1 - \cos(2\omega_j t)\} = T/2,$$

$$\sum_{t=1}^T c_{jt} s_{kt} = \sum_{t=1}^T c_{jt} c_{kt} = \sum_{t=1}^T s_{jt} s_{kt} = 0, \quad j \neq k.$$

Using these, we have

$$\hat{\theta} = \begin{pmatrix} \hat{\alpha}_0 \\ \hat{\alpha}_1 \\ \vdots \\ \hat{\beta}_m \end{pmatrix} = \begin{pmatrix} T & 0 & \cdots & 0 \\ 0 & T/2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & T/2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_t y_t \\ \sum_t c_{1t} y_t \\ \vdots \\ \sum_t s_{mt} y_t \end{pmatrix} = \begin{pmatrix} \bar{y} \\ (2/T) \sum_t c_{1t} y_t \\ \vdots \\ (2/T) \sum_t s_{mt} y_t \end{pmatrix}$$

and the regression sum of squares is

$$\hat{Y}^\top \hat{Y} = Y^\top X (X^\top X)^{-1} X^\top Y = T \bar{y}^2 + \sum_{j=1}^m \frac{2}{T} \left[\left\{ \sum_{t=1}^T c_{jt} y_t \right\}^2 + \left\{ \sum_{t=1}^T s_{jt} y_t \right\}^2 \right].$$

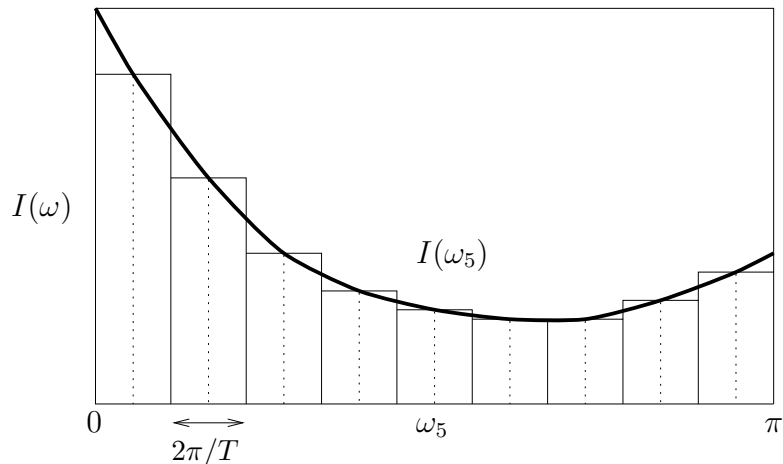
Since we are fitting T unknown parameters to T data points, the model fits with no residual error, i.e., $\hat{Y} = Y$. Hence

$$\sum_{t=1}^T (y_t - \bar{y})^2 = \sum_{j=1}^m \frac{2}{T} \left[\left\{ \sum_{t=1}^T c_{jt} y_t \right\}^2 + \left\{ \sum_{t=1}^T s_{jt} y_t \right\}^2 \right].$$

This motivates definition of the **periodogram** as

$$I(\omega) = \frac{1}{\pi T} \left[\left\{ \sum_{t=1}^T y_t \cos(\omega t) \right\}^2 + \left\{ \sum_{t=1}^T y_t \sin(\omega t) \right\}^2 \right].$$

A factor of $(1/2\pi)$ has been introduced into this definition so that the sample variance, $\hat{\gamma}_0 = (1/T) \sum_{t=1}^T (y_t - \bar{y})^2$, equates to the sum of the areas of m rectangles, whose heights are $I(\omega_1), \dots, I(\omega_m)$, whose widths are $2\pi/T$, and whose bases are centred at $\omega_1, \dots, \omega_m$. I.e., $\hat{\gamma}_0 = (2\pi/T) \sum_{j=1}^m I(\omega_j)$. These rectangles approximate the area under the curve $I(\omega)$, $0 \leq \omega \leq \pi$.



Using the fact that $\sum_{t=1}^T c_{jt} = \sum_{t=1}^T s_{jt} = 0$, we can write

$$\begin{aligned}
\pi T I(\omega_j) &= \left\{ \sum_{t=1}^T y_t \cos(\omega_j t) \right\}^2 + \left\{ \sum_{t=1}^T y_t \sin(\omega_j t) \right\}^2 \\
&= \left\{ \sum_{t=1}^T (y_t - \bar{y}) \cos(\omega_j t) \right\}^2 + \left\{ \sum_{t=1}^T (y_t - \bar{y}) \sin(\omega_j t) \right\}^2 \\
&= \left| \sum_{t=1}^T (y_t - \bar{y}) e^{i\omega_j t} \right|^2 \\
&= \sum_{t=1}^T (y_t - \bar{y}) e^{i\omega_j t} \sum_{s=1}^T (y_s - \bar{y}) e^{-i\omega_j s} \\
&= \sum_{t=1}^T (y_t - \bar{y})^2 + 2 \sum_{k=1}^{T-1} \sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y}) \cos(\omega_j k).
\end{aligned}$$

Hence

$$I(\omega_j) = \frac{1}{\pi} \hat{\gamma}_0 + \frac{2}{\pi} \sum_{k=1}^{T-1} \hat{\gamma}_k \cos(\omega_j k).$$

$I(\omega)$ is therefore a sample version of the spectral density $f(\omega)$.

4.2 Distribution of spectral estimates

If the process is stationary and the spectral density exists then $I(\omega)$ is an almost unbiased estimator of $f(\omega)$, but it is a rather poor estimator without some smoothing.

Suppose $\{y_t\}$ is Gaussian white noise, i.e., y_1, \dots, y_T are iid $N(0, \sigma^2)$. Then for any Fourier frequency $\omega = 2\pi j/T$,

$$I(\omega) = \frac{1}{\pi T} [A(\omega)^2 + B(\omega)^2], \quad (4.1)$$

where

$$A(\omega) = \sum_{t=1}^T y_t \cos(\omega t), \quad B(\omega) = \sum_{t=1}^T y_t \sin(\omega t). \quad (4.2)$$

Clearly $A(\omega)$ and $B(\omega)$ have zero means, and

$$\begin{aligned}
\text{var}[A(\omega)] &= \sigma^2 \sum_{t=1}^T \cos^2(\omega t) = T\sigma^2/2, \\
\text{var}[B(\omega)] &= \sigma^2 \sum_{t=1}^T \sin^2(\omega t) = T\sigma^2/2,
\end{aligned}$$

$$\text{cov}[A(\omega), B(\omega)] = \mathbb{E} \left[\sum_{t=1}^T \sum_{s=1}^T y_t y_s \cos(\omega t) \sin(\omega s) \right] = \sigma^2 \sum_{t=1}^T \cos(\omega t) \sin(\omega t) = 0.$$

Hence $A(\omega)\sqrt{2/T\sigma^2}$ and $B(\omega)\sqrt{2/T\sigma^2}$ are independently distributed as $N(0, 1)$, and $2[A(\omega)^2 + B(\omega)^2]/(T\sigma^2)$ is distributed as χ_2^2 . This gives $I(\omega) \sim (\sigma^2/\pi)\chi_2^2/2$. Thus we see that $I(\omega)$ is an unbiased estimator of the spectrum, $f(\omega) = \sigma^2/\pi$, but it is not consistent, since $\text{var}[I(\omega)] = \sigma^4/\pi^2$ does not tend to 0 as $T \rightarrow \infty$. This is perhaps surprising, but is explained by the fact that as T increases we are attempting to estimate $I(\omega)$ for an increasing number of Fourier frequencies, with the consequence that the precision of each estimate does not change.

By a similar argument, we can show that for any two Fourier frequencies, ω_j and ω_k the estimates $I(\omega_j)$ and $I(\omega_k)$ are statistically independent. These conclusions hold more generally.

THEOREM 4.1 Let $\{Y_t\}$ be a stationary Gaussian process with spectrum $f(\omega)$. Let $I(\cdot)$ be the periodogram based on samples Y_1, \dots, Y_T , and let $\omega_j = 2\pi j/T$, $j < T/2$, be a Fourier frequency. Then in the limit as $T \rightarrow \infty$,

- (a) $I(\omega_j) \sim f(\omega_j)\chi_2^2/2$.
- (b) $I(\omega_j)$ and $I(\omega_k)$ are independent for $j \neq k$.

Assuming that the underlying spectrum is smooth, $f(\omega)$ is nearly constant over a small range of ω . This motivates use of an estimator for the spectrum of

$$\hat{f}(\omega_j) = \frac{1}{2p+1} \sum_{\ell=-p}^p I(\omega_{j+\ell}).$$

Then $\hat{f}(\omega_j) \sim f(\omega_j)\chi_{2(2p+1)}^2/[2(2p+1)]$, which has variance $f(\omega)^2/(2p+1)$. The idea is to let $p \rightarrow \infty$ as $T \rightarrow \infty$.

4.3 The fast Fourier transform

$I(\omega_j)$ can be calculated from (4.1)–(4.2), or from

$$I(\omega_j) = \frac{1}{\pi T} \left| \sum_{t=1}^T y_t e^{i\omega_j t} \right|^2.$$

Either way, this requires of order T multiplications. Hence to calculate the complete periodogram, i.e., $I(\omega_1), \dots, I(\omega_m)$, requires of order T^2 multiplications. Computation effort can be reduced significantly by use of the **fast Fourier transform**, which computes $I(\omega_1), \dots, I(\omega_m)$ using only order $T \log_2 T$ multiplications.

6 Estimation of trend and seasonality

6.1 Moving averages

Consider a decomposition into trend, seasonal, cyclic and residual components.

$$X_t = T_t + I_t + C_t + E_t.$$

Thus far we have been concerned with modelling $\{E_t\}$. We have also seen that the periodogram can be useful for recognising the presence of $\{C_t\}$.

We can estimate trend using a **symmetric moving average**,

$$\hat{T}_t = \sum_{s=-k}^k a_s X_{t+s},$$

where $a_s = a_{-s}$. In this case the transfer function is real-valued.

The choice of moving averages requires care. For example, we might try to estimate the trend with

$$\hat{T}_t = \frac{1}{3} (X_{t-1} + X_t + X_{t+1}).$$

But suppose $X_t = T_t + \epsilon_t$, where trend is the quadratic $T_t = a + bt + ct^2$. Then

$$\hat{T}_t = T_t + \frac{2}{3}c + \frac{1}{3}(\epsilon_{t-1} + \epsilon_t + \epsilon_{t+1}),$$

so $\mathbb{E}\hat{T}_t = \mathbb{E}X_t + \frac{2}{3}c$ and thus \hat{T} is a biased estimator of the trend.

This problem is avoided if we estimate trend by fitting a polynomial of sufficient degree, e.g., to find a cubic that best fits seven successive points we minimize

$$\sum_{t=-3}^3 (X_t - b_0 - b_1t - b_2t^2 - b_3t^3)^2.$$

So

$$\begin{aligned} \sum X_t &= 7\hat{b}_0 && + 28\hat{b}_2 \\ \sum tX_t &= &28\hat{b}_1 &+ 196\hat{b}_3 \\ \sum t^2X_t &= 28\hat{b}_0 && + 196\hat{b}_2 \\ \sum t^3X_t &= &196\hat{b}_1 &+ 1588\hat{b}_3 \end{aligned}$$

Then

$$\begin{aligned} \hat{b}_0 &= \frac{1}{21} (7\sum X_t - \sum t^2X_t) \\ &= \frac{1}{21} (-2X_{-3} + 3X_{-2} + 6X_{-1} + 7X_0 + 6X_1 + 3X_2 - 2X_3). \end{aligned}$$

We estimate the trend at time 0 by $\hat{T}_0 = \hat{b}_0$, and similarly,

$$\hat{T}_t = \frac{1}{21} (-2X_{t-3} + 3X_{t-2} + 6X_{t-1} + 7X_t + 6X_{t+1} + 3X_{t+2} - 2X_{t+3}).$$

A notation for this moving average is $\frac{1}{21}[-2, 3, 6, 7, 6, 3, -2]$. Note that the weights sum to 1. In general, we can fit a polynomial of degree q to $2q+1$ points by applying a symmetric moving average. (We fit to an odd number of points so that the midpoint of fitted range coincides with a point in time at which data is measured.)

A value for q can be identified using the **variate difference method**: if $\{X_t\}$ is indeed a polynomial of degree q , plus residual error $\{\epsilon_t\}$, then the trend in $\Delta^r X_t$ is a polynomial of degree $q - r$ and

$$\Delta^q X_t = \text{constant} + \Delta^q \epsilon_t = \text{constant} + \epsilon_t - \binom{q}{1} \epsilon_{t-1} + \binom{q}{2} \epsilon_{t-2} - \dots + (-1)^q \epsilon_{t-q}.$$

The variance of $\Delta^q X_t$ is therefore

$$\text{var}(\Delta^q \epsilon_t) = \left[1 + \binom{q}{1}^2 + \binom{q}{2}^2 + \dots + 1 \right] \sigma^2 = \binom{2q}{q} \sigma^2,$$

where the simplification in the final line comes from looking at the coefficient of z^q in expansions of both sides of

$$(1+z)^q(1+z)^q = (1+z)^{2q}.$$

Define $V_r = \text{var}(\Delta^r X_t) / \binom{2r}{r}$. The fact that the plot of V_r against r should flatten out at $r \geq q$ can be used to identify q .

6.2 Centred moving averages

If there is a seasonal component then a **centred-moving average** is useful. Suppose data is measured quarterly, then applying twice the moving average $\frac{1}{4}[1, 1, 1, 1]$ is equivalent to applying once the moving average $\frac{1}{8}[1, 2, 2, 2, 1]$. Notice that this so-called **centred average of fours** weights each quarter equally. Thus if $X_t = I_t + \epsilon_t$, where I_t has period 4, and $I_1 + I_2 + I_3 + I_4 = 0$, then \hat{T}_t has no seasonal component. Similarly, if data were monthly we use a centred average of 12s, that is, $\frac{1}{24}[1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1]$.

6.3 The Slutsky-Yule effect

To remove both trend and seasonal components we might successively apply a number of moving averages, one or more to remove trend and another to remove seasonal effects. This is the procedure followed by some standard forecasting packages.

However, there is a danger that application of successive moving averages can introduce spurious effects. The **Slutsky-Yule effect** is concerned with the fact that a moving average repeatedly applied to a purely random series can introduce artificial cycles. Slutsky (1927) showed that some trade cycles of the nineteenth century were no more than artifacts of moving averages that had been used to smooth the data.

To illustrate this idea, suppose the moving average $\frac{1}{6}[-1, 2, 4, 2, -1]$ is applied k times to a white noise series. This moving average has transfer function, $a(\omega) = \frac{1}{6}(4 + 4\cos\omega - 2\cos 2\omega)$, which is maximal at $\omega = \pi/3$. The smoothed series has a spectral density, say $f_k(\omega)$, proportional to $a(\omega)^{2k}$, and hence for $\omega \neq \pi/3$, $f_k(\omega)/f_k(\pi/3) \rightarrow 0$ as $k \rightarrow \infty$. Thus in the limit the smoothed series is a periodic wave with period 6.

6.4 Exponential smoothing

Single exponential smoothing

Suppose the mean level of a series drifts slowly over time. A naive one-step-ahead forecast is $X_t(1) = X_t$. However, we might let all past observations play a part in the forecast, but give greater weights to those that are more recent. Choose weights to decrease exponentially and let

$$X_t(1) = \frac{1 - \omega}{1 - \omega^t} (X_t + \omega X_{t-1} + \omega^2 X_{t-2} + \cdots + \omega^{t-1} X_1),$$

where $0 < \omega < 1$. Define S_t as the right hand side of the above as $t \rightarrow \infty$, i.e.,

$$S_t = (1 - \omega) \sum_{s=0}^{\infty} \omega^s X_{t-s}.$$

S_t can serve as a one-step-ahead forecast, $X_t(1)$. S_t is known as **simple exponential smoothing**. Let $\alpha = 1 - \omega$. Simple algebra gives

$$\begin{aligned} S_t &= \alpha X_t + (1 - \alpha) S_{t-1} \\ X_t(1) &= X_{t-1}(1) + \alpha [X_t - X_{t-1}(1)]. \end{aligned}$$

This shows that the one-step-ahead forecast at time t is the one-step-ahead forecast at time $t - 1$, modified by α times the forecasting error incurred at time $t - 1$.

To get things started we might set S_0 equal to the average of the first few data points. We can play around with α , choosing it to minimize the mean square forecasting error. In practice, α in the range 0.25–0.5 usually works well.

Double exponential smoothing

Suppose the series is approximately linear, but with a slowly varying trend. If it were true that $X_t = b_0 + b_1 t + \epsilon_t$, then

$$\begin{aligned} S_t &= (1 - \omega) \sum_{s=0}^{\infty} \omega^s (b_0 + b_1(t - s) + \epsilon_t) \\ &= b_0 + b_1 t - b_1(1 - \omega) \sum_{s=0}^{\infty} \omega^s s + b_1(1 - \omega) \sum_{s=0}^{\infty} \omega^s \epsilon_{t-s}, \end{aligned}$$

and hence

$$\mathbb{E}S_t = b_0 + b_1t - b_1\omega/(1 - \omega) = \mathbb{E}X_{t+1} - b_1/(1 - \omega).$$

Thus the forecast has a bias of $-b_1/(1 - \omega)$. To eliminate this bias let $S_t^1 = S_t$ be the first smoothing, and $S_t^2 = \alpha S_t^1 + (1 - \alpha)S_{t-1}^2$ be the simple exponential smoothing of S_t^1 . Then

$$\begin{aligned}\mathbb{E}S_t^2 &= \mathbb{E}S_t^1 - b_1\omega/(1 - \omega) = \mathbb{E}X_t - 2b_1\omega/(1 - \omega), \\ \mathbb{E}(2S_t^1 - S_t^2) &= b_0 + b_1t, \quad \mathbb{E}(S_t^1 - S_t^2) = b_1(1 - \alpha)/\alpha.\end{aligned}$$

This suggests the estimates $\hat{b}_0 + \hat{b}_1t = 2S_t^1 - S_t^2$ and $\hat{b}_1 = \alpha(S_t^1 - S_t^2)/(1 - \alpha)$. The forecasting equation is then

$$X_t(s) = \hat{b}_0 + \hat{b}_1(t + s) = (2S_t^1 - S_t^2) + s\alpha(S_t^1 - S_t^2)/(1 - \alpha).$$

As with single exponential smoothing we can experiment with choices of α and find S_0^1 and S_0^2 by fitting a regression line, $X_t = \hat{\beta}_0 + \hat{\beta}_1t$, to the first few points of the series and solving

$$S_0^1 = \hat{\beta}_0 - (1 - \alpha)\hat{\beta}_1/\alpha, \quad S_0^2 = \hat{\beta}_0 - 2(1 - \alpha)\hat{\beta}_1/\alpha.$$

6.5 Calculation of seasonal indices

Suppose data is quarterly and we want to fit an additive model. Let \hat{I}_1 be the average of X_1, X_5, X_9, \dots , let \hat{I}_2 be the average of X_2, X_6, X_{10}, \dots , and so on for \hat{I}_3 and \hat{I}_4 . The cumulative seasonal effects over the course of year should cancel, so that if $X_t = a + I_t$, then $X_t + X_{t+1} + X_{t+2} + X_{t+3} = 4a$. To ensure this we take our final estimates of the seasonal indices as $I_t^* = \hat{I}_t - \frac{1}{4}(\hat{I}_1 + \dots + \hat{I}_4)$.

