Visual Numerics

# IMSL®

C Numerical Library

## User Guide

VOLUME 2 of 2 : C Stat Library

VERSION 7.0

Document ID: DC01033-01-1510-01

IMSL® C Numerical Stat Library

The IMSL C Stat Library is a library of C functions useful in scientific programming. Each function is designed and documented to be used in research activities as well as by technical specialists.



C, C#, Java™, Java™, and Fortran
Application Development Tools

# Contents

# Introduction

## Table of Contents



## IMSL C Stat Library

The IMSL C Stat Library is a library of C functions useful in scientific programming. Each function is designed and documented to be used in research activities as well as by technical specialists.

## Organization of the Documentation

This manual contains a concise description of each function. All information pertaining to a particular function is in one place within a chapter.

Each chapter begins with an introduction followed by a table of contents listing the functions included in the chapter. Documentation of the functions consists of the following information:

- Section Name: **Usually, the common root for the type *float* and type *double* versions of the function.**

- Purpose: **A statement of the purpose of the function.**

- Synopsis: **The form for referencing the subprogram with required arguments listed.**

  **Required Arguments:** A description of the required arguments in the order of their occurrence.

  **Input**: Argument must be initialized; it is not changed by the function.

**Input/Output:** Argument must be initialized; the function returns output through this argument. The argument cannot be a constant or an expression.

**Output:** No initialization is necessary. The argument cannot be a constant or an expression; the function returns output through this argument.

- **Return Value**: The value returned by the function.

- **Synopsis with Optional Arguments**: The form for referencing the function with both required and optional arguments listed.

- **Optional Arguments**: A description of the optional arguments in the order of their occurrence.

- **Description**: A description of the algorithm and references to detailed information. In many cases, other IMSL functions with similar or complementary functions are noted.

- **Errors**: Listing of any errors that may occur with a particular function. A discussion on error types is given in the "User Errors" section of the Reference Material. The errors are listed by their type as follows:

**Informational Errors:** List of informational errors that may occur with the function.

**Alert Errors:** List of alert errors that may occur with the function.

**Warning Errors:** List of warning errors that may occur with the function.

**Fatal Errors:** List of fatal errors that may occur with the function.

**References:** References are listed alphabetically by author.

## Finding the Right Function

The C Stat Library documentation is organized into chapters; each chapter contains functions with similar computational or analytical capabilities. To locate the right function for a given problem, use the table of contents located in each chapter introduction.

## Naming Conventions

Most functions are available in both a type *float* and a type *double* version, with names of the two versions sharing a common root. Some functions are also available in type *int*. The following list is of each type and the corresponding prefix of the function name in which multiple type versions exist:

| Type | Prefix |
|---|---|
| *float* | `imsls_f_` |
| *double* | `imsls_d_` |
| *int* | `imsls_i_` |

The section names for the functions contain only the common root to make finding the functions easier.

Where appropriate, the same variable name is used consistently throughout the C Stat Library. For example, `anova_table` denotes the array containing the analysis of variance statistics and `y` denotes a vector of responses for a dependent variable.

When writing programs accessing the C Stat Library, choose C names that do not conflict with IMSL external names. The careful user can avoid any conflicts with IMSL names if, in choosing names, the following rule is observed:

- **Do not choose a name beginning with "`imsls_`" in any combination of uppercase or lowercase characters.**

## Error Handling, Underflow, and Overflow

The functions in the C Stat Library attempt to detect and report errors and invalid input. This error-handling capability provides automatic protection for the user without requiring the user to make any specific provisions for the treatment of error conditions. Errors are classified according to severity and are assigned a code number. By default, errors of moderate or higher severity result in messages being automatically printed by the function. Moreover, errors of highest severity cause program execution to stop. The severity level, as well as the general nature of the error, is designated by an "error type" with symbolic names `IMSLS_FATAL`, `IMSLS_WARNING`, etc. See the section "User Errors" in the Reference Material for further details.