If the model is multiplicative and $X_t = aI_t$, we again wish to see the cumulative effects over a year cancel, so that $X_t + X_{t+1} + X_{t+2} + X_{t+3} = 4a$. This means that we should take $I_t^* = \hat{I}_t - \frac{1}{4}(\hat{I}_1 + \dots + \hat{I}_4) + 1$, adjusting so the mean of $I_1^*, I_2^*, I_3^*, I_4^*$ is 1.

When both trend and seasonality are to be extracted a two-stage procedure is recommended:

(a) Make a first estimate of trend, say \hat{T}_t^1 .

Subtract this from $\{X_t\}$ and calculate first estimates of the seasonal indices, say I_t^1 , from $X_t - \hat{T}_t^1$.

The first estimate of the deseasonalized series is $Y_t^1 = X_t - I_t^1$.

(b) Make a second estimate of the trend by smoothing Y_t^1 , say \hat{T}_t^2 .

Subtract this from $\{X_t\}$ and calculate second estimates of the seasonal indices, say I_t^2 , from $X_t - \hat{T}_t^2$.

The second estimate of the deseasonalized series is $Y_t^2 = X_t - I_t^2$.

7 Fitting ARIMA models

7.1 The Box-Jenkins procedure

A general ARIMA(p, d, q) model is $\phi(B)\nabla(B)^dX = \theta(B)\epsilon$, where $\nabla(B) = I - B$.

The **Box-Jenkins** procedure is concerned with fitting an ARIMA model to data. It has three parts: **identification**, **estimation**, and **verification**.

7.2 Identification

The data may require pre-processing to make it stationary. To achieve stationarity we may do any of the following.

- Look at it.
- Re-scale it (for instance, by a logarithmic or exponential transform.)
- Remove deterministic components.
- Difference it. That is, take $\nabla(B)^dX$ until stationary. In practice $d = 1, 2$ should suffice.

We recognise stationarity by the observation that the autocorrelations decay to zero exponentially fast.

Once the series is stationary, we can try to fit an ARMA(p, q) model. We consider the correlogram $r_k = \hat{\gamma}_k/\hat{\gamma}_0$ and the partial autocorrelations $\hat{\phi}_{k,k}$. We have already made the following observations.

- An MA(q) process has negligible ACF after the q th term.
- An AR(p) process has negligible PACF after the p th term.

As we have noted, very approximately, both the sample ACF and PACF have standard deviation of around $1/\sqrt{T}$, where T is the length of the series. A rule of thumb is that ACF and PACF values are negligible when they lie between $\pm 2/\sqrt{T}$. An ARMA(p, q) process has k th order sample ACF and PACF decaying geometrically for $k > \max(p, q)$.

7.3 Estimation

AR processes

To fit a pure AR(p), i.e., $X_t = \sum_{r=1}^p \phi_r X_{t-r} + \epsilon_t$ we can use the **Yule-Walker equations** $\gamma_k = \sum_{r=1}^p \phi_r \gamma_{|k-r|}$. We fit ϕ by solving $\hat{\gamma}_k = \sum_{r=1}^p \phi_r \hat{\gamma}_{|k-r|}$, $k = 1, \dots, p$. These can be solved by a **Levinson-Durbin recursion**, (similar to that used to solve for partial autocorrelations in Section 2.6). This recursion also gives the estimated

residual variance $\hat{\sigma}_p^2$, and helps in choice of p through the approximate log likelihood $-2 \log L \simeq T \log(\hat{\sigma}_p^2)$.

Another popular way to choose p is by minimizing **Akaike's AIC** (*an information criterion*), defined as $\text{AIC} = -2 \log L + 2k$, where k is the number of parameters estimated, (in the above case p). As motivation, suppose that in a general modelling context we attempt to fit a model with parameterised likelihood function $f(X | \theta)$, $\theta \in \Theta$, and this includes the true model for some $\theta_0 \in \Theta$. Let $X = (X_1, \dots, X_n)$ be a vector of n independent samples and let $\hat{\theta}(X)$ be the maximum likelihood estimator of θ . Suppose Y is a further independent sample. Then

$$-2n \mathbb{E}_Y \mathbb{E}_X \log f(Y | \hat{\theta}(X)) = -2 \mathbb{E}_X \log f(X | \hat{\theta}(X)) + 2k + O(1/\sqrt{n}),$$

where $k = |\Theta|$. The left hand side is $2n$ times the conditional entropy of Y given $\hat{\theta}(X)$, i.e., the average number of bits required to specify Y given $\hat{\theta}(X)$. The right hand side is approximately the AIC and this is to be minimized over a set of models, say $(f_1, \Theta_1), \dots, (f_m, \Theta_m)$.

ARMA processes

Generally, we use the maximum likelihood estimators, or at least squares numerical approximations to the MLEs. The essential idea is prediction error decomposition. We can factorize the joint density of (X_1, \dots, X_T) as

$$f(X_1, \dots, X_T) = f(X_1) \prod_{t=2}^T f(X_t | X_1, \dots, X_{t-1}).$$

Suppose the conditional distribution of X_t given (X_1, \dots, X_{t-1}) is normal with mean \hat{X}_t and variance P_{t-1} , and suppose also that X_1 is normal $N(\hat{X}_1, P_0)$. Here \hat{X}_t and P_{t-1} are functions of the unknown parameters $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q$ and the data.

The log likelihood is

$$-2 \log L = -2 \log f = \sum_{t=1}^T \left[\log(2\pi) + \log P_{t-1} + \frac{(X_t - \hat{X}_t)^2}{P_{t-1}} \right].$$

We can minimize this with respect to $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q$ to fit ARMA(p, q).

Additionally, the second derivative matrix of $-\log L$ (at the MLE) is the observed information matrix, whose inverse is an approximation to the variance-covariance matrix of the estimators.

In practice, fitting ARMA(p, q) the log likelihood ($-2 \log L$) is modified to sum only over the range $\{m+1, \dots, T\}$, where m is small.

EXAMPLE 7.1

For AR(p), take $m = p$ so $\hat{X}_t = \sum_{r=1}^p \phi_r X_{t-r}$, $t \geq m+1$, $P_{t-1} = \sigma_\epsilon^2$.

Note. When using this approximation to compare models with different numbers of parameters we should always use the same m .

Again we might choose p and q by minimizing the AIC of $-2 \log L + 2k$, where $k = p + q$ is the total number of parameters in the model.

7.4 Verification

The third stage in the Box-Jenkins algorithm is to check whether the model fits the data. There are several tools we may use.

- Overfitting. Add extra parameters to the model and use likelihood ratio test or t -test to check that they are not significant.
- Residuals analysis. Calculate the residuals from the model and plot them. The autocorrelation functions, ACFs, PACFs, spectral densities, estimates, etc., and confirm that they are consistent with white noise.

7.5 Tests for white noise

Tests for white noise include the following.

- The turning point test (explained in Lecture 1) compares the number of peaks and troughs to the number that would be expected for a white noise series.
- The **Box–Pierce test** is based on the statistic

$$Q_m = T \sum_{k=1}^m r_k^2,$$

where r_k is the k th sample autocorrelation coefficient of the residual series, and $p + q < m \ll T$. It is called a ‘portmanteau test’, because it is based on the all-inclusive statistic. If the model is correct then $Q_m \sim \chi_{m-p-q}^2$ approximately.

In fact, r_k has variance $(T - k)/(T(T + 2))$, and a somewhat more powerful test uses the Ljung-Box statistic quoted in Section 2.7,

$$Q'_m = T(T + 2) \sum_{k=1}^m (T - k)^{-1} r_k^2,$$

where again, $Q'_m \sim \chi_{m-p-q}^2$ approximately.

- Another test for white noise can be constructed from the periodogram. Recall that $I(\omega_j) \sim (\sigma^2/\pi)\chi_2^2/2$ and that $I(\omega_1), \dots, I(\omega_m)$ are mutually independent. Define $C_j = \sum_{k=1}^j I(\omega_k)$ and $U_j = C_j/C_m$. Recall that χ_2^2 is the same as the exponential distribution and that if Y_1, \dots, Y_m are i.i.d. exponential random variables,

then $(Y_1 + \dots + Y_j)/(Y_1 + \dots + Y_m)$, $j = 1, \dots, m - 1$, have the distribution of an ordered sample of $m - 1$ uniform random variables drawn from $[0, 1]$. Hence under the hypothesis that $\{X_t\}$ is Gaussian white noise U_j , $j = 1, \dots, m - 1$ have the distribution of an ordered sample of $m - 1$ uniform random variables on $[0, 1]$. The standard test for this is the Kolomogorov-Smirnov test, which uses as a test statistic, D , defined as the maximum difference between the theoretical distribution function for $U[0, 1]$, $F(u) = u$, and the empirical distribution $\hat{F}(u) = \{\#(U_j \leq u)\}/(m - 1)$. Percentage points for D can be found in tables.

7.6 Forecasting with ARMA models

Recall that $\phi(B)X = \theta(B)\epsilon$, so the power series coefficients of $C(z) = \theta(z)/\phi(z) = \sum_{r=0}^{\infty} c_r z^r$ give an expression for X_t as $X_t = \sum_{r=0}^{\infty} c_r \epsilon_{t-r}$.

But also, $\epsilon = D(B)X$, where $D(z) = \phi(z)/\theta(z) = \sum_{r=0}^{\infty} d_r z^r$ — as long as the zeros of θ lie strictly outside the unit circle and thus $\epsilon_t = \sum_{r=0}^{\infty} d_r X_{t-r}$.

The advantage of the representation above is that given (\dots, X_{t-1}, X_t) we can calculate values for $(\dots, \epsilon_{t-1}, \epsilon_t)$ and so can forecast X_{t+1} .

In general, if we want to forecast X_{T+k} from (\dots, X_{T-1}, X_T) we use

$$\hat{X}_{T,k} = \sum_{r=k}^{\infty} c_r \epsilon_{T+k-r} = \sum_{r=0}^{\infty} c_{k+r} \epsilon_{T-r},$$

which has the least mean squared error over all linear combinations of $(\dots, \epsilon_{T-1}, \epsilon_T)$. In fact,

$$\mathbb{E} \left((\hat{X}_{T,k} - X_{T+k})^2 \right) = \sigma_{\epsilon}^2 \sum_{r=0}^{k-1} c_r^2.$$

In practice, there is an alternative recursive approach. Define

$$\hat{X}_{T,k} = \begin{cases} X_{T+k}, & -(T-1) \leq k \leq 0, \\ \text{optimal predictor of } X_{T+k} \text{ given } X_1, \dots, X_T, & 1 \leq k. \end{cases}$$

We have the recursive relation

$$\hat{X}_{T,k} = \sum_{r=1}^p \phi_r \hat{X}_{T,k-r} + \hat{\epsilon}_{T+k} + \sum_{s=1}^q \theta_s \hat{\epsilon}_{T+k-s}$$

For $k = -(T-1), -(T-2), \dots, 0$ this gives estimates of $\hat{\epsilon}_t$ for $t = 1, \dots, T$.

For $k > 0$, this gives a forecast $\hat{X}_{T,k}$ for X_{T+k} . We take $\hat{\epsilon}_t = 0$ for $t > T$.

But this needs to be started off. We need to know $(X_t, t \leq 0)$ and $\epsilon_t, t \leq 0$. There are two standard approaches.

1. Conditional approach: take $X_t = \epsilon_t = 0, t \leq 0$.
2. Backcasting: we forecast the series in the reverse direction to determine estimators of X_0, X_{-1}, \dots and $\epsilon_0, \epsilon_{-1}, \dots$

MATLAB and Simulation

prof. Dr. Taha Hussein Ali

Department of Informatics & Statistics, College of Administration
and Economics, Salahaddin University, Erbil, Iraq

1.1: Introduction

This chapter gives you aggressively a gentle introduction to MATLAB programming language. It is designed to give students fluency in MATLAB programming language. Problem-based MATLAB examples have been given in a simple and easy way to make your learning fast and effective.

MATLAB is a programming language developed by MathWorks. It started as a matrix programming language with simple linear algebra programming. It can be run both under interactive sessions and as a batch job.

We assume you have a little knowledge of any computer programming and understand concepts like variables, constants, expressions, statements, etc. If you have done programming in any other high-level language like C, C++, or Java, then it will be very beneficial, and learning MATLAB will be fun for you.

MATLAB (**MA**Trix **LAB**oratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization, and programming.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyzing data developing algorithms; creating models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

The reporting of a simulation experiment should receive the same care and consideration that would be accorded the reporting of other scientific experiments. Hoaglin and Andrews (1975) outline the items that should be included in a report of a simulation study. In addition to a careful general description of the experiment, the report should include a mention of the random number generator used, any variance-reducing methods employed, and a justification of the simulation sample size. The *Journal of the American Statistical Association* includes these reporting standards in its style guide for authors.

Closely related to the choice of the sample size is the standard deviation of the estimates that result from the study. The sample standard deviations actually achieved should be included as part of the report. Standard deviations are often reported in parentheses beside the estimates with which they are associated. A formal analysis, of course, would use the sample variance of each estimate to assess the significance of the differences observed between points in the design space; that is, a formal analysis of the simulation experiment would be a standard analysis of variance.

CONTENTS

| | | |
|--------|--|----|
| 1.1 | Introduction | 1 |
| 1.2 | MATLAB's Power of Computational Mathematics | 4 |
| 1.3 | Features of MATLAB | 4 |
| 1.4 | Desktop Basics | 4 |
| 1.5 | Matrices and Vectors | 6 |
| 1.5.1 | Assignment and Operators | 7 |
| 1.5.2 | Extracting a Sub-Matrix | 7 |
| 1.5.3 | Matrix Functions in Matlab | 8 |
| 1.6 | Pre-Defined Variables | 9 |
| 1.7 | Plotting in Matlab | 10 |
| 1.8 | Logical Subscribing | 14 |
| 1.9 | Multidimensional Arrays | 15 |
| 1.10 | Programming in MATLAB | 16 |
| 1.10.1 | Relational Operators | 17 |
| 1.10.2 | Logical Operators | 17 |
| 1.10.3 | Conditional Structures | 17 |
| 1.11 | Matlab Iteration Structures | 19 |
| 1.12 | M-Files | 20 |
| 1.12.1 | M-Files –Scripts | 20 |
| 1.12.2 | M-Files –Functions | 21 |
| 1.13 | Debugging in Matlab | 22 |
| 1.14 | Advanced Features to Explore | 23 |
| 1.15 | Descriptive statistics with the Statistics Toolbox of MATLAB | 24 |
| 1.16 | Simulation of linear models | 26 |

| | | |
|--------|--|----|
| 1.16.1 | Simulation of simple linear model | 26 |
| 1.16.2 | Ordinary Least Squares Regression | 30 |
| 1.16.3 | Simple linear regression in matrix form | 33 |
| 1.16.4 | Multiple Linear Regression | 35 |
| 1.16.5 | Multiple linear regression with the Statistics Toolbox of MATLAB | 37 |
| 1.17 | Simulation of Stochastic processes | 40 |
| 1.17.1 | Simulation of Bernoulli process | 40 |
| 1.17.2 | Simulation of Random walk | 41 |
| 1.17.3 | Simulation of Poisson process | 41 |
| 1.17.4 | Simulation of Autoregressive process | 42 |
| 1.17.5 | Simulation of Moving average process | 43 |
| 1.18 | Nonlinear Regression | 44 |
| 1.18.1 | Nonlinear Transformations | 44 |
| 1.18.2 | Polynomial fitting | 46 |
| | PROBLEMS | 49 |
| | Reference | 52 |

1.2: MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

1.3: Features of MATLAB

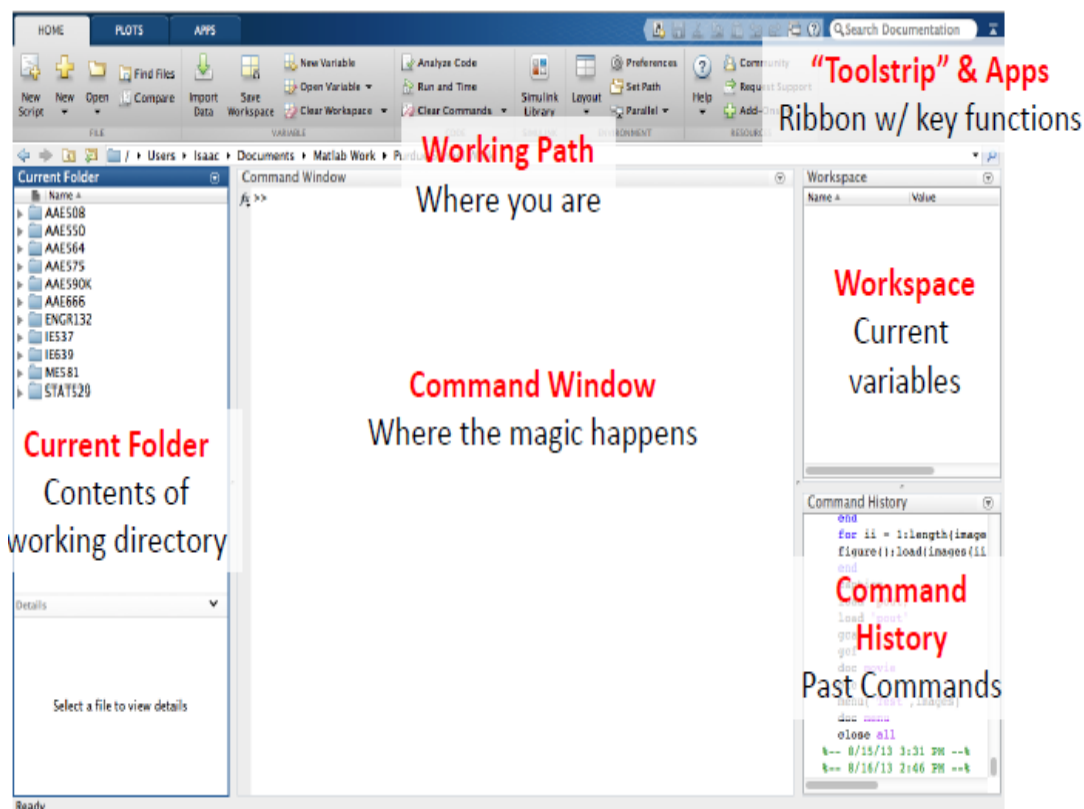
The following are the basic features of MATLAB:

- High-level language for numerical computation, visualization, and application development.
- Interactive environment for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations.
- Built-in graphics for visualizing data and tools for creating custom plots.
- Development tools for improving code quality and maintainability and maximizing performance.
- Tools for building applications with custom graphical interfaces.
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel.

1.4: Desktop Basics

MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

MATLAB (R2013a) Environment



The desktop has the following panels:

- **Current Folder** — Access your files.
- **Command Window** — Enter commands at the command line, indicated by the prompt (`>>`).
- **Workspace** — Explore data that you create or import from files.

As you work in MATLAB, you issue commands that create variables and call functions.