In general, the C Stat Library codes are written so that computations are not affected by underflow, provided the system (hardware or software) replaces an underflow with the value 0. Normally, system error messages indicating underflow can be ignored.

IMSL codes also are written to avoid overflow. A program that produces system error messages indicating overflow should be examined for programming errors such as incorrect input data, mismatch of argument types, or improper dimensions.

In many cases, the documentation for a function points out common pitfalls that can lead to failure of the algorithm.

# Time Series and Forecasting

## Routines

### ARIMA Models



### Model Construction and Evaluation Utilities

## Usage Notes

The functions in this chapter assume the time series does not contain any missing observations. If missing values are present, they should be set to NaN, and the routine will return an appropriate error message. To enable fitting of the model, the missing values must be replaced by appropriate estimates.

### *Time Domain Methodology*

Once the data are transformed to stationarity, a tentative model in the time domain is often proposed and parameter estimation, diagnostic checking and forecasting are performed.

### *ARIMA Model (Autoregressive Integrated Moving Average)*

A small, yet comprehensive, class of stationary time-series models consists of the nonseasonal ARMA processes defined by

$$\phi(B)\,(W_t - \mu) = \theta(B)A_t, \qquad t \in Z$$

where $Z = \{..., -2, -1, 0, 1, 2, ...\}$ denotes the set of integers, $B$ is the backward shift operator defined by $B^k W_t = W_{t-k}$, $\mu$ is the mean of $W_t$, and the following equations are true:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p,\ p \geq 0$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q,\ q \geq 0$$

The model is of order ($p$, $q$) and is referred to as an ARMA ($p$, $q$) model.

An equivalent version of the ARMA ($p$, $q$) model is given by

$$\phi(B)\,W_t = \theta_0 + \theta(B)A_t, \qquad t \in Z$$

where $\theta 0$ is an overall constant defined by the following:

$$\theta_0 = \mu\left(1 - \sum_{i=1}^{p} \phi_i\right)$$

See Box and Jenkins (1976, pp. 92–93) for a discussion of the meaning and usefulness of the overall constant.

If the "raw" data, {*Z*t}, are homogeneous and nonstationary, then differencing using `imsls_f_difference` induces stationarity, and the model is called ARIMA (AutoRegressive Integrated Moving Average). Parameter estimation is performed on the stationary time series *W*t, = $\nabla$d*Z*t , where $\nabla$d = (1 – *B*)d is the backward difference operator with period 1 and order *d*, d > 0.

Typically, the method of moments includes argument `IMSLS_METHOD_OF_MOMENTS` in a call to function `imsls_f_arma` for preliminary parameter estimates. These estimates can be used as initial values into the least-squares procedure by including argument `IMSLS_LEAST_SQUARES` in a call to function `imsls_f_arma`. Other initial estimates provided by the user can be used. The least-squares procedure can be used to compute conditional or unconditional least-squares estimates of the parameters, depending on the choice of the backcasting length. The functions for preliminary parameter estimation, least-squares parameter estimation, and forecasting follow the approach of Box and Jenkins (1976, Programs 2–4, pp. 498–509).

## arma

Computes least-square estimates of parameters for an ARMA model.

### *Synopsis*

*float* `*imsls_f_arma` (*int* `n_observations`, *float* `z[]`, *int* `p`, *int* `q`, ..., 0)

The type *double* function is `imsls_d_arma`.

### *Required Arguments*

*int* `n_`*observations* (Input)
Number of observations.

*float* `z[]` (Input)
Array of length `n_observations` containing the observations.

*int* `p` (Input)
Number of autoregressive parameters.

*int* `q` (Input)
Number of moving average parameters.

### *Return Value*

Pointer to an array of length 1 + *p* + *q* with the estimated constant, AR, and MA parameters. If `IMSLS_NO_CONSTANT` is specified, the 0-th element of this array is 0.0.