For example, create a variable named `x` by typing this statement at the command line:

```
>> x = 3
```

MATLAB adds variable `x` to the workspace and displays the result in the Command Window.

```
x =
    3
```

Create a few more variables.

```
>> y = 5
```

```
y =
    5
```

```
>> z = x + y
```

```
z =
```

8

```
>> d = cos(x)
```

```
d =
```

```
-0.989995
```

When you do not specify an output variable, MATLAB uses the variable `ans`, short for *answer*, to store the results of your calculation.

```
>> sin(x)
```

```
ans =
```

```
0.14112
```

If you end a statement with a semicolon, MATLAB performs the computation, but suppresses the display of output in the Command Window.

```
>> e = x * y;
```

You can recall previous commands by pressing the up- and down-arrow keys, \uparrow and \downarrow . Press the arrow keys either at an empty command line or after you type the first few characters of a command. For example, to recall the command `y = 5`, type `b`, and then press the up-arrow key.

1.5: Matrices and Vectors

MATLAB is an abbreviation for "matrix laboratory." While other programming languages mostly work with numbers one at a time, MATLAB is designed to operate primarily on whole matrices and Vectors.

All MATLAB variables are multidimensional *Vectors*, no matter what type of data. A *matrix* is a two-dimensional *Vectors* often used for linear algebra.

Vector Creation

To create a vector with four elements in a single row, separate the elements with either a comma (,) or a space

```
>> a = [1 2 3 4]
```

```
a =
```

```
1 2 3 4
```

This type of array is a *row vector*.

To create a matrix that has multiple rows, separate the rows with semicolons.

```
>> a = [1 2 3; 4 5 6; 7 8 10]
```

```
a =
```

```
1 2 3  
4 5 6  
7 8 10
```

Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 5-by-1 column vector of zeros.

```
>> z = zeros(5,1)
```

```
z =
```

```
0
0
0
0
0
```

And we have:

```
>> y = ones(1,5)
```

```
y =
```

```
1 1 1 1 1
```

1.5.1: Assignment and Operators

| | | |
|------------------------------------|----|---------------|
| Assignment (assign b to a) | = | a = b |
| Addition | + | a + b |
| Subtraction | - | a - b |
| Multiplication: Matrix | * | a * b |
| Multiplication: Element-by-Element | .* | a .* b |
| Division: Matrix | / | a / b |
| Division: Element-by-Element | ./ | a ./ b |
| Power: Matrix | ^ | a ^ b |
| Power: Element-by-Element | .^ | a .^ b |

1.5.2: Extracting a Sub-Matrix

A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both the rows and columns to extract

```
sub_matrix = matrix(r1:r2 , c1:c2)
```

```
sub_matrix = matrix(rows, columns)
```

Where r1 and r2 specify the beginning and ending rows, and c1 and r2 specify the beginning and ending columns to extract

Colon Operator

The colon operator helps to specify ranges

a : b Goes from **a** to **b** in increments of 1. If **a > b**, results in null vector

a : n : b Goes from **a** to **b** in increments of **n**. If **n** < 0 then **a** > **b**

A(:, b) The b^{th} column of **A**

A(a, :) The a^{th} row of **A**

A(:, :) All of the rows and columns of **A** (i.e., the **A** matrix)

A(a : b) Elements **a** to **b** (in increments of 1) of **A**. **NOTE:** Elements are counted down the columns and then across the rows!

A(:, a : b) All rows and columns **a** to **b** (in increments of 1)

A(:) All elements of **A** in a single column vector

Matrices

- Accessing single elements of a matrix:

A(a, b) → Element in row **a** and column **b**

- Accessing multiple elements of a matrix:

A(1,4) + A(2,4) + A(3,4) + A(4,4)
sum(A(1:4,4)) or **sum(A(:,end))**

– In locations, the keyword **end** refers to the *last* row or column

- Deleting rows and columns:

A(:, 2) = [] → Deletes the second column of **A**

- Concatenating matrices **A** and **B**:

C = [A ; B] for vertical concatenation
C = [A , B] for horizontal concatenation

1.5.3: Matrix Functions in Matlab

A = ones(m, n) Creates an $m \times n$ matrix of 1's

A = zeros(n,m) Creates an $m \times n$ matrix of 0's

A = eye(n) Creates an $n \times n$ identity matrix

A = NaN(m,n) Creates an $m \times n$ matrix of NaN's

A = inf(m,n) Creates an $m \times n$ matrix of inf's

A = diag(x) Creates a diagonal matrix **A** of **x**

x = diag(A) Extracts diagonal elements from **A**

[m,n] = size(A) Returns the dimensions of **A**

n = length(A) Returns the largest dimension of **A**

n = numel(A) Returns number of elements of **A**

| | |
|-------------------------|--------------------------------|
| x = sum(A) | Vector with sum of columns |
| x = prod(A) | Vector with product of columns |
| B = A' | Transposed matrix |
| d = det(A) | Determinant |
| [x,y] = eig(A) | Eigenvalues and eigenvectors |
| B = inv(A) | Inverse of square matrix |
| B = pinv(A) | Moore-Penrose pseudoinverse |
| B = chol(A) | Cholesky decomposition |
| [Q,R] = qr(A) | QR decomposition |
| [U,D,V] = svd(A) | Singular value decomposition |

1.5.4: Logic in Matrices

| | |
|-------------------------------------|--|
| B = any(A) | Determine if any elements in each column of A are nonzero |
| B = all(A) | Determine if all elements in each column of A are nonzero |
| B = find(A) | Find indices of all non-zero elements of A Can also use logic! |
| B = find(A>4 &A<5) | Elements > 4 and < 5 |
| B = all(A~=9) | Elements not equal to 9 |
| B = any(A==3 A==5) | Elements equal to 3 or 5 |

1.6: Pre-Defined Variables

MATLAB has several pre-defined / reserved variables, (**Beware**): These variables can be overwritten with custom values!

| | |
|----------|--|
| ans | Default variable name for results |
| pi | Value of π |
| eps | Smallest incremental number (2.2204e-16) |
| Inf/ inf | Infinity |
| NaN/ nan | Not a number (e.g., 0/0) |
| realmin | Smallest usable positive real number (2.2251e-308) |
| realmax | Largest usable positive real number (1.7977e+308) |
| i / j | Square root of (-1) |

1.7: Plotting in Matlab

- Matlab has extensive plotting capabilities
- Basic function is **plot** to plot one vector vs. another vector (vectors must have same length)

plot(x, y)

- Can also simply plot one vector vs. its index

plot(x)

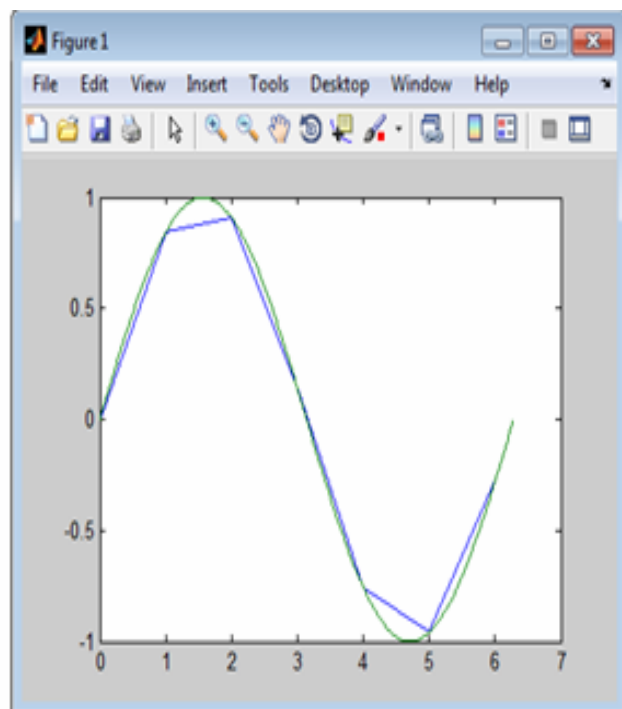
- Repeat three arguments to plot multiple vectors, different pairs of x and y data can have different sizes!

plot(x1, y1, x2, y2, x3, y3)

Example 1.1:

```
>> x1 = 0:1:2*pi;  
>> y1 = sin(x1);  
>> x2 = 0:0.01:2*pi;  
>> y2 = sin(x2);  
>> plot(x1, y1, x2, y2)
```

Matlab will automatically change the colors of the lines if plotted with one plot command!

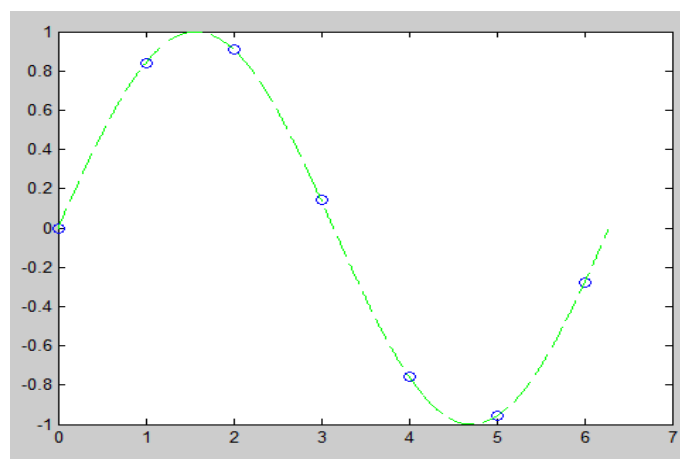


- The line style, marker symbol, and color of the plot are specified by the **Line Spec**.
- **Line Spec** is specified for each line after the y data and is optional.
- To see all options in Matlab: **doc Line Spec**
- Common formatting:

| Lines | Markers | Colors |
|---------------|-------------|-------------|
| '-' solid | '+' plus | 'r' red |
| '- -' dashed | 'o' circle | 'g' green |
| ':' dotted | '*' star | 'b' blue |
| '.-' dash-dot | '.' point | 'k' black |
| | 's' square | 'y' yellow |
| | 'd' diamond | 'c' cyan |
| | 'x' cross | 'm' magenta |

Example 1.2:

```
>> x1 = 0:1:2*pi; y1 = sin(x1);
>> x2 = 0:0.01:2*pi; y2 = sin(x2);
>> plot(x1,y1,'bo',x2,y2,'g--')
```



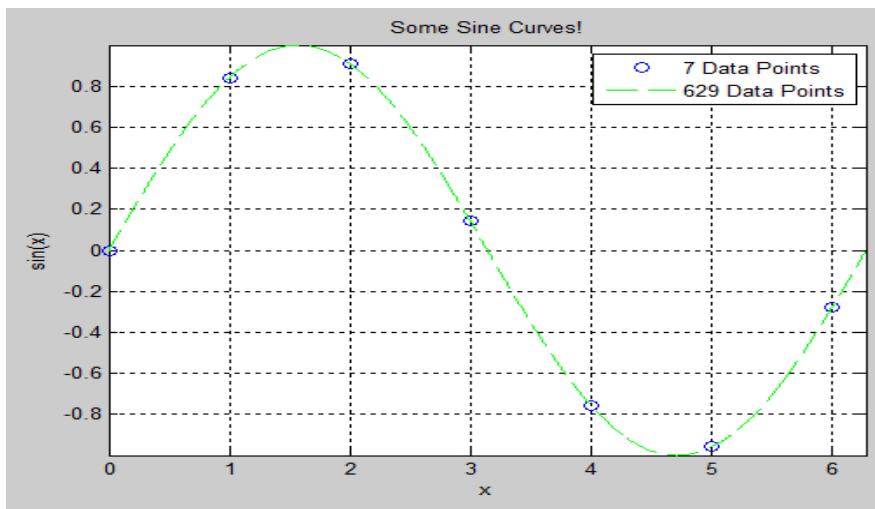
- Other commands allow you to modify the plot
 - Annotation: title, x label, y label, z label
 - Grid: grid **on**, grid **off**, grid **minor**
 - Axes: **axis**([xmin xmax ymin ymax]), axis **keyword**(doc **axis** for full keyword list)
 - Legend: **legend**('Line 1','Line 2','Location','Position')
- Another way to plot multiple lines is with the **hold** command

```
hold on
plot(x1,y1)
plot(x2,y2)
hold off
```

- Unless a new figure is created using **figure()**, any plotting function will overwrite the current plot

Example 1.3:

```
x1 = 0:1:2*pi; y1 = sin(x1);
x2 = 0:0.01:2*pi; y2 = sin(x2);
plot(x1,y1,'bo',x2,y2,'g--')
legend('7 Data Points','629 Data Points','Location','NorthEast')
title('Some Sine Curves!')
xlabel('x')
ylabel('sin(x)')
grid on
axis tight
```

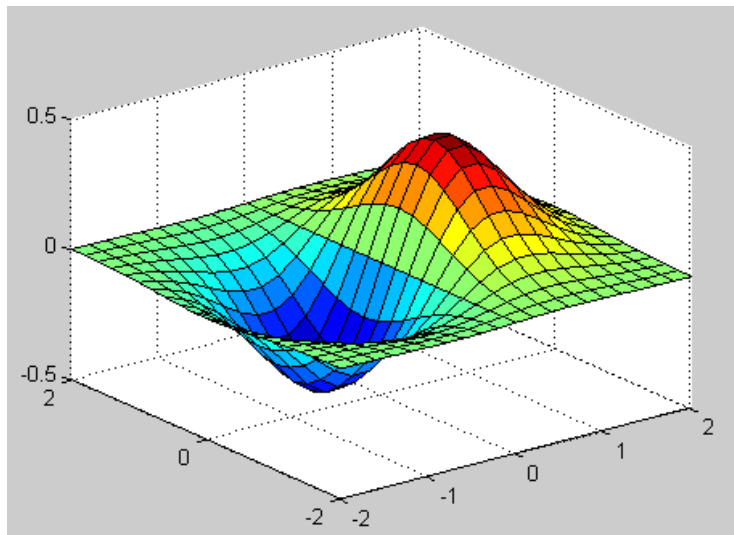


- 3-D Plots: Three-dimensional plots typically display a surface defined by a function in two variables, $z = f(x, y)$.

To evaluate z , first create a set of (x,y) points over the domain of the function using *meshgrid*.

Example 1.4:

```
>> [X,Y] = meshgrid(-2: .2: 2);
>> Z = X .* exp(-X.^2 - Y.^2);
>> surf(X,Y,Z)
```



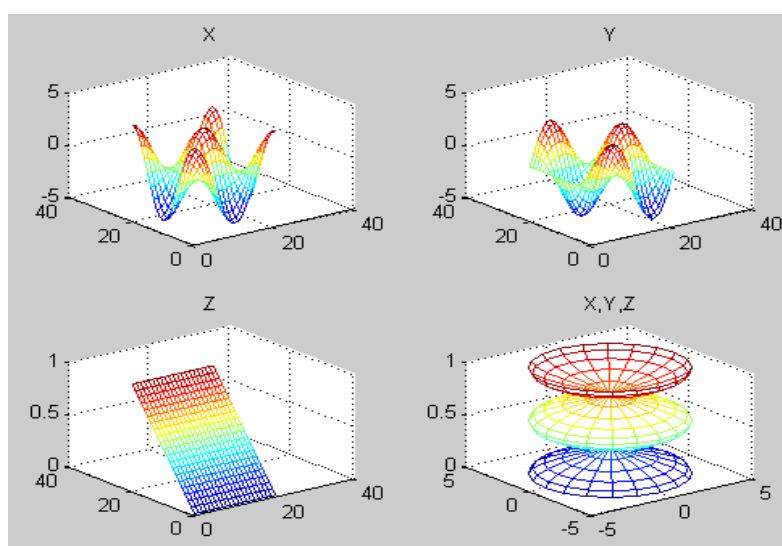
Both the surf function and its companion mesh display surfaces in three dimensions. surf displays both the connecting lines and the faces of the surface in color. Mesh produces wireframe surfaces that color only the lines connecting the defining points.

- Subplots: You can display multiple plots in different subregions of the same window using the subplot function.

The first two inputs to subplot indicate the number of plots in each row and column. The third input specifies which plot is active. As the following example shows:

Example 1.5: create four plots in a 2-by-2 grid within a figure window.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
subplot(2,2,4); mesh(X,Y,Z); title('X,Y,Z');
```



- Other plotting functions in Matlab

– **Log scales:** semilogx, semilogy, loglog

- **Two y-axes scales:** plotyy
- 3D line plots: plot3
- **Surface and mesh plots:** surf, surfc, mesh, meshc, waterfall, ribbon, trisurf, trimesh
- **Histograms:** hist, histc, area, pareto
- **Bar plots:** bar, bar3, barh, bar3h
- **Pie charts:** pie, pie3, rose
- **Discrete data:** stem, stem3, stairs, scatter, scatter3, spy, plotmatrix
- **Polar plots:** polar, rose, compass
- **Contour plots:** contour, contourf, contourc, contour3, contourslice
- **Vector fields:** feather, quiver, quiver3, compass, streamslice, streamline

1.8: Logical Subscripting

The logical vectors created from logical and relational operations can be used to reference subarrays. Suppose X is an ordinary matrix and L is a matrix of the same size that is the result of some logical operation. Then $X(L)$ specifies the elements of X where the elements of L are nonzero.

This kind of subscripting can be done in one step by specifying the logical operation as the subscripting expression. Suppose you have the following set of data:

```
x = [2.1 1.7 1.6 1.5 NaN 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8];
```

The NaN is a marker for a missing observation, such as a failure to respond to an item on a questionnaire. To remove the missing data with logical indexing, use **isfinite(x)**, which is true for all finite numerical values and false for NaN and Inf:

```
x = x(isfinite(x))
```

```
x =
```

```
2.1 1.7 1.6 1.5 1.9 1.8 1.5 5.1 1.8 1.4 2.2 1.6 1.8
```

Now there is one observation, 5.1, which seems to be very different from the others. It is an *outlier*. The following statement removes outliers, in this case those elements more than three standard deviations from the mean:

```
x = x(abs(x-mean(x)) <= 3*std(x))
```

```
x =
```

```
2.1 1.7 1.6 1.5 1.9 1.8 1.5 1.8 1.4 2.2 1.6 1.8
```

1.9: Multidimensional Arrays

Multidimensional arrays in the MATLAB environment are arrays with more than two subscripts. One way of creating a multidimensional array is by calling zeros, ones, rand, or randn with more than two arguments. For example,

```
R = randn(3,4,2)
```

```
R(:, :, 1) =
```

```

-1.0891      1.1006     -1.4916      2.3505
 0.032557    1.5442     -0.7423     -0.6156
 0.55253     0.085931    -1.0616     0.74808

```

```
R(:, :, 2) =
```

```

-0.19242    -1.4023    -0.17738     0.29158
 0.88861    -1.4224    -0.19605     0.19781
-0.76485     0.48819     1.4193      1.5877

```

Creates a 3-by-4-by-2 array, with a total of ($3*4*2 = 24$) normally distributed random elements.

A three-dimensional array might represent three-dimensional physical data; say the temperature in a room, sampled on a rectangular grid. Or it might represent a sequence of matrices, $A^{(k)}$, or samples of a time-dependent matrix, $A(t)$. In these latter cases, the $(i, j)^{th}$ element of the k^{th} matrix, or the t^{kth} matrix, is denoted by $A(i, j, k)$.

MATLAB and Dürer's versions of the magic square of order 4 differ by an interchange of two columns. Many different magic squares can be generated by interchanging columns. The statement

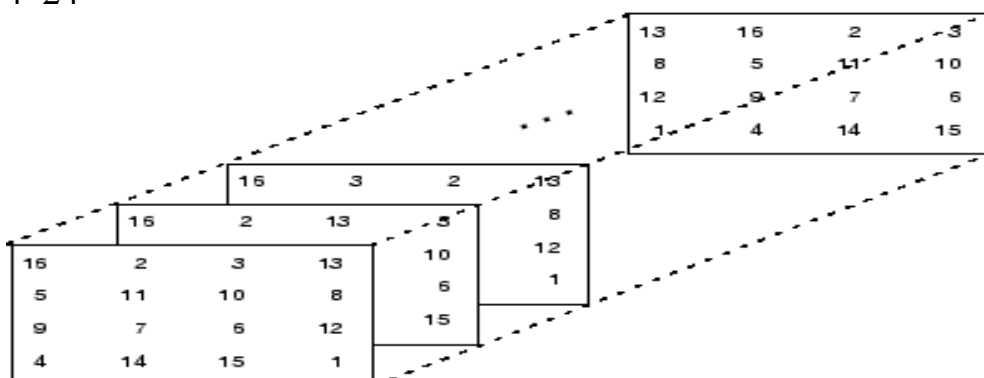
```
p = perms(1:4);
```

Generates the $4! = 24$ permutations of 1:4. The k^{th} permutation is the row vector $p(k,:)$. Then stores the sequence of (24) magic squares in a three-dimensional array, M. The size of M is

```
size(M)
```

```
ans =
```

```
4 4 24
```



Note: The order of the matrices shown in this illustration might differ from your results. The perms function always returns all permutations of the input vector, but the order of the permutations might be different for different MATLAB versions.

The statement

```
sum(M,d)
```

Computes sums by varying the d^{th} subscript. So

```
sum(M,1)
```

Is a 1-by-4-by-24 array containing 24 copies of the row vector:

```
34 34 34 34
```

And

```
sum(M,2)
```

Is a 4-by-1-by-24 array containing 24 copies of the column vector

```
34  
34  
34  
34
```

Finally,

```
S = sum(M,3)
```

Adds the (24) matrices in the sequence. The result has size 4-by-4-by-1, so it looks like a 4-by-4 array:

```
S =  
204 204 204 204  
204 204 204 204  
204 204 204 204  
204 204 204 204
```

1.10: Programming in Matlab

- Elements of Matlab as a programming language:
 - Expressions
 - Flow Control Blocks
 - Conditional
 - Iterations (Loops)
 - Scripts
 - Functions
 - Objects and classes (not covered here)
- Be mindful of existing variables and function names!

- Creating a variable or function that is already used by Matlab will cause troubles and errors!
- Example: Saving a variable as **sin = 10** will prevent you from using the sine function! Use something more descriptive such as **sin_x= 10**

1.10.1: Relational Operators

- Matlab has six relational Operators

- Less Than <
- Less Than or Equal <=
- Greater Than >
- Greater Than or Equal >=
- Equal to ==
- Not Equal to ~=

- Relational operators can be used to compare scalars to scalars, scalars to matrices/vectors, or matrices/vectors to matrices/vectors of the same size

- Relational operators to precedence after addition / subtraction

1.10.2: Logical Operators

- Matlab supports four logical operators

- Not ~
- And & or &&
- Or | or ||
- Exclusive Or (xor) xor()

- Not has the highest precedence and is evaluated after parentheses and exponents

- And, or, xor have lowest precedence and are evaluated last

1.10.3: Conditional Structures

- If / Then Structure

```
if expression
  commands
end
```

- Example

```
if (x > 4) && (y < 10)
  z = x + y;
end
```

- If / Else Structure

```
if expression
  commands
else
  commands
end
```

- Example

```
if (x > 4) && (y < 10)
  z = x + y;
else
  z = x * y;
end
```

- If / Elseif/ Else Structure

```
if expression
    commands
elseif expression
    commands
else
    commands
end
```

- Example

```
if (x > 4) && (y < 10)
    z = x + y;
elseif (x < 3)
    z = 10 * x;
elseif (y > 12)
    z = 5 / y;
else
    z = x * y;
end
```

- Conditional Structures can be nested inside each other

```
if (x > 3)
if (y > 5)
    z = x + y;
elseif (y < 5)
    z = x - y;
end
elseif (y < 10)
    z = x * y;
else
    z = x / y;
end
```

- Matlab will auto-indent for you, but indentation is not required

- Switch / Case / Otherwise function used if known cases of a variable will exist

– Used in place of If / Elseif/ Else structure

- Syntax

```
switch switch_expression
case case_expression
    statements
case case_expression
    statements
otherwise
    statements
end
```

| if-elseif-else | switch -case -otherwise |
|--|--|
| <pre> if x == 1 z = 5; elseif x == 2 z = 4; elseif x == 3 z = 3; elseif (x == 4) (x == 5) z = 2; else z = 1; end </pre> | <pre> switch x case 1 z = 5; case 2 z = 4; case 3 z = 3; case{4 , 5} z = 2; otherwise z = 1; end </pre> |

1.11: Matlab Iteration Structures

- Definite looping structures (**for**)

```

for variable = expression
    commands
end

```

- Can also nest loops!
- Can mix for / while loops

- Indefinite looping structures (**while**)

```

while expression
    commands
end

```

- You need to make sure the variable in the while loop expression is changed during the loop!

- May lead to an infinite loop!

- Example

```

for i = 1:1:25
    A(i) = i^2;
end

```

- Nested For Loop Example

```

for i = 1:1:25
    for j = 1:1:4
        A(i,j) = i*j;
    end
end

```

- Example

```

x = 0; y = 0;
while x < 10
    y = y + x;
    x = x + 1;
end

```

- Example for infinite Loop

```

x = 0;
while x < 10
    y = x;
end

```

1.12: M-Files

- Text files containing Matlab programs
 - Can be called from the command line or from other M-Files
- Contain “.m” file extension
- Two main types of M-Files
 - Scripts
 - Functions
- Comment character is %
 - % will comment out rest of line

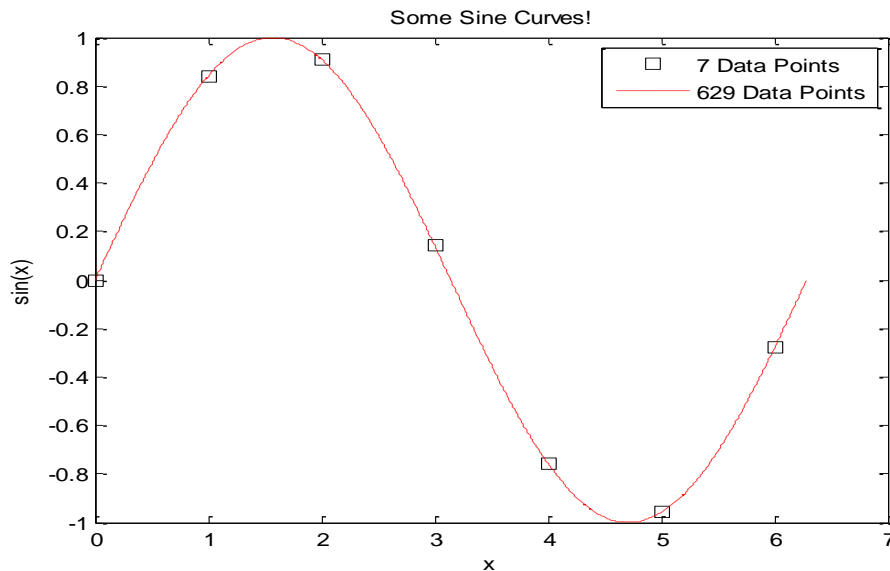
1.12.1: M-Files –Scripts

- Scripts are simply M-Files with a set of commands to run
 - Do not require input values or have output values
 - Execute commands similarly to how they would be done if typed into the command window
 - Ctrl + N
 - Select New → Script from Menu
- To run M-File:

–>> F5 or Run 

Example 1.6:

```
figure() % New Figure
x1 = 0:1:2*pi; y1 = sin(x1); % First Data Set
x2 = 0:0.01:2*pi; y2 = sin(x2); % Second Data Set
plot(x1,y1,'sk',x2,y2,'r--') % Make Plot
title('Some Sine Curves!') % Add Title, Labels, Legend,
etc.
xlabel('x')
ylabel('sin(x)')
legend('7 Data Points','629 Data
Points','Location','NorthEast')
```



1.12.2: M-Files –Functions

- Functions typically require input or output values
- “What happens in the function, stays in the function”
 - Only variables visible *after* function executes are those variables defined as output
- Usually one file for each function defined
- Structure:


```
function [outputs] = funcName (inputs)
    commands;
end
```
- Function Definition Line Components
 1. Function keyword → Identifies M-File as a function
 2. Output Variables → Separated by commas, contained in **square brackets**
 - Output variables must match the name of variables inside the function!
 3. Function Name → must match the name of the .m file!
 4. Input Variables → Separated by commas, contained in **parentheses**
 - Input variables must match the name of variables inside the function!
- When calling a function, you can use any name for the variable as input or output
 - The names **do not** have to match the names of the .m file

Example 1.7: Explain function to calculate the area and perimeter of a rectangle

```
function [area, perimeter] = dF(base, height)
% "df" Demo func. to calculate the area and perimeter of a rectangle
% Function can handle scalar and vector inputs
% Isaac Tetzloff -Aug 2013
area = base .* height; % Calculate the area
```

```
perimeter = 2 * (base + height); % Calculate the perimeter
end
```

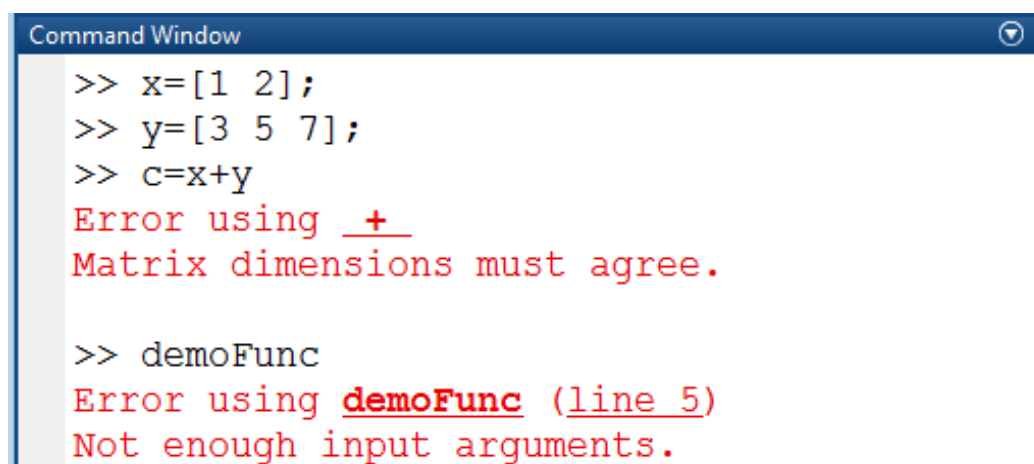
```
>> [a, p] = dF(10, 15); % Returns both values as a & p
>> area = dF(10, 5);% Returns area and saves as area
>> perim= dF(5, 15);% Returns area and saves as perim!
>> [perim, area] = dF(5, 15);% Saves area as perim, and vice versa!
>> x = [1 2 3]; y = [5 4 3];
>> [x, y] = dF(x, y);% Returns both and overwrites input!
```

- In modified function below, only variables output are **area** and **perimeter**
 - Matlab and other functions will not have access to **depth**, **mult**, **add**, or **volume**!
 - **REMEMBER:** *What happens in the function stays in the function!*

```
function [area, perimeter] = dF(base, height)
depth = 10; % Assume 3D prism has depth of 10
mult= base .* height; % Multiply base by height
add = base + height; % Add base and height
area = mult; % Calculate the area
perimeter = 2 * add; % Calculate the perimeter
volume = mult* depth; % Calculate the volume
end
```

1.13: Debugging in Matlab

- Matlab errors are very descriptive and provide specifics about error
 - If a function or script causes an error, Matlab will give the line of code and file with the error



```
Command Window
>> x=[1 2];
>> y=[3 5 7];
>> c=x+y
Error using +
Matrix dimensions must agree.

>> demoFunc
Error using demoFunc (line 5)
Not enough input arguments.
```

- The Matlab Editor provides on-the-fly debugging help!

```

1 - figure() % New Figure
2 - x1 = 0:1:2*pi; y1 = sin(x1); % First Data Set
3 - x2 = 0:0.01:2*pi; y2 = sin(x2); % Second Data Set
4 - plot(x1,y1,'sk',x2,y2,'r--') % Make Plot
5 - title('Some Sine Curves!') % Add Title, Labels, Legend, etc.
6 - xlabel('x')
7 - ylabel('sin(x)')
8 - legend('7 Data Points','629 Data Points','Location','NorthEast')
9
10

```

Green square
No errors or warning

```

1 function [area, perimeter] = dF(base, height)
2 depth = 10; % Assume 3D prism has depth of 10
3 mult= base .* height; % Multiply base by height
4 add = base + height; % Add base and height
5 area = mult; % Calculate the area
6 perimeter = 2 * add; % Calculate the perimeter
7 volume = mult* depth; % Calculate the volume
8
9

```

Orange Square
Warning present, but
code will still run
Indicated by orange bar

Mouse over for warning
message

- The Matlab Editor provides on-the-fly debugging help!

```

1 function [area, perimeter] = dF(base, height)
2 depth = 10; % Assume 3D prism has depth of 10
3 mult= base .* height; % Multiply base by height
4 add = base + height; % Add base and height
5 area = mult; % Calculate the area
6 perimeter = 2 * add; % Calculate the perimeter
7 volume = mult* depth; % Calculate the volume
8 error=error4
9
10

```

Red square
Error present and
Code will not run!
Indicated by red bar

Mouse over for error message

1.14: Advanced Features to Explore

Symbolic Math

- Allows for symbolic manipulation of equations, including solving, simplifying, differentiating, etc.

Inline Functions

- Creates a workspace variable that is a simple equation

```
>> f = x^2 + 2*x + 1
```

```
>> y = f(3) → y = 16
```

Optimization

- Solve constrained problems with **fmincon**, unconstrained with **fminunc**, bounded problems with **fminbnd**, etc.

Many Others!

- Matlab is extremely powerful and has a lot of advanced features, too many to go through here!
- Within Matlab:
 - Type **help function** to provide information about the function in the command window
 - Type **doc function** to open the documentation about the function
 - Type **doc** to pull up the documentation within Matlab to explore
- Online
 - Documentation: <http://www.mathworks.com/help/matlab/>
 - Tutorials: http://www.mathworks.com/academia/student_center/tutorials/
 - Matlab Primer / Getting Started with Matlab(pdf): http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

1.15: Descriptive statistics with the Statistics Toolbox of MATLAB

Some of the functions to compute the most frequent statistics are the following:

```

mean(x)           % Mean value of the elements in x.
median(x)         % Median value of the elements in x.
std(x),var(x)     % Standard deviation and variance of x normalized by n - 1.
std(x,1),var(x,1) % Standard deviation and variance of x normalized by n.
range(x)          % Range of x.
iqr(x)           % Interquartile range of x.
mad(x)           % Mean absolute deviation of x.
max(x),min(x)    % Maximum and minimum element of x.
skewness(x), kurtosis(x) % Skewness and kurtosis of x.
moment(x, order) % Central moment of x specified by order.
prctile(x,p)     % pth percentile of x (if p=50, returns the median of x)

```

Observe that if x is a matrix, then the result of these functions is a row vector containing the statistic for each column of x .

Other two interesting functions are **cov** and **corrcoef**. For vectors, the command **cov** returns the variance:

```
>> x=rand(100,1); cov(x)
```

For matrices, where each row is an observation, and each column a variable, returns the covariance matrix:

```
>> x=rand(100,5); cov(x)
```

For two vectors, z and w , of equal length, `cov(z , t)` returns a matrix with the variances of z and w in the diagonal and the covariance of z and w in the two off-diagonal entries.


```
>> z=rand(100,1); t=rand(100,1); cov(z , t)
```

Observe that $\text{cov}(z , t) = \text{cov}([z \ t])$. For two matrices,

$\text{cov}(X,Y)=\text{cov}(X(:),Y(:))$. Finally, $\text{cov}(x)$ or $\text{cov}(x,y)$ normalizes by $(n - 1)$ and $\text{cov}(x,1)$ or $\text{cov}(x,y,1)$ normalizes by n , where n is the number of observations.

The `corrcoef(X)` command calculates a matrix of correlation coefficients for an array X , in which each row is an observation and each column is a variable. Observe that $\text{corrcoef}(X,Y)$, where X and Y are column vectors, is the same as $\text{corrcoef}([X \ Y])$.

```
>> corrcoef(x)
```

The Statistics Toolbox and some built-in functions of MATLAB allows to plot a number of useful graphics in descriptive statistics.