### *Description*

Function `imsls_f_arma` computes estimates of parameters for a nonseasonal ARMA model given a sample of observations, {*W*t}, for *t* = 1, 2, ..., *n*, where *n* = `n_observations`. There are two methods, method of moments and least squares, from which to choose. The default is method of moments.

Two methods of parameter estimation, method of moments and least squares, are provided. The user can choose the method of moments algorithm with the optional argument `IMSLS_METHOD_OF_MOMENTS`. The least-squares algorithm is used if the user specifies `IMSLS_LEAST_SQUARES`. If the user wishes to use the least-squares algorithm, the preliminary estimates are the method of moments estimates by default. Otherwise, the user can input initial estimates by specifying optional argument `IMSLS_INITIAL_ESTIMATES`. The following table lists the appropriate optional arguments for both the method of moments and least-squares algorithm:

| Method of Moments only | Least Squares only | Both Method of Moments and Least Squares |
|---|---|---|
| IMSLS_METHOD_OF_MOMENTS | IMSLS_LEAST_SQUARES | IMSLS_RELATIVE_ERROR |
| | IMSLS_CONSTANT (or IMSLS_NO_CONSTANT) | IMSLS_MAX_ITERATIONS |
| | IMSLS_AR_LAGS | IMSLS_MEAN_ESTIMATE |
| | IMSLS_MA_LAGS | IMSLS_AUTOCOV(_USER) |
| | IMSLS_BACKCASTING | IMSLS_RETURN_USER |
| | IMSLS_CONVERGENCE_TOLERANCE | IMSLS_ARMA_INFO |
| | IMSLS_INITIAL_ESTIMATES | |
| | IMSLS_RESIDUAL (_USER) | |
| | IMSLS_PARAM_EST_COV (_USER) | |
| | IMSLS_SS_RESIDUAL | |

### *Method of Moments Estimation*

Suppose the time series $\{Z_t\}$ is generated by an ARMA ($p$, $q$) model of the form

$$\phi(B)Z_t = \theta_0 + \theta(B)A_t$$

for $t \in \{0, \pm1, \pm2, ...\}$

Let $\hat{\mu}$ = `z_mean` be the estimate of the mean μ of the time series$\{Z_t\}$, where $\hat{\mu}$ equals the following:

$$\hat{\mu} = \begin{cases} \mu & \text{for } \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} Z_t & \text{for } \mu \text{ unknown} \end{cases}$$

The autocovariance function is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(Z_t - \hat{\mu})(Z_{t+k} - \hat{\mu})$$

for $k$ = 0, 1, ..., $K$, where $K = p + q$. Note that $\hat{\sigma}$ (0) is an estimate of the sample variance.

Given the sample autocovariances, the function computes the method of moments estimates of the autoregressive parameters using the extended Yule-Walker equations as follows:

$$\hat{\Sigma}\hat{\phi} = \hat{\sigma}$$

where

$$\begin{aligned} &\hat{\phi} = \left(\hat{\phi}_1, \ldots, \hat{\phi}_p\right)^T \\ &\hat{\Sigma}_{ij} = \hat{\sigma}\left(|\, q + i - j\,|\right),\ i, j = 1, \ldots, p \\ &\hat{\sigma}_i = \hat{\sigma}\left(q + i\right), \qquad i = 1, \ldots, p \end{aligned}$$

The overall constant $\theta_0$ is estimated by the following:

$$\hat{\theta}_0 = \begin{cases} \hat{\mu} & \text{for } p = 0 \\ \hat{\mu}\left(1 - \sum_{i=1}^{p}\hat{\phi}_i\right) & \text{for } p > 0 \end{cases}$$

The moving average parameters are estimated based on a system of nonlinear equations given $K = p + q + 1$ autocovariances, $\sigma(k)$ for $k$ = 1, ..., $K$, and $p$ autoregressive parameters $\phi_i$ for $i$ = 1, ..., $p$.