```
hist(x)           % Histogram.
boxplot(x)        % Boxplots of a data matrix (one per column).
cdfplot(x)        % Plot of empirical cumulative distribution function.
normplot(x)       % Normal probability plot (one per column).
qqplot(x,y)       % Quantile-Quantile plot.
```

You can change the way any toolbox function works by copying and renaming the M-file, then modifying your copy. You can also extend the toolbox by adding your own M-files.

For example, imagine we are interested in plotting a variant of the histogram where the counts are replaced by the normalized counts, that is, the relative histogram. By normalized count, we mean the count in a class divided by the total number of observation times the class width. For this normalization, the area (or integral) under the histogram is equal to one. Now, we can look for the file `hist.m` and modify it. This file is usually in the following path (or something similar):

```
c:\MATLAB6p5\toolbox\matlab\datafun
```

Open it and let's try to change it. Observe that the `hist` command produces a histogram bar plot if there are no output arguments, that is, we look for the sentences:

```
if nargout == 0
bar(x,nn,'hist');
...
```

The sentence `bar(x,nn,'hist')` draws the values of the vector `nn` (frequency) as a group of vertical bars whose midpoints are the values of `x`, see `help bar`. For example, we can change the previous sentences by the following ones to obtain a white normalized histogram:

```
if nargout == 0
bar(x,nn/(length(y)*(x(2)-x(1))), 'hist', 'w');
...
```

You can also change the help section including for example a sentence like this:

```
% HIST(...) without output arguments produces a normalized histogram bar
```

`% plot of the results.`

And now, save the changed file as `histn.m`, for example. If you want `histn` to be a global function, you can save it in the same folder `histn.m` was. Otherwise, you can save it in a different folder and then `histn` will only work if you are in this directory or if you add it to the MATLAB's search path, (see `path`).

1.16: Simulation of linear models

The reporting of a simulation experiment should receive the same care and consideration that would be accorded the reporting of other scientific experiments. Hoaglin and Andrews (1975) outline the items that should be included in a report of a simulation study. In addition to a careful general description of the experiment, the report should include mention of the random number generator used, any variance-reducing methods employed, and a justification of the simulation sample size. The *Journal of the American Statistical Association* includes these reporting standards in its style guide for authors.

Closely related to the choice of the sample size is the standard deviation of the estimates that result from the study. The sample standard deviations actually achieved should be included as part of the report. Standard deviations are often reported in parentheses beside the estimates with which they are associated. A formal analysis, of course, would use the sample variance of each estimate to assess the significance of the differences observed between points in the design space; that is, a formal analysis of the simulation experiment would be a standard analysis of variance.

1.16.1: Simulation of simple linear model

Consider the simple linear regression model:

$$y_i = \beta_0 + \beta_1 x_i + E$$

Where a response or “dependent variable”, y , is modeled as a linear function of a single regressor or “independent variable”, x , plus a random variable, E , called the “error”. Because E is a random variable, y is also a random variable. The statistical problem is to make inferences about the unknown, constant parameters β_0 and β_1 and about distributional parameters of the random variable, E .

We also generally assume that the realizations of the random error are independent and are unrelated to the value of x .

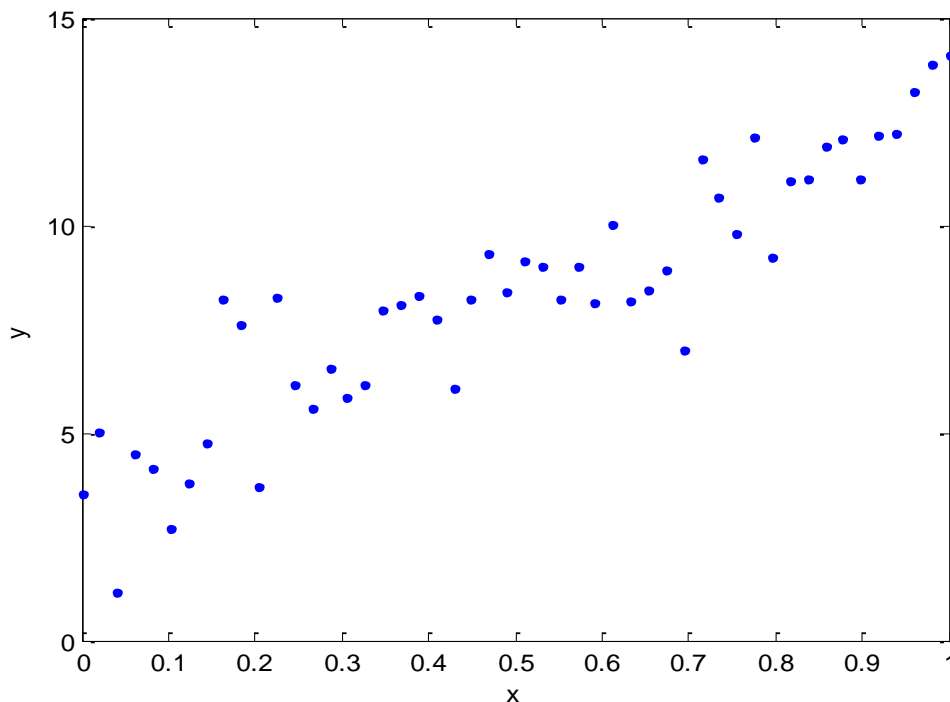
A bivariate scatter plot is a simple plot of x versus y between two variables. A bivariate scatter plot is a convenient first step to visualize the relationship between the two variables.

Assume that we have two variables that are linearly related, except some Gaussian noise term with mean 0 and standard deviation 1:

$$y = 3 + 10x + \text{noise}$$

Assuming that the variable x is a linearly spaced row vector of length 50, between 0 and 1, generate the y vector:

```
n=50; % number of observations
x=linspace(0,1,n); % linearly spaced vector a length n
beta0=3;
beta1=10;
E=randn(1,n);
y= beta0+beta1*x +E;
plot(x,y, '. ')
xlabel('x')
ylabel('y')
```



Each time the command is used, a different number will be generated. The “random” numbers generated by Matlab (and others) are actually pseudorandom numbers as they are computed using a deterministic algorithm. The algorithm, however, is very complicated, and the output does not appear to follow a predictable pattern. For this reason the output can be treated as random for most practical purposes. The same sequence of numbers will not be generated unless the same starting point is used. This starting point is called the “seed”. Each time you start Matlab, the random number generator is initialized to the same seed value. The current seed value can be seen using:

```
randn('seed',1) % specify a seed (optional)
```

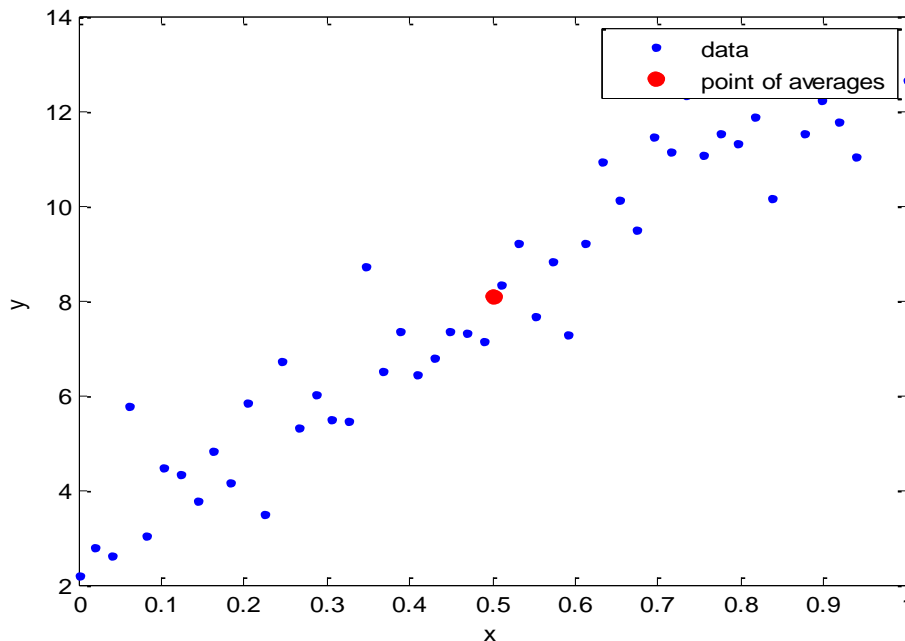
By setting a seed value, we ensure that the same results will be produced each time the script is executed. The seed can be set to a value (say, 1234) as follows:

```
randn('seed',1234)
```

The purpose here is to make sure that the program starts from the same seed. The value of the seed is not important.

In a bivariate scatter plot (x,y) , the point with coordinates $(\text{mean}(x), \text{mean}(y))$, is known as the point of averages.

```
mx=mean(x);
my=mean(y);
hold on;
plot(mx,my, 'ro', 'markerfacecolor','r')
legend('data', 'point of averages')
```



Covariance:

Covariance between vectors x and y can be computed in “unbiased” and “biased” versions as:

```
c= mean((x-mx).*(y-my)) % covariance (biased)
n=length(x);
cs= c*n/(n-1) % sample covariance(unbiased)
```

Ans:

$c = 0.85307$ $cs = 0.87048$

Correlation coefficient:

The correlation coefficient between two variables is a measure of the linear relationship between them. The correlation coefficient between two vectors can be found using the average of the product of the z-scores of x and y . The “biased” version is:

```
zx=zscore(x,1);
zy=zscore(y,1);
r=mean(zx.*zy)
```

Ans:

```
r =  
    0.94845
```

Correlation coefficient can also be computed from the covariance, as follows:

```
sx=std(x,1);  
sy=std(y,1);  
r=c/(sx*sy)
```

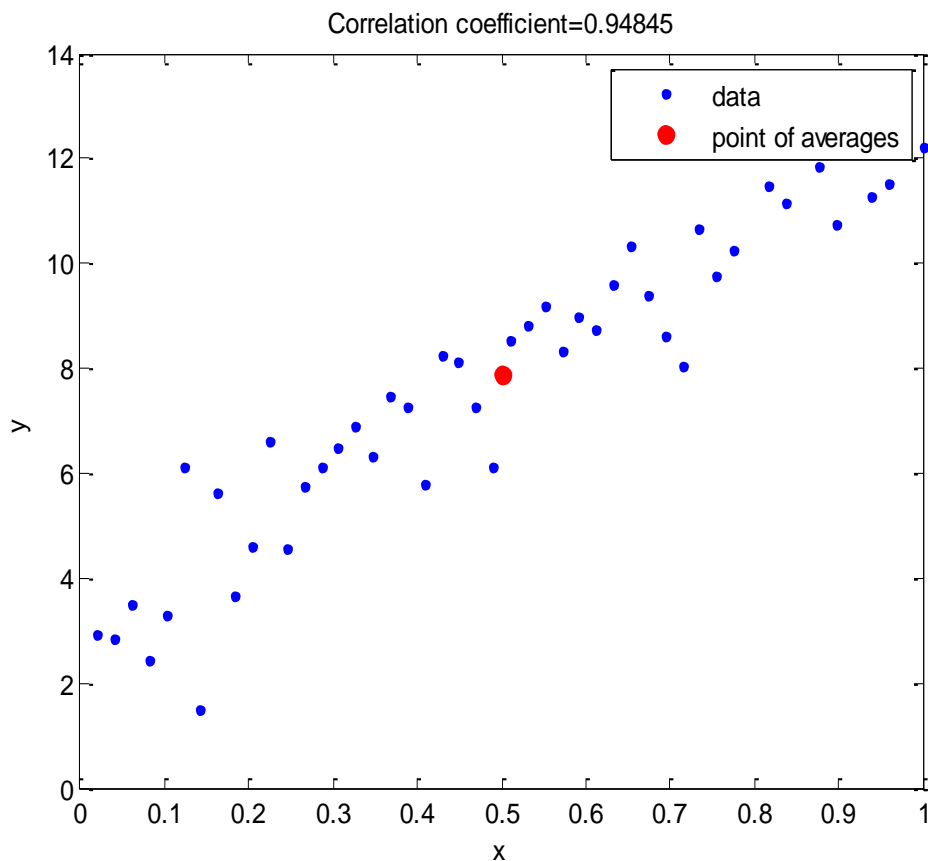
Ans:

```
r =  
    0.94845
```

The “unbiased” version (sample correlation coefficient) is computed the same way, except that the flag “1” is replaced by “0”.

Add a title that shows the correlation coefficient to the previous plot. For this, we need to convert the numerical value to a string, using the num2str command:

```
title(['Correlation coefficient=',num2str(r)])
```



The correlation coefficient is sensitive to outliers. To see this, change the first element of y to 40 and recompute the correlation coefficient:

```
y(1)=40;  
zx=zscore(x,1)
```

```
zy=zscore(y,1)
r=mean(zx.*zy)
```

Ans:

```
r =
    0.31003
```

Notice that a single outlier has significantly reduced the correlation coefficient.

1.16.2: Ordinary Least Squares Regression

Regression is a way to understand the mathematical relationship between variables. This relationship can then be used to

- Describe the linear dependence of one variable on another.
- Predict values of one variable from values of another.
- Correct for the linear dependence of one variable on another, in order to clarify other features of its variability.

Unlike the correlation coefficient, which measures the strength of a linear relationship, regression focuses on the mathematical form of the relationship.

In simple linear regression, the mathematical problem is as follows: Given a set of k points (x_i, y_i) , $i = 1, 2, \dots, k$, which are related through the equation $y_i = b_0 + b_1 x_i + n_i$, where b_0 and b_1 are constant (unknown) coefficients and n_i is a realization of zero-mean Gaussian noise with variance σ^2 . That is, $n_i \sim \mathcal{N}(0, \sigma^2)$. As the noise term n_i is a realization of a random variable, so is y_i . Because of the random noise, the coefficients b_0 and b_1 cannot be determined with certainty. Our goal is to find the best fit line $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$ minimizing the sum of squared errors:

$$S = \sum_{i=1}^k (y_i - \hat{y}_i)^2$$

The \hat{b}_1 and \hat{b}_0 values minimizing S are found by setting $\frac{\partial S}{\partial b_1} = 0$, $\frac{\partial S}{\partial b_0} = 0$. The result is:

$$\hat{b}_1 = \frac{\text{Covariance between } x \text{ and } y}{\text{Variance of } x}$$

$$\hat{b}_0 = (\text{mean of } y) - \hat{b}_1 (\text{mean of } x)$$

These \hat{b}_1 and \hat{b}_0 values are the Ordinary Least Square (OLS) estimates of b_1 and b_0 , respectively. The equation of the regression line (also known as the “best fit line”) is then $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$

```
bh1=c/sx^2;           % covariance divided by variance of x
bh0=my-bh1*mx;
yhat=bh0+bh1*x;     % regression line
```

Ans:

bh1 =

9.8354

bh0 =

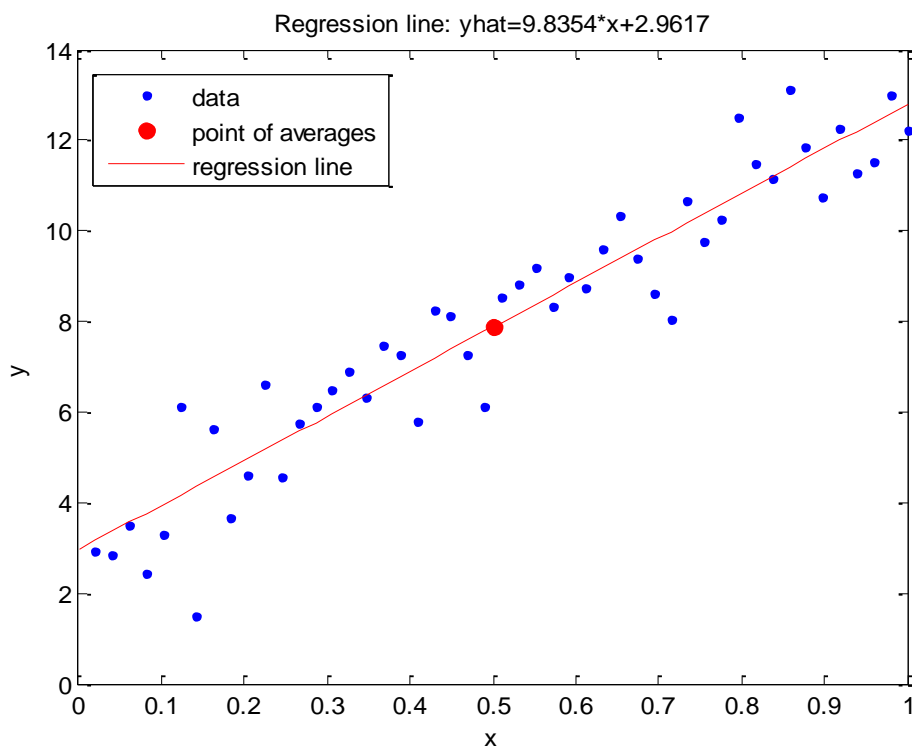
2.9617

Plot the regression line in red, and update the legend and the title:

```
plot(x,yhat,'r')
```

```
legend('data', 'point of averages','regression line')
```

```
title(['Regression line: yhat=',num2str(bh1),'*x+',num2str(bh0)])
```



Note that the regression line passes through the point of averages. The equation of the regression line shown in the title should be close to the original equation from which the data was generated:

$$y = 3 + 10x + \text{noise}$$

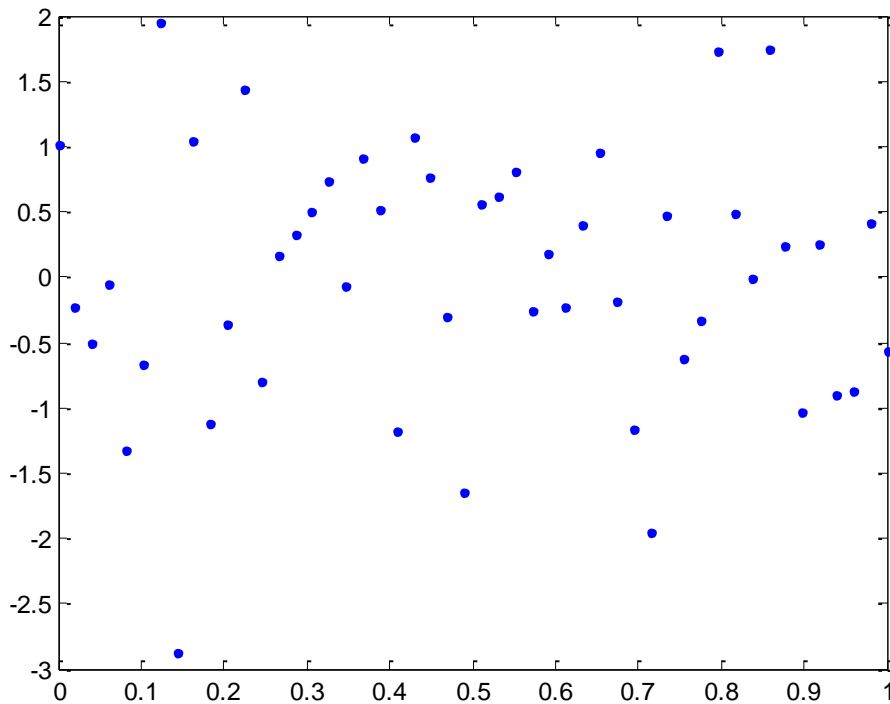
Because of the noise, the predictions will not exactly coincide with the observations. The residuals e_i are defined as the deviations of each observation from its estimate:

$$e_i = y_i - \hat{y}_i$$