Let $Z'_t = \phi(B)Z_t$. The autocovariances of the derived moving average process $Z'_t = \theta(B)A_t$ are estimated by the following relation:

$$\hat{\sigma}'(k) = \begin{cases} \hat{\sigma}(k) & \text{for } p = 0 \\ \sum_{i=0}^{p}\sum_{j=0}^{p}\hat{\phi}_i\hat{\phi}_j\left(\hat{\sigma}\left(|k + i - j|\right)\right) & \text{for } p \geq 1, \hat{\phi}_0 \equiv -1 \end{cases}$$

The iterative procedure for determining the moving average parameters is based on the relation

$$\sigma(k) = \begin{cases} \left(1+\theta_1^2+\ldots+\theta_q^2\right)\sigma_A^2 & \text{for } k=0 \\ \left(-\theta_k+\theta_1\theta_{k+1}+\ldots+\theta_{q-k}\theta_q\right)\sigma_A^2 & \text{for } k\geq 1 \end{cases}$$

where $\sigma(k)$ denotes the autocovariance function of the original $Z$t process.

Let $\tau = (\tau 0, \tau 1, \ldots, \tau q)\mathrm{T}$ and $f = (f0, f1, \ldots, fq)\mathrm{T}$, where

$$\tau_j = \begin{cases} \sigma_A & \text{for } j=0 \\ -\theta_j/\tau_0 & \text{for } j=1,\ldots,q \end{cases}$$

and

$$f_j = \sum_{i=0}^{q-j} \tau_i \tau_{i+j} - \hat{\sigma}'(j) \qquad \text{for } j=0,1,\ldots,q$$

Then, the value of $\tau$ at the ($i$ + 1)-th iteration is determined by the following:

$$\tau^{i+1} = \tau^i - \left(T^i\right)^{-1} f^i$$

The estimation procedure begins with the initial value

$$\tau^0 = (\sqrt{\hat{\sigma}'(0)}, \quad 0,\ldots,0)^T$$

and terminates at iteration $i$ when either $||f\,\mathrm{i}||$ is less than `relative_error` or
$i$ equals `max_iterations`. The moving average parameter estimates are obtained from the final estimate of $\tau$ by setting

$$\hat{\theta}_j = -\tau_j/\tau_0 \text{ for } j=1,\ldots,q$$

The random shock variance is estimated by the following:

$$\hat{\sigma}_A^2 = \begin{cases} \hat{\sigma}(0) - \sum_{i=1}^{p} \hat{\phi}_i \hat{\sigma}(i) & \text{for } q = 0 \\ \tau_0^2 & \text{for } q \geq 0 \end{cases}$$

See Box and Jenkins (1976, pp. 498–500) for a description of a function that performs similar computations.

### *Least-squares Estimation*

Suppose the time series {*Z*t} is generated by a nonseasonal ARMA model of the form,

$\phi(B)$ ($Z$t – $\mu$) = $\theta(B)A$t          for t $\in$ {0, ±1, ±2, …}

where *B* is the backward shift operator, $\mu$ is the mean of *Z*t, and

$$\phi(B) = 1 - \phi_1 B^{l_\phi(1)} - \phi_2 B^{l_\phi(2)} - \ldots - \phi_p B^{l_\phi(p)} \quad \text{for } p \geq 0$$
$$\theta(B) = 1 - \theta_1 B^{l_\theta(1)} - \theta_2 B^{l_\theta(2)} - \ldots - \theta_q B^{l_\theta(q)} \quad \text{for } q \geq 0$$

with *p* autoregressive and *q* moving average parameters. Without loss of generality, the following is assumed:

1 ≤ *l*f (1) ≤ *l*f (2) ≤ … ≤ *l*f (*p*)

1 ≤ *l*q (1) ≤ *l*q (2) ≤ … ≤ *l*q (*q*)

so that the nonseasonal ARMA model is of order (*p*′, *q*′), where *p*′ = *l*q (*p*) and *q*′ = *l*q (*q*). Note that the usual hierarchical model assumes the following:

*l*f (*i*) = *i*, 1 ≤ *i* ≤ *p*

*l*q (*j*) = *j*, 1 ≤ *j* ≤ *q*

Consider the sum-of-squares function

$$S_T(\mu,\phi,\theta) = \sum_{-T+1}^{n} [A_t]^2$$

where

$$[A_t] = E\left[A_t \middle| (\mu,\phi,\theta,Z)\right]$$

and $T$ is the backward origin. The random shocks {$A$t} are assumed to be independent and identically distributed

$$N\left(0,\sigma_A^2\right)$$

random variables. Hence, the log-likelihood function is given by

$$l(\mu,\phi,\theta,\sigma_A) = f(\mu,\phi,\theta) - n\ln(\sigma_A) - \frac{S_T(\mu,\phi,\theta)}{2\sigma_A^2}$$

where $f(\mu, \phi, \theta)$ is a function of $\mu$, $\phi$, and $\theta$.

For $T = 0$, the log-likelihood function is conditional on the past values of both $Z$t and $A$t required to initialize the model. The method of selecting these initial values usually introduces transient bias into the model (Box and Jenkins 1976, pp. 210–211). For $T = \infty$, this dependency vanishes, and estimation problem concerns maximization of the unconditional log-likelihood function. Box and Jenkins (1976, p. 213) argue that

$$S_\infty(\mu,\phi,\theta)/\left(2\sigma_A^2\right)$$

dominates

$$l\left(\mu,\phi,\theta,\sigma_A^2\right)$$

The parameter estimates that minimize the sum-of-squares function are called least-squares estimates. For large *n*, the unconditional least-squares estimates are approximately equal to the maximum likelihood-estimates.

In practice, a finite value of *T* will enable sufficient approximation of the unconditional sum-of-squares function. The values of [*A*T] needed to compute the unconditional sum of squares are computed iteratively with initial values of *Z*t obtained by back forecasting. The residuals (including backcasts), estimate of random shock variance, and covariance matrix of the final parameter estimates also are computed. ARIMA parameters can be computed by using `imsls_f_difference` with `imsls_f_arma`.

### *Warning Errors*

| | |
|---|---|
| `IMSLS_LEAST_SQUARES_FAILED` | Least-squares estimation of the parameters has failed to converge. Increase "maxbc" and/or "tolerance" and/or "convergence_tolerance." The estimates of the parameters at the last iteration may be used as new starting values. |

### *Fatal Errors*

| | |
|---|---|
| `IMSLS_TOO_MANY_CALLS` | The number of calls to the function has exceeded "itmax"*("n"+1) = %(i1). The user may try a new initial guess. |
| `IMSLS_INCREASE_ERRREL` | The bound for the relative error, "errrel" = %(r1), is too small. No further improvement in the approximate solution is possible. The user should increase "errrel". |
| `IMSLS_NEW_INITIAL_GUESS` | The iteration has not made good progress. The user may try a new initial guess. |

## max_arma

Exact maximum likelihood estimation of the parameters in a univariate ARMA (autoregressive, moving average) time series model.

### *Synopsis*

*float* `*imsls_f_max_arma` (*int* `n_obs`, *float* `w[]`, *int* `p`, *int* `q`,…,0)

The type *double* function is `imsls_d_max_arma`.

### Required Arguments

*int* `n_obs` (Input)
Number of observations in the time series.

*float* `w[]` (Input)
Array of length `n_obs` containing the time series.

*int* `p` (Input)
Non-negative number of autoregressive parameters.

*int* `q` (Input)
Non-negative number of moving average parameters.

### Return Value

Pointer to an array of length 1+`p`+`q` with the estimated constant, AR and MA parameters. If no value can be computed, `NULL` is returned.