```
e=y-yhat; %residuals
```

```
figure;
```

```
plot(x,e,'!')
```



Ideally, the residuals should be more or less symmetrically distributed around zero (have mean $\cong 0$):

`M = mean(e) % average residual`

Ans:

M =

-2.1583e-15

In addition, the amount of scatter should not show a systematic increase or decrease with increasing values of x . In other words, the scatter plot should be homoscedastic, not heteroscedastic. The variance of the noise can be estimated from the residuals (MSE) as follows:

$$\text{MSE} = \hat{\sigma}^2 = \frac{\sum_{i=1}^n e_i^2}{n-2}$$

`MSE = sum(e.^2)/(n-2) % OLS estimator of noise variance`

Ans:

MSE =

0.97588

The $n-2$ in the denominator is known as the “degrees of freedom”, and is computed by subtracting the number of parameters estimated (b_0 and b_1) from the number of observations.

The estimated noise variance for this particular problem should be close to 1, which is the variance of the noise used in generating the data.

The coefficient of determination (R^2) is a measure of how well the regression line represents the data. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad \text{where } \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

In simple linear regression, R^2 is equal to the square of the correlation coefficient (r^2) between x and y . If $r = 0.9$, then $R^2 = r^2 = 0.81$ which means that 81% of the total variation in y can be explained by the linear relationship between x and y . The other 19% of the total variation in y remains unexplained.

$R^2 = 1 - \text{sum}(e.^2) / \text{sum}((y - \text{my}).^2)$ % coefficient of determination
 $r^2 = r^2$ % correlation coefficient squared

Ans:

$R^2 =$
 0.89956
 $r^2 =$
 0.89956

Save the code as chapter1simsimple.m. This file will be used in future chapters.

1.16.3: Simple linear regression in matrix form

Consider the simple linear regression equation $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$.

Note that same equation can be written as $\hat{y}_i = [1 \quad x_i] \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \end{bmatrix}$.

This means that if the two coefficients are combined into a single column vector $\hat{\mathbf{b}} = \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \end{bmatrix}$, and the independent variable is augmented by adding a “1” to the front $\bar{x}_i = [1 \quad x_i]$, the i^{th} predicted value can be computed as $\hat{y}_i = \bar{x}_i \hat{\mathbf{b}}$. For the entire set of observations, we can write $\hat{\mathbf{Y}} = \mathbf{X} \hat{\mathbf{b}}$ where $\hat{\mathbf{Y}}$ is a column of predicted values, \mathbf{X} is the design matrix, where the first column consists of ones, the second column is the values of the independent variables, and $\hat{\mathbf{b}} = \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \end{bmatrix}$.

The OLS (ordinary least squares) estimate of the regression coefficients is given by $\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Y}$. Recall the simple linear regression data generated from

$y = 3 + 10x + \text{noise}$

$n = 50$;

$x = \text{linspace}(0, 1, n)$;

% linearly spaced vector a length n

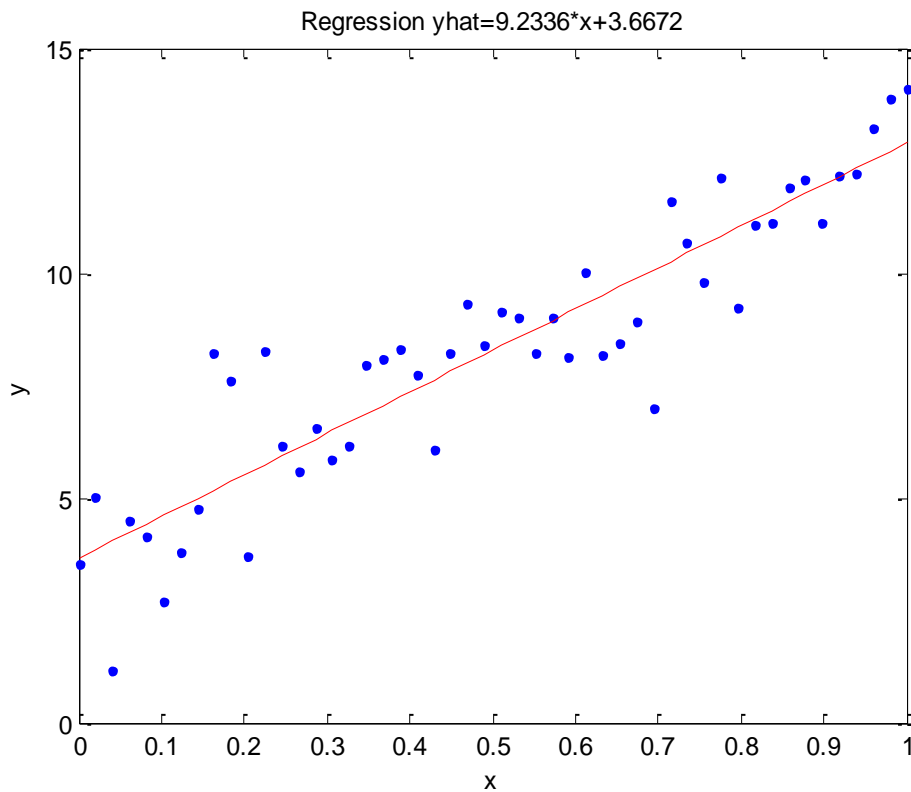
$y = 10 * x + 3 + \text{randn}(1, n)$;

$m_x = \text{mean}(x)$, $m_y = \text{mean}(y)$, $s_x = \text{std}(x, 1)$;

```

c= mean((x-mx).*(y-my))           %covariance
bh1=c/sx^2
bh0=my-bh1*mx
yhat=bh0+bh1*x;                   %regression line
figure;
plot(x,y,'.')
hold on
plot(x,yhat,'r')
xlabel('x'), ylabel('y')
title(['Regression yhat=',num2str(bh1),'*x+',num2str(bh0)])

```



The same estimates of the regression coefficients can be obtained using the matrix form:

```

x=x(:);                           % make x a column
y=y(:);                           % make y a column
XX=[ones(n,1),x];                 % create the design matrix
bh=(XX'*XX)^-1*XX'*y              % OLS estimate of b

```

Ans.

```

bh =
    3.6672
    9.2336

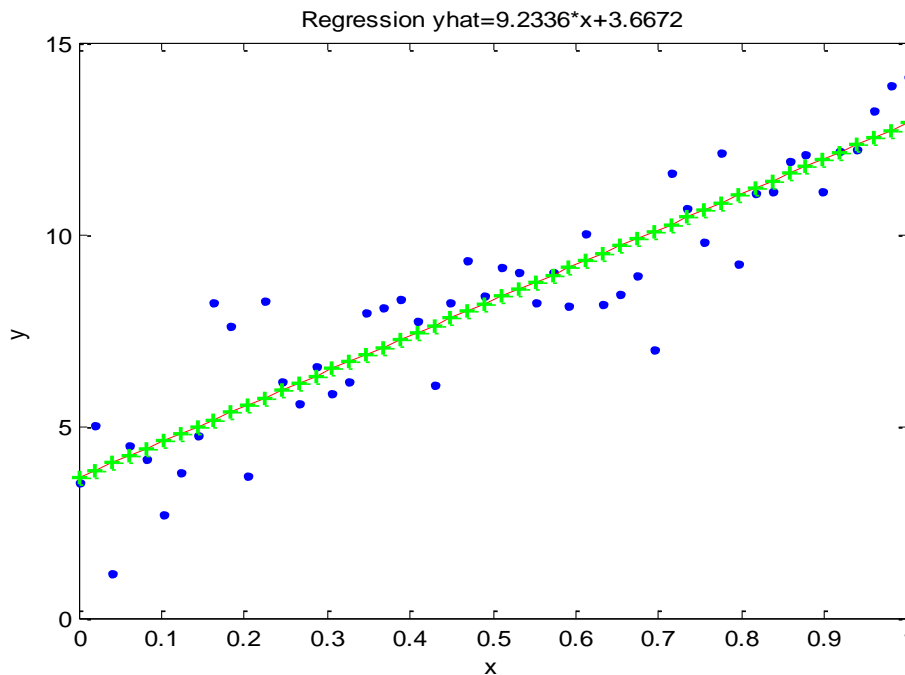
```

The \hat{b} vector should contain the previously computed b_0 and b_1 values. The new regression line should also coincide with the previous line.

```

yhat=XX*bh;
hold on
plot(x,yhat,'g+', 'linewidth',2)

```



The residuals and the estimated noise variance are computed as

```

e=y-yhat;           % residuals
dof= n-rank(XX);    % degrees of freedom
MSE=sum(e.^2)/dof   % estimated noise variance

```

Ans.

MSE=

1.5741

Save the code as SIMSIMPLEMATRIX.m. This file will be used in future chapters.

1.16.4: Multiple Linear Regression

In multiple linear regression, the regression equation is

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \dots + \hat{\beta}_k x_{ik}$$

And each observation is equal to the predicted value and a residual term e_i :

$$y_i = \hat{y}_i + e_i$$

The matrix-based analysis presented in the previous section is equally applicable to multiple independent variables. For each additional independent variable, another column is added to the design matrix, X. With k independent variables, the design matrix contains k+1 columns, the first column containing 1's. One difficulty with multiple independent variables is that the entire analysis cannot be summarized in a single figure, and the residuals need to be plotted with respect to each independent variable separately.

By using matrices, the multiple linear regression model, $Y = X\beta + \varepsilon$

Where $\varepsilon \sim N(0, \sigma^2 I_n)$ and Y is an $n \times 1$ vector of observations, X is an $n \times k$ matrix of regressors, β is a $n \times 1$ vector of parameters and ε is an $n \times 1$ vector of random disturbances. The least squares estimator of β is given by,

$$\hat{\beta} = (X'X)^{-1} X'Y$$

Whose variance is, $\text{Var}(\hat{\beta}) = \sigma^2 (X'X)^{-1}$

The predicted values are given by,

$$\hat{Y} = X\hat{\beta}$$

The residuals are,

$$e = Y - \hat{Y}$$

And the residual variance is,

$$\text{MSE} = \hat{\sigma}^2 = \frac{\sum_{i=1}^n e_i^2}{n - k - 1}$$

We can now define the following function to solve the regression problem:

The coefficient of determination (R^2) is computed the same way as in the simple linear case:

$$R^2 = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad \text{where } \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

The R^2 value in multiple linear regression is often called the “coefficient of multiple determination.”

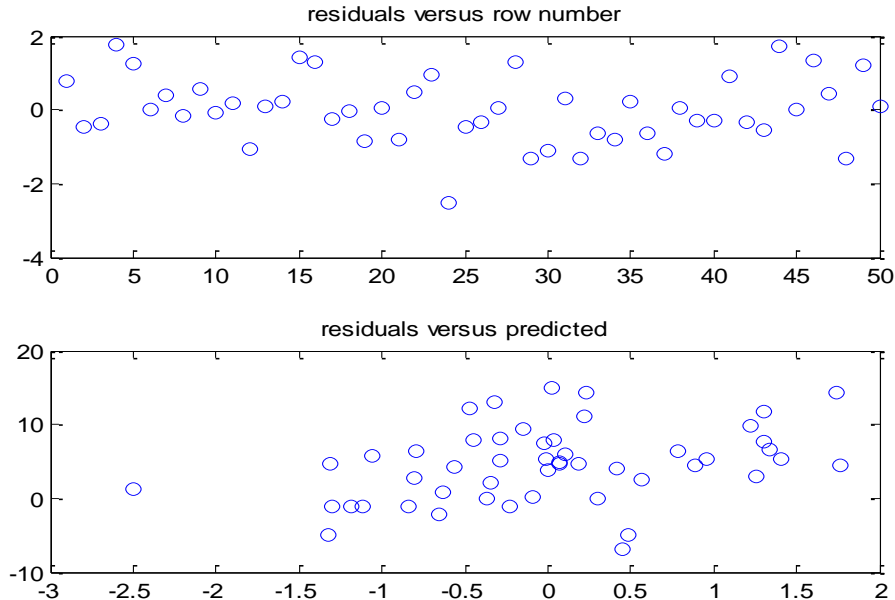
```
randn('seed',1234) % specify a seed (optional)
n = 50; k = 4;
X = [ones(n,1) randn(n,k)];
b = [5;1;2;3;4];
y = X*b + randn(n,1);[beta,Var_beta,resid,sR2] = regress(y,X)
MSE = sum(resid.^2)/(n - k - 1),Var_Cov_beta=inv(X'*X)*MSE
R2=1-sum(resid.^2)/sum((y-mean(y)).^2)
subplot(2,1,1),plot(resid,'o'),title('residuals versus row number')
subplot(2,1,2),plot(resid,ypred,'o'),title('residuals versus predicted')
```

Ans.

| beta | R2 | MSE |
|---------|---------|---------|
| 5.1611 | 0.96567 | 0.87179 |
| 0.78908 | | |
| 2.1569 | | |
| 2.9181 | | |
| 4.0902 | | |

Var_Cov_beta =

```
0.018533 -0.002176 -0.0023977 0.0011898 0.0028394
-0.002176 0.022436 0.0048211 0.0030259 -0.0016523
-0.0023977 0.0048211 0.020029 0.0066967 0.001665
0.0011898 0.0030259 0.0066967 0.016782 -0.0006353
0.0028394 -0.0016523 0.001665 -0.0006353 0.024338
```



Save the code as SIMMULTIPLEMATRIX.m. This file will be used in future chapters.

1.16.5: Multiple linear regression with the Statistics Toolbox of MATLAB

The Statistics Toolbox provides the regress function to address the multiple linear regression problems. regress uses QR decomposition of X followed by the backslash operator to compute $\hat{\beta}$. The QR decomposition is not necessary for computing $\hat{\beta}$, but the matrix R is useful for computing confidence intervals.

`b = regress(y,X)` returns the least squares estimator $\hat{\beta}$.

`[b, bint, r, rint, stats] = regress(y, X)` returns an estimate of β

Interval for β in the $k \times 2$ array bint. The residuals are returned in r and a 95% confidence interval for each residual is returned in the $n \times 2$ array rint. The vector stats contain the R^2 statistic along with the F and p values for the regression.

`[b,bint,r,rint,stats] = regress(y,X,alpha)` gives $100(1 - \alpha)\%$ confidence intervals for bint and rint. For example, `alpha = 0.2` gives 80% confidence intervals. Let's see

an example. Suppose the true model is, $Y = X \begin{pmatrix} 10 \\ 1 \end{pmatrix} + \varepsilon$, $\varepsilon \sim N(0, 0.01I_n)$

Where I is the identity matrix. Suppose we have the following data:

```
randn('seed',1234);n=10; X = [ones(n,1) (1:n)']
```

```
y = X * [5;2] + normrnd(0,0.1,n,1)
```

```
[b,bint] = regress(y,X,0.05)
```

| | X | | y |
|---|---|----|------------------|
| 1 | | 1 | 6.92063102736475 |
| 1 | | 2 | 8.95834974723594 |
| 1 | | 3 | 10.9217439183399 |
| 1 | | 4 | 13.2145703970460 |
| 1 | | 5 | 14.9213956160792 |
| 1 | | 6 | 17.0448135509485 |
| 1 | | 7 | 19.0098435509121 |
| 1 | | 8 | 20.9326093816663 |
| 1 | | 9 | 23.0200396628726 |
| 1 | | 10 | 24.9311656046701 |

b =

4.9845
2.0005

bint =

4.8304 5.1386
1.9757 2.0254

Compare b to [10 1]'. Note that bint includes the true model values.

Another example comes from Chatterjee and Hadi (1986) in a paper on regression diagnostics. The data set (originally from Moore (1975)) has five predictor variables and one response.

load **moore**

X = [ones(size(moore,1),1) moore(:,1:5)];

Matrix X has a column of ones, and then one column of values for each of the five predictor variables. The column of ones is necessary for estimating the y-intercept of the linear model.

y = moore(:,6);

[beta, beta_interval, resid, resid_interval, STATS] = regress(y,X)

Where **regress** Multiple linear regression using least squares.

beta = regress(Y,X) returns the vector beta of regression coefficients in the linear model $Y = X * beta$. X is an n-by-p design matrix, with rows corresponding to observations and columns to predictor variables. Y is an n-by-1 vector of response observations.

[beta, beta_interval] = regress(Y,X) returns a matrix beta_interval of 95% confidence intervals for beta.

[beta, beta_interval, resid] = regress(Y,X) returns a vector resid of residuals.

[beta, beta_interval, resid, resid_interval] = regress(Y,X) returns a matrix resid_interval of intervals that can be used to diagnose outliers. If RINT(i,:) does not contain zero, then the i-th residual is larger than would be expected, at the 5% significance level. This is evidence that the I-th observation is an outlier.

[beta, beta_interval, resid, resid_interval, STATS] = regress(Y,X) returns a vector STATS containing, in the following order, the R-square statistic, the F statistic and p value for the full model, and an estimate of the error variance.

Ans.

| beta | | beta_interval | |
|-------------|-----------|----------------|--|
| -2.1561 | -4.11538 | -0.19691 | |
| -9.0116e-06 | -0.00112 | 0.001103 | |
| 0.0013159 | -0.00139 | 0.004026 | |
| 0.0001278 | -3.71e-05 | 0.000293 | |
| 0.0078989 | -0.02213 | 0.037926 | |
| 0.00014165 | -1.65e-05 | 0.0003 | |
| resid | | resid_interval | |
| 0.562317 | 0.225802 | 0.898832 | |
| -0.14555 | -0.54763 | 0.256525 | |
| 0.088524 | -0.32617 | 0.50322 | |
| -0.04788 | -0.55146 | 0.455704 | |
| -0.2307 | -0.70433 | 0.242926 | |
| 0.170682 | -0.28023 | 0.621592 | |
| -0.34134 | -0.83769 | 0.155007 | |
| -0.07079 | -0.62602 | 0.484439 | |
| -0.01029 | -0.47488 | 0.454305 | |
| -0.10945 | -0.63998 | 0.421089 | |
| 0.171722 | -0.3311 | 0.674541 | |
| 0.050437 | -0.49066 | 0.591533 | |
| -0.03991 | -0.59383 | 0.514003 | |
| 0.022723 | -0.49909 | 0.544541 | |
| -0.39447 | -0.87015 | 0.081217 | |
| 0.081334 | -0.41688 | 0.579544 | |
| 0.072986 | -0.08787 | 0.233845 | |
| 0.011354 | -0.4987 | 0.521405 | |
| -0.22227 | -0.66763 | 0.223093 | |
| 0.380568 | -0.00711 | 0.768246 | |

STATS =

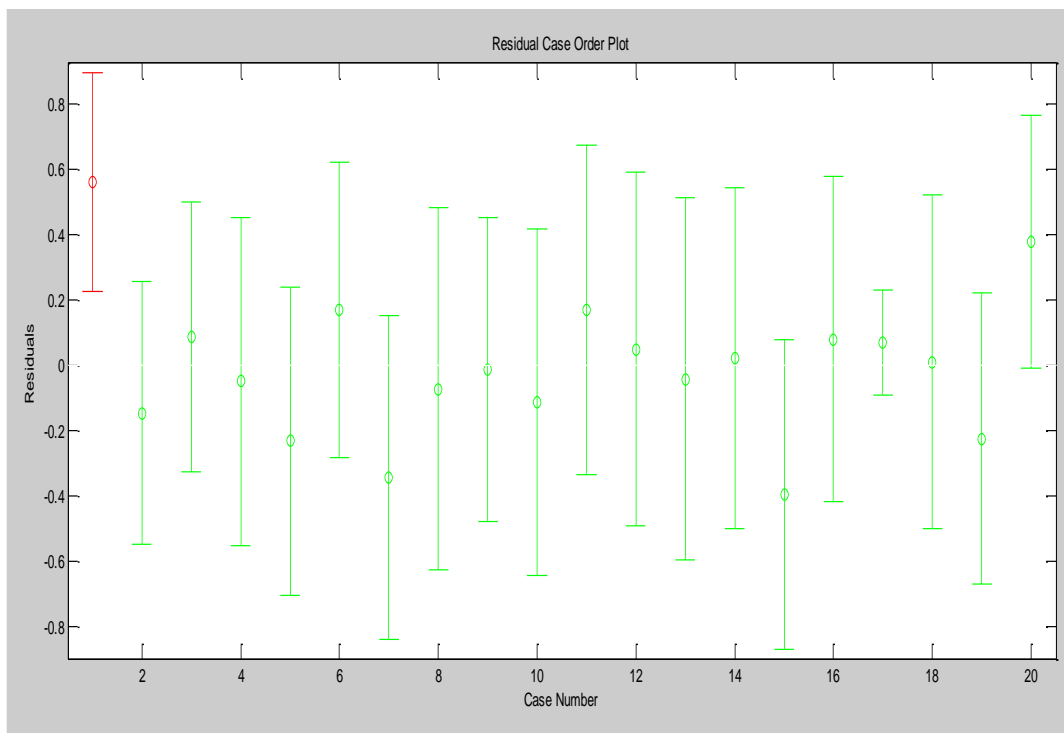
| R^2 | F | p-value | error variance |
|----------|----------|----------|----------------|
| 0.810665 | 11.98861 | 0.000118 | 0.068538 |

The y-intercept is b_0 , which corresponds to the column index of the column of ones.

The elements of the vector stats are the regression R^2 statistic, the F statistic (for the hypothesis test that all the regression coefficients are zero), the p-value associated with this F statistic, and error variance

R^2 is 0.8107 indicating the model accounts for over 80% of the variability in the observations.

The F statistic of about 12 and its p-value of 0.0001 indicate that it is highly unlikely that all of the regression coefficients are zero.



The plot shows the residuals plotted in case order (by row). The 95% confidence intervals about these residuals are plotted as error bars. The first observation is an outlier since its error bar does not cross the zero-reference line. [The program name: CONFIDENC]

1.17: Simulation of Stochastic processes

In this section, we will simulate and represent graphically various simple stochastic processes.

1.17.1: Simulation of Bernoulli process

A Bernoulli process is a discrete-time stochastic process consisting of finite or infinite sequence of independent random variables x_1, x_2, x_3, \dots such that,

$$x_i = \begin{cases} 1, & \text{with } prop = p \\ -1, & \text{with } prop = 1 - p \end{cases}$$

Random variables associated with the Bernoulli process include:

- The number of successes in the first n trials; this has a binomial distribution;
- The number of trials needed to get r successes; this has a negative binomial distribution.
- The number of trials needed to get one success; this has a geometric distribution, which is a special case of the negative binomial distribution.

We can simulate a realization of size 100 of a Bernoulli process with $p = 0.5$ as follows.

```
u=rand(10,1);
```

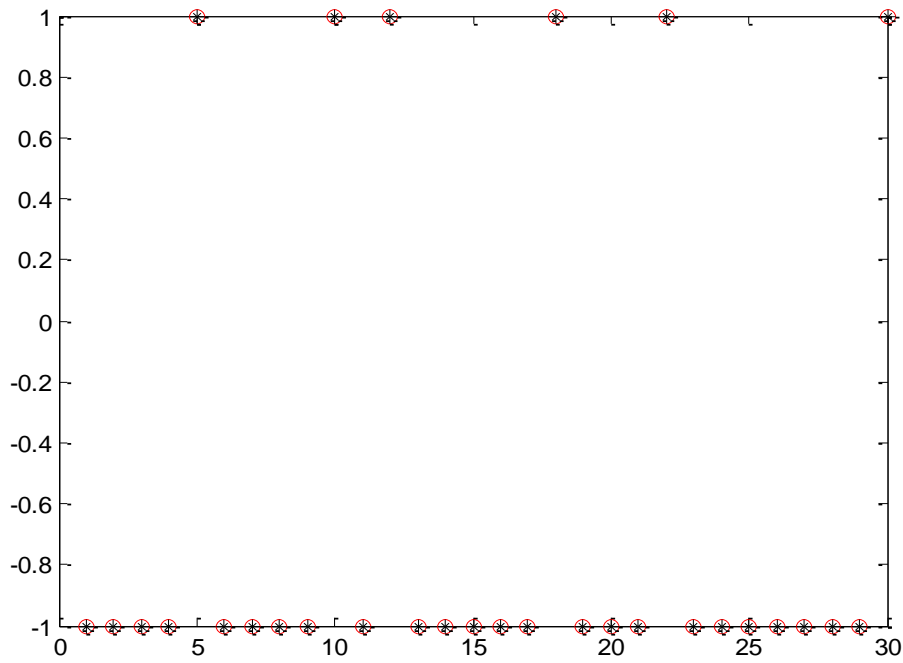
```
X=1-2*floor(u*2)
```

Where (floor) Round towards minus infinity,

`floor(X)` rounds the elements of `X` to the nearest integers towards minus infinity.

We can simulate another realization of a Bernoulli process with $p = 0.25$ and observe the differences. [The program name `BERNOULLI.m`]

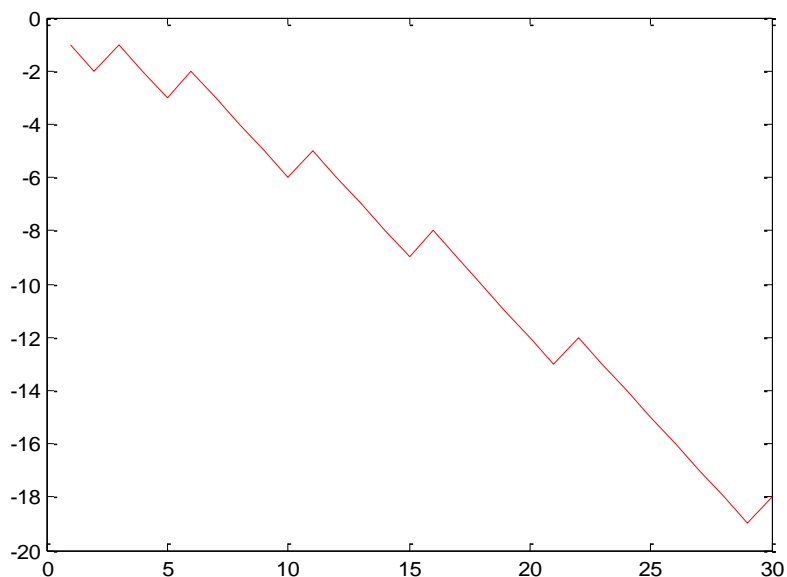
```
u=rand(30,1);  
Y(u<0.25)=1;Y(u>0.25)=-1;  
plot(1:30,Y,'ro',1:30,Y,'k*')
```



1.17.2: Simulation of Random walk

By using the `cumsum` command, we can simulate random walks from the Bernoulli processes simulated previously. [The program name `RANDOMWALK.m`].

```
u=rand(30,1);  
Y(u<0.25)=1;Y(u>0.25)=-1;  
plot(1:30,cumsum(Y),'r')
```



1.17.3: Simulation of Poisson process

Firstly, observe that continuous time processes are only possible to simulate by discretization of the unit time.

A Poisson process, x_t , with rate λ verifies the following property:

$$x_t = \text{Number of occurrences in } [0, t) \sim \text{Po}(\lambda t).$$

If we want to simulate a realization with 10 occurrences from a Poisson process of rate $\lambda = 2$, we can first simulate 10 exponential times of mean $1/\lambda = 0.5$ between occurrences. [The program name POISSONPROCES.m].

```
x=exprnd(0.5,1,10);
```

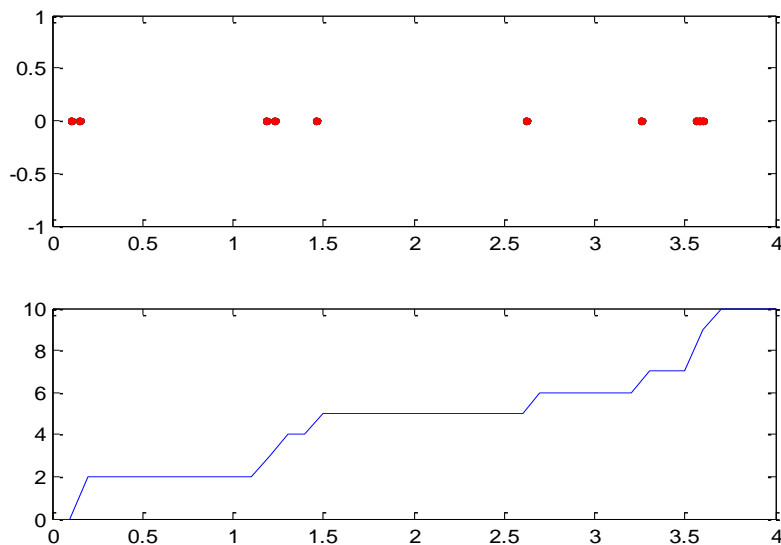
Then, we can obtain the occurrence times as follows.

```
x=cumsum(x);  
subplot(2,1,1),plot(x,zeros(length(x)),'.')
```

Suppose we want to know the value of the process x_t at the following instant times:

Then, we can compute:

```
for i=1:length(t);X(i)=sum(x<t(i));end  
subplot(2,1,2),plot(t,X)
```



1.17.4: Simulation of Autoregressive process

Suppose we want to simulate $T = 100$ values from an autoregressive model AR(1),

$$x_t = \alpha x_{t-1} + e_t$$

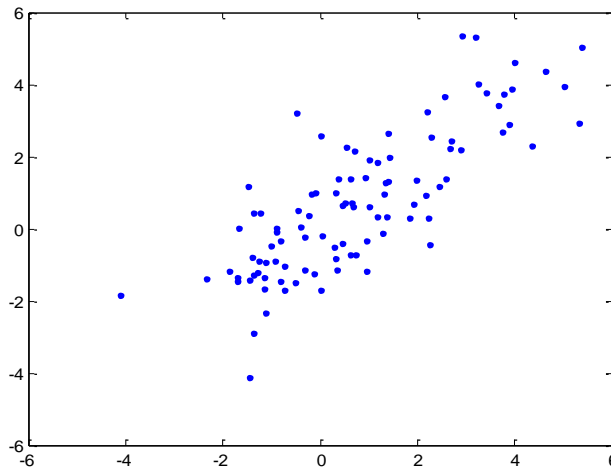
where e_t are i.i.d. $N(0, 1)$ and assume three values for $\alpha \in \{0.8, 0.5, -0.8\}$. One possibility is to assume $x_1 = e_1$ and then obtain recursively the remaining values. [The program name AR1.m].

```
e=randn(100,1);  
x=zeros(100,1);
```

```
x(1)=e(1);
alpha=0.8;
for i=2:100, x(i)=alpha*x(i-1)+e(i); end
```

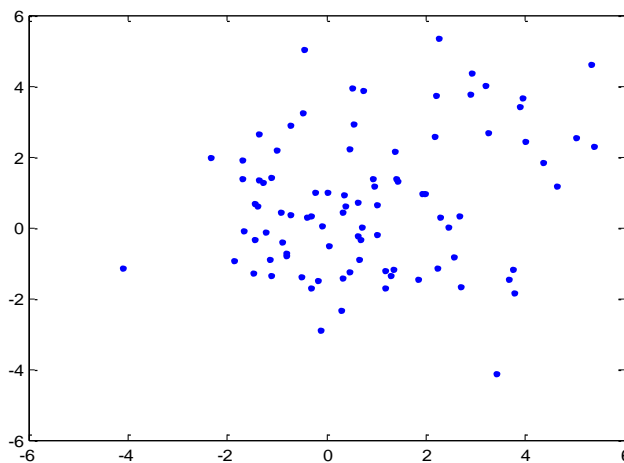
We can calculate the sample coefficient of the autocorrelation function. For example, the first coefficient is the sample correlation coefficient of x_{t-1} and x_t :

```
corrcoef(x(1:99),x(2:100));
plot(x(1:99),x(2:100),'.'
```



Observe that after 10 lags, there is almost no relation between x_{t-1} and x_t :

```
plot(x(1:90),x(11:100),'.'
```



1.17.5: Simulation of Moving average process

Suppose now that we want to simulate $T = 100$ values from a moving average model MA(1),

$$x_t = \theta e_{t-1} + e_t$$

Where e_t are i.i.d. $N(0, 1)$ and assume three values for $\theta \in \{0.8, 0.5, -0.8\}$. [The program name MA1.m].

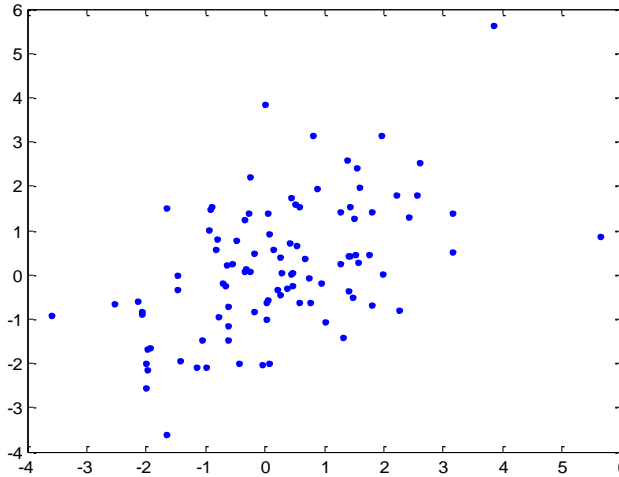
This process is easier to initialize because we just have to simulate e_0 .

```
e=randn(101,1);
```

```
theta=0.8;
x=theta*e(1:100,1)+e(2:101,1);
```

Compute the first two coefficients of the autocorrelation function and observe the following plots:

```
plot(x(1:99),x(2:100),'.'')
plot(x(1:98),x(3:100),'.'');
```



1.18: Nonlinear Regression

When the relationship between the independent variable(s) and the dependent variable cannot be approximated as a line (or a hyperplane), approaches beyond linear regression are needed. There are many different methods for dealing with nonlinear relationships, but we will focus on two approaches: (a) Using a nonlinear transformation which makes the data approximately linear; (b) Polynomial fitting.

1.18.1: Nonlinear Transformations

Sometimes a non-linear relationship can be transformed into a linear one by a mathematical transformation. Examples include the exponential growth equation:

$$y = A e^{bx} u \Leftrightarrow \log(y) = \log(A) + bx + \log(u)$$

And the constant-elasticity equation

$$y = A x^b u \Leftrightarrow \log(y) = \log(A) + b \cdot \log(x) + \log(u)$$

Linear regression can now be performed using the transformed variables.

Example 1.8: The table below shows data to test the relationship between porosity and sandstone strength.

| x=porosity | y=unconfined strength (psi) | Source: Hale, P. A. & Shakoor, A., 2003, A laboratory investigation of the Effects of Cyclic Heating and Cooling, Wetting and Drying, and Freezing and Thawing on the Compressive Strength of Selected Sandstones: Environmental and Engineering geoscience, vol IX, p. 117-130. |
|------------|-----------------------------|--|
| 12.32 | 2636 | |
| 13.94 | 3162 | |
| 6.94 | 7580 | |
| 4.0 | 16899 | |
| 2.94 | 23739 | |
| 0.86 | 14224 | |

Plot the data and the regression line, and compute the coefficient of determination. [The program name example118.m].

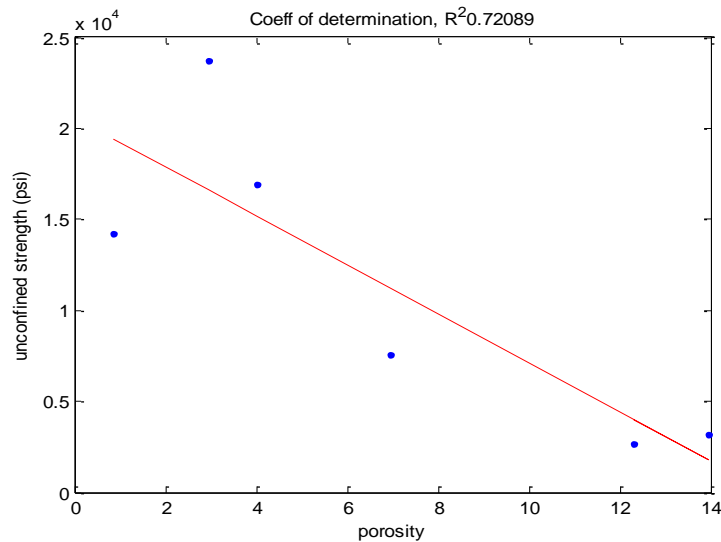
```
x=[12.32,13.94,6.94,4,2.94,0.86];
y=[ 2636, 3162, 7580, 16899, 23739, 14224];
x=x(:); y=y(:);
n=length(x);
XX=[ones(n,1),x];
b=(XX'*XX)^-1*XX'*y
yhat=XX*b;
e=y-yhat;
my=mean(y);
R2=1-sum(e.^2)/sum((y-my).^2)
figure;
plot(x,y,'b');
hold on , plot(x,yhat,'r')
title(['Coeff of determination, R^2' ,num2str(R2)])
xlabel('porosity'), ylabel('unconfined strength (psi)')
MSE=sum(e.^2)/(n-2)
```

Ans.

b =
20560
-1344.4

R2 =
0.72089

MSE =
2.4403e+07

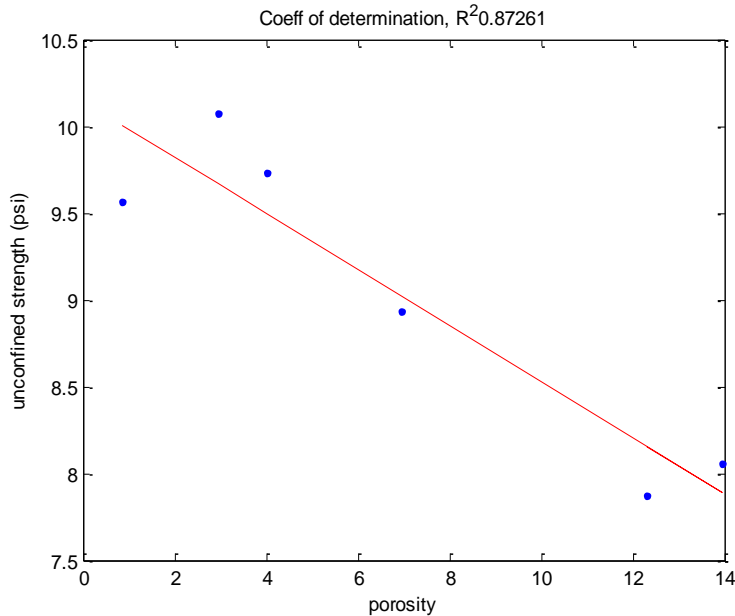


The coefficient of determination is $R^2 = 0.72$, indicating that the regression equation can explain 72% of the variation in unconfined strength. And MSE equals 2.4403e+07

Repeat the same analysis, using a nonlinear transformation: [The program name example118.m].

```
y=log(y)
```

$b =$ $R^2 =$ $MSE =$
 10.142 0.87261 0.13228
 -0.1612



The coefficient of determination has increased to $R^2 = 0.87$ and MSE has decreased to 0.13228

There are a few points to keep in mind when using this method. First, we are assuming that the errors in the transformed equation follow a zero-mean Gaussian distribution, which may not be a reasonable assumption. Second, once we get the estimates from the transformed equation, going back to the original equation can be tricky. Some parameter estimates are biased, and the confidence intervals are no longer symmetrical around the predicted values. We need to get the confidence interval from the transformed equation and then transform the bounds back.

1.18.2: Polynomial fitting

The commands **polyfit** and **polyval** can be used whenever the data can be approximated by a polynomial.

1- **polyfit** Fit polynomial to data.

$P = \text{polyfit}(X, Y, N)$ finds the coefficients of a polynomial $P(X)$ of degree N that fits the data Y best in a least-squares sense. P is a row vector of length $N+1$ containing the polynomial coefficients in descending powers,

$$P(1)*X^N + P(2)*X^{(N-1)} + \dots + P(N)*X + P(N+1).$$

$[P, S] = \text{polyfit}(X, Y, N)$ returns the polynomial coefficients P and a structure S for use with **POLYVAL** to obtain error estimates for predictions. S contains fields for the triangular factor (R) from a QR decomposition of the Vandermonde matrix of X , the degrees of freedom (df), and the norm of the residuals ($normr$). If the data Y are random, an estimate of the covariance matrix of P is $(R_{\text{inv}}*R_{\text{inv}}')*normr^2/df$, where R_{inv} is the inverse of R .

[P,S,MU] = polyfit(X,Y,N) finds the coefficients of a polynomial in

XHAT = (X-MU(1))/MU(2) where MU(1) = MEAN(X) and MU(2) = STD(X). This centering and scaling transformation improves the numerical properties of both the polynomial and the fitting algorithm.

Warning messages result if N is \geq length(X), if X has repeated, or nearly repeated, points, or if X might need centering and scaling.

Class support for inputs X,Y: float: double, single

2- **polyval** Evaluate polynomial.

Y = polyval(P,X) returns the value of a polynomial P evaluated at X. P is a vector of length N+1 whose elements are the coefficients of the polynomial in descending powers.

$$Y = P(1)*X^N + P(2)*X^{(N-1)} + \dots + P(N)*X + P(N+1)$$

If X is a matrix or vector, the polynomial is evaluated at all points in X. See POLYVALM for evaluation in a matrix sense.

[Y,DELTA] = polyval(P,X,S) uses the optional output structure S created by POLYFIT to generate prediction error estimates DELTA. DELTA is an estimate of the standard deviation of the error in predicting a future observation at X by P(X).

If the coefficients in P are least squares estimates computed by POLYFIT, and the errors in the data input to POLYFIT are independent, normal, with constant variance, then Y +/- DELTA will contain at least 50% of future observations at X.

Y = polyval(P,X,[],MU) or [Y,DELTA] = polyval(P,X,S,MU) uses XHAT = (X-MU(1))/MU(2) in place of X. The centering and scaling parameters MU are optional output computed by POLYFIT.

Consider the following nonlinear system:

```
randn('seed', 1);  
x=(1:50)';  
y = sin(x/50)./ x + 0.002 * randn(50,1)
```

Fit a polynomial of order 5:

```
order=5;  
poly = polyfit(x, y, order);
```

Evaluate the polynomial at the data points:

```
yhat= polyval(poly,x)
```

An approximate 95% prediction interval for y (including the noise) can be constructed as follows: [The program name NONLINEAR.m].

```
randn('seed', 1);  
x=(1:50)'; y = sin(x/50)./ x + 0.002 * randn(50,1); n=length(x); order=5; poly =  
polyfit(x, y, order); yhat= polyval(poly,x)  
[poly model] = polyfit(x, y, order); % fit a polynomial
```

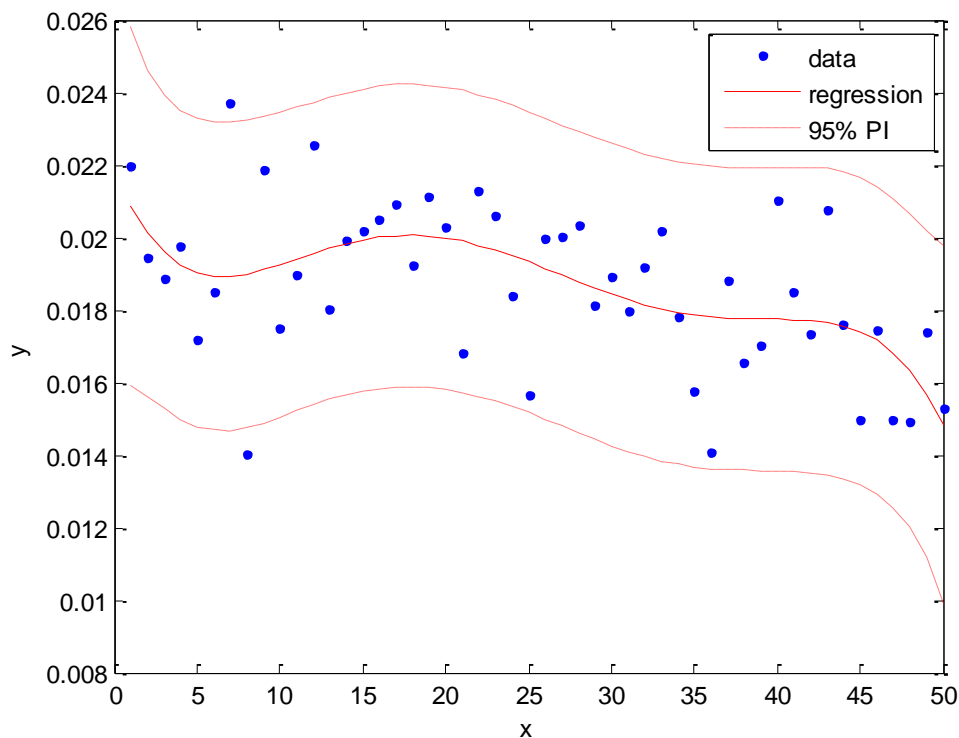
```

[yhat s] = polyval(poly, x, model); % evaluate the polynomial
alpha=0.05; % for 95% confidence
p=1-alpha/2; % probability to be used in CDF
df=50-(5+1); % degrees of freedom
t=tinv(p,df); % t-value, may need tinv558
PI_lower=yhat-t*s; PI_upper=yhat+t*s;
figure;
plot(x,y,'b')
hold on
plot(x,yhat, 'r')
plot(x, PI_lower, 'r:')
plot(x, PI_upper, 'r:')
legend('data','regression','95% PI')
xlabel('x'), ylabel('y')
my=mean(y); e=y-yhat;
MSE=sum(e.^2)/(n-2); R1=1-sum(e.^2)/sum((y-my).^2)

```

Ans.

MSE = 0.30322, R2 = 3.616e-06



PROBLEMS

1.1: Define MATLAB

1.2: What is interest MATLAB?

1.3: where the name came from MATLAB?

1.4: What MATLAB language characterized for other programming languages?

1.5: What magic matrix and how do we get them?

1.6: In analyzing linear equations if you know that:

$$A = \begin{bmatrix} 9 & 4 & 1 \\ 8 & 5 & 2 \\ 6 & 3 & 4 \end{bmatrix}$$

Find the following:

- 1- The inverse of the matrix.
- 2- Cholesky factorization.
- 3- Upper and lower trigonometric matrix.
- 4- Pseudoinverse matrix.

1.7: In the analysis of the Eigenvalues if you know that:

$$B = \begin{bmatrix} 3 & 4 & 1 \\ 5 & 7 & 8 \\ 1 & 2 & 1 \end{bmatrix}$$

A- Eigen values and Eigen vector.

B- Singular value decomposition.

1.8: Analysis functions of matrices if you know that:

$$C = \begin{bmatrix} 5 & 1 & 2 & 4 \\ 6 & 2 & 5 & 1 \\ 4 & 3 & 1 & 5 \\ 8 & 9 & 3 & 2 \end{bmatrix}$$

Find the following:

- 1- Matrix exponential
- 2- Matrix logarithm
- 3- Matrix square root

1.9: Explain the command Kronecker with a practical example?

1.10: Solving linear systems following:

$$A * X = B$$

If you know that A represents Pascal matrix (Dim.3) and

$$B = [3 \quad 1 \quad 4]^T$$

1.11: Estimate and draw the negative exponential model using (OLS) method for the following data:

$$t = [0 \ .3 \ .8 \ 1.1 \ 1.6 \ 2.3]' \text{ and } y = [.82 \ .72 \ .63 \ .60 \ .55 \ .50]'$$

$$\text{Where } y(t) = c_1 + c_2 e^{-t}$$

1.12: Estimate the Simple Linear Model using method (OLS) for the following data:

| | | | | | | | | |
|---|---|----|----|----|----|----|----|----|
| y | 2 | 3 | 5 | 7 | 8 | 10 | 12 | 15 |
| x | 8 | 10 | 14 | 16 | 17 | 20 | 22 | 26 |

$$\text{Where } y_i = c_1 + c_2 x_i$$

Find the following:

- 1- Average of D.V.
- 2- Variance of I.V.
- 3- Standard Deviation of the D.V.
- 4- Simple Linear Correlation Coefficient.
- 5- Mean Square Error.
- 6- The Coefficient of Determination.
- 7- Standard Error.
- 8- Covariance between the I.V. and D.V.

1.13: Draw the scatter plot of the following data:

$$z = [0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7]$$

$$x = [3 \ 5 \ 7 \ 9 \ 11 \ 13 \ 15 \ 17]$$

$$y = [1 \ 6 \ 8 \ 12 \ 16 \ 20 \ 24 \ 29]$$

1.14: Select outlier values for the following data:

$$z = [0 \ 1 \ 2 \ 3 \ 20 \ 5 \ 6 \ 7]$$

$$x = [3 \ 5 \ 7 \ 9 \ 11 \ 13 \ 15 \ 17]$$

$$y = [1 \ 6 \ 8 \ 12 \ 16 \ 20 \ 24 \ 29]$$

1.15: Estimate the Multiple Linear Model using method (OLS) for the following data:

$$\text{Where } y_i = c_1 + c_2 x_i + c_3 z_i$$

The required account the following:

- 1 - Average of D.V.
 - 2 - Variance of x.
 - 3 - Mean Square Error.
 - 4- Standard Error.
 - 5 - Covariance between the variables.
- 1.16: Write a computer program to implement for generating a F-distribution with (8) & (11) degrees of freedom respectively, for $n = 30$
- 1.17: Write a computer program to implement for generating a Exp(6) random deviate, $n = 20$

- 1.18: Compute possible some cases Normal output matrix of random matrix generated from Uniform distribution (3×2) multiplied by 10 for just the integer values.
- 1.19: Write a computer program to implement for generating a t -distribution with (20) degree of freedom, for $n = 25$ by using Direct Method.
- 1.20: Write a computer program to implement for generating a multivariate normal distribution for ($k=4$) variables, $n=30$ and:

$$\Sigma = \begin{bmatrix} 1 & 1 & 1 & 1 \\ & 2 & 3 & 4 \\ & & 6 & 10 \\ & & & 20 \end{bmatrix}$$

For means equal to [2 15 6 12], find mean, variance and correlation matrix.

- 1.21: Write a computer program to implement for generating:
- Poisson(5) random deviate, $n = 20$
 - Exp(2) random deviate, $n = 10$

References

Ali Taha Hussein Ali, 2018, Solving Multi-collinearity Problem by Ridge and Eigen value Regression with Simulation, *Journal of Humanity Sciences*, 22.5: 262-276.

Ali, Taha Hussein & Awaz shahab M. "Uses of Waveshrink in detection and Treatment of Outlier Values in Linear Regression analysis and comparison With Some Robust Methods", *Journal of Humanity Sciences* 21.5 (2017): 38-61.

Ali, Taha Hussein & Jalil T. S. "The Construction of Bayes Chart to control qualitative attributes", *Journal of Tanmiyat Al-Rafidain*, 14.2 (1998): 56-78.

Ali, Taha Hussein & Jalil T. S. "Construction Quality Control Charts using Bayes approach" *Zanko Scientific Journal*, 20 (2003): 87-105

Ali, Taha Hussein & Kurdistan L. Mawlood. "Dealing with the Contamination and Heterosedasticity Problems In the CRD by Using the Wavelet Filter" *Iraqi Journal of Statistical Sciences*, 18 (2010): 237-258.

Ali, Taha Hussein & Mardin Samir Ali. "Analysis of Some Linear Dynamic Systems with Bivariate Wavelets" *Iraqi Journal of Statistical Sciences* 16.3 (2019): 85-109.

Ali, Taha Hussein & Qais Mustafa. "Reducing the orders of mixed model (ARMA) before and after the wavelet de-noising with application." *Journal of Humanity Sciences* 20.6 (2016): 433-442.

Ali, Taha Hussein and Jwana Rostam Qadir. "Using Wavelet Shrinkage in the Cox Proportional Hazards Regression model (simulation study)", *Iraqi Journal of Statistical Sciences*, 19, 1, 2022, 17-29.

Ali, Taha Hussein, "Estimation of Multiple Logistic Model by Using Empirical Bayes Weights and Comparing it with the Classical Method with Application" *Iraqi Journal of Statistical Sciences* 20 (2011): 348-331.

Ali, Taha Hussein, and Dlshad Mahmood Saleh. "COMPARISON BETWEEN WAVELET BAYESIAN AND BAYESIAN ESTIMATORS TO REMEDY CONTAMINATION IN LINEAR REGRESSION MODEL" *PalArch's Journal of Archaeology of Egypt/Egyptology* 18.10 (2021): 3388-3409.

Ali, Taha Hussein, and Saleh, Dlshad Mahmood, "Proposed Hybrid Method for Wavelet Shrinkage with Robust Multiple Linear Regression Model: With Simulation Study" *QALAAI ZANIST JOURNAL* 7.1 (2022): 920-937.

Ali, Taha Hussein, Avan Al-Saffar, and Sarbast Saeed Ismael. "Using Bayes weights to estimate parameters of a Gamma Regression model." *Iraqi Journal of Statistical Sciences* 20.20 (2023): 43-54.

Ali, Taha Hussein, Heyam Abd Al-Majeed Hayawi, and Delshad Shaker Ismael Botani. "Estimation of the bandwidth parameter in Nadaraya-Watson kernel non-parametric regression based on universal threshold level." *Communications in Statistics-Simulation and Computation* 52.4 (2023): 1476-1489. <https://doi.org/10.1080/03610918.2021.1884719>

Ali, Taha Hussein, Mohammed Abdul Majeed Badal, & Awaz shahab M. (2018), "Use proposed methods for estimating the Ridge Regression parameter and comparing it with some classical methods" The 6th International Conference of Union if Arab Statistician, pp.297-309.

Ali, Taha Hussein, Mohammed Abdul Majeed Badal, & Safaa S. "Estimations of AR(p) Model using Wave Shrink" *Iraqi Journal of Statistical Sciences*, 17 (2010): 97-114.

Ali, Taha Hussein, Nasradeen Haj Salih Albarwari, and Diyar Lazgeen Ramadhan. "Using the hybrid proposed method for Quantile Regression and Multivariate Wavelet in estimating the linear model parameters." *Iraqi Journal of Statistical Sciences* 20.20 (2023): 9-24.

Ali, Taha Hussein, Nazeera Sedeek Kareem, and mohammad, Awaz Shahab "Construction robust simple linear regression profile Monitoring" *journal of kirkuk University for Administrative and Economic Sciences*, 9.1. (2019): 242-257.

Ali, Taha Hussein, Rahim, Alan Ghafur, and Saleh, Dlshad Mahmood. "Construction of Bivariate F-Control Chart with Application" *Eurasian Journal of Science & Engineering*, 4.2 (2018): 116-133.

Ali, Taha Hussein, Saman Hussein Mahmood, and Awat Sirdar Wahdi. "Using Proposed Hybrid method for neural networks and wavelet to estimate time series model." *Tikrit Journal of Administration and Economics Sciences* 18.57 part 3 (2022).

Ali, Taha Hussein. "Modeling Markov chains of Bernoulli processes using the Bayesian approach" *Zanko Scientific Journal*, 26 (2003): 239-260

Ali, Taha Hussein. "Modification of the adaptive Nadaraya-Watson kernel method for nonparametric regression (simulation study)." *Communications in Statistics-Simulation and Computation* 51.2 (2022): 391-403.

Ali, Taha Hussein. "The Construction of Bayes Chart of Single Value to Control Marble Pressure in Erbil Factory", *Journal of Tanmiyat Al-Rafidain*, 85.29 (2007): 29-48.

Ali, Taha Hussein. "The Sequential Bayesian Approach for Poisson Processes", *Journal of Tanmiyat Al-Rafidain*, 75.26 (2004): 83-99.

Ali, Taha Hussein; Esraa Awni Haydier. "Using Wavelet in constructing some of Average Charts for Quality control with application on Cubic Concrete in Erbil", *Polytechnic Journal*, 6.2 (2016): 171-209.

Ali, Taha Hussein; Mahmood M. Al-Abady. "Bayes's Analysis for Poisson Processes with Practical Application in Al-Salam Hospital/ Mosul", *Journal of Tanmiyat Al-Rafidain*, 31.94 (2009): 319-334.

Ali, Taha Hussein; Saleh, Dlshad Mahmood; Rahim, Alan Ghafur. "Comparison between the median and average charts using applied data representing pressing power of ceramic tiles and power of pipe concrete", *Journal of Humanity Sciences* 21.3 (2017): 141-149.

Ali, Taha Hussein; Shaymaa Mohammed Shakir. "Using Bayesian Weighted Method to Estimate the Parameters of Qualitative Regression Depending on Poisson distribution "A comparative Study", *ZANCO Journal of Pure and Applied Sciences*, 28.5 (2016): 41-52.

Ali, Taha Hussein; Tara Ahmed Hassan. "A comparison of methods for estimating regression parameters when there is a heterogeneity problem of variance with practical application", *Journal of Economics and Administrative Sciences*, 16.60 (2010): 216-227.

Ali, Taha Hussein; Tara Ahmed Hassan. "Estimating of Logistic Model by using Sequential Bayes Weights", *Journal of Economics and Administrative Sciences*, 13.46 (2007): 217-235.

Ali, Taha Hussien, (2017), "Using Proposed Nonparametric Regression Models for Clustered Data (A simulation study)." *Journal of Humanity Sciences*, 29.2: 78-87.

Ali, Taha Hussien, Nazeera Sedeek Kareem, and Awaz shahab mohammad, (2021), Data de-noise for Discriminant Analysis by using Multivariate Wavelets (Simulation with practical application), *Journal of Arab Statisticians Union (JASU)*, 5.3: 78-87

Kareem, Nazeera Sedeek, Taha Hussein Ali, and Awaz shahab M, "De-noise data by using Multivariate Wavelets in the Path analysis with

application", *Kirkuk University Journal of Administrative and Economic Sciences*, 10.1 (2020): 268-294.

Mustafa, Qais, and Ali, Taha Hussein. "Comparing the Box Jenkins models before and after the wavelet filtering in terms of reducing the orders with application." *Journal of Concrete and Applicable Mathematics* 11 (2013): 190-198.

Omar, Cheman, Taha Hussien Ali, and Kameran Hassn, Using Bayes weights to remedy the heterogeneity problem of random error variance in linear models, *IRAQI JOURNAL OF STATISTICAL SCIENCES*, 17, 2, 2020, 58-67.

Qais Mustafa Abd alqader and Taha Hussien Ali, (2020), Monthly Forecasting of Water Consumption in Erbil City Using a Proposed Method, *Al-Atroha journal*, 5.3:47-67.

Raza, Mahdi Saber, Taha Hussein Ali, and Tara Ahmed Hassan. "Using Mixed Distribution for Gamma and Exponential to Estimate of Survival Function (Brain Stroke)." *Polytechnic Journal* 8.1 (2018).

Shahla Hani Ali, Heyam A.A.Hayawi, Nazeera Sedeek K., and Taha Hussein Ali, (2023) "Predicting the Consumer price index and inflation average for the Kurdistan Region of Iraq using a dynamic model of neural networks with time series", *The 7th International Conference of Union if Arab Statistician-Cairo, Egypt* 8-9/3/2023:137-147.