### Synopsis with Optional Arguments

```
float  *imsls_f_max_arma (int n_obs, float w[], int p, int q,
         IMSLS_INITIAL_ESTIMATES, float init_ar[] float init_ma[],
         IMSLS_PRINT_LEVEL, int iprint,
         IMSLS_MAX_ITERATIONS, int maxit,
         IMSLS_LOG_LIKELIHOOD, float *log_likeli,
         IMSLS_VAR_NOISE, float *avar,
         IMSLS_ARMA_INFO, Imsls_f_arma  **arma_info,
         IMSLS_MEAN_ESTIMATE, float *w_mean,
         IMSLS_RETURN_USER, float *constant, float ar[], float ma[],
         0)
```

### Optional Arguments

`IMSLS_INITIAL_ESTIMATES`, *float* `init_ar[]`, *float* `init_ma[]` (Input)
If specified, `init_ar` is an array of length `p` containing preliminary estimates of the autoregressive parameters, and `init_ma` is an array of length `q` containing preliminary

estimates of the moving average parameters; otherwise, they are computed internally. If `p`=0 or `q`=0, then the corresponding arguments are ignored.

`IMSLS_PRINT_LEVEL`, *int* `iprint` (Input)

Printing option:

0 — No printing.

1 — Prints final results only.

2 — Prints intermediate and final results.

Default: `iprint` = 0

`IMSLS_MAX_ITERATIONS`, *int* `maxit` (Input)

Maximum number of estimation iterations.

Default: `maxit` = 300

`IMSLS_VAR_NOISE`, *float* `*avar` (Output)

Estimate of innovation variance.

`IMSLS_LOG_LIKELIHOOD`, *float* `*log_likeli` (Output)

Value of -2*(ln(likelihood)) for the fitted model.

`IMSLS_ARMA_INFO`, *Imsls_f_arma* `**arma_info` (Output)

Address of a pointer to an internally allocated structure of type *Imsls_f_arma* that contains information necessary in the call to `imsls_f_arma_forecast`.

`IMSLS_MEAN_ESTIMATE`, *float* `*w_mean` (Input/Output)

Estimate of the mean of the time series `w`. On return, `w_mean` contains an update of the mean.

Default: Time series `w` is centered about its sample mean.

`IMSLS_RETURN_USER`, *float* `*constant`, *float* `ar[]`, *float* `ma[]` (Output)

If specified, `constant` is the constant parameter estimate, `ar` is an array of length `p` containing the final autoregressive parameter estimates, and `ma` is an array of length `q` containing the final moving average parameter estimates.

### *Description*

The function `imsls_f_max_arma` is derived from the maximum likelihood estimation algorithm described by Akaike, Kitagawa, Arahata and Tada (1979), and the XSARMA routine published in the TIMSAC-78 Library.

Using the notation developed in the Time Domain Methodology at the beginning of this chapter, the stationary time series $W_t$ with mean $\mu$ can be represented by the nonseasonal autoregressive moving average (ARMA) model by the following relationship:

$$\phi(B)(W_t - \mu) = \theta(B)a_t$$

where

$$t \in ZZ = \{\cdots, -2, -1, 0, 1, 2, \cdots\},$$

*B* is the backward shift operator defined by $B^k W_t = W_{t-k}$,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p, \quad p \ge 0,$$

and

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q, \quad q \ge 0.$$

Function `imsls_f_max_arma` estimates the AR coefficients $\phi_1, \phi_2, \cdots, \phi_p$ and the MA coefficients $\theta_1, \theta_2, \cdots, \theta_q$ using maximum likelihood estimation.

Function `imsls_f_max_arma` checks the initial estimates for both the autoregressive and moving average coefficients to ensure that they represent a stationary and invertible series respectively.

If

$$\phi_1, \phi_2, \cdots, \phi_p$$

are the initial estimates for a stationary series then all (complex) roots of the following polynomial will fall outside the unit circle:

$$1-\phi_1 z-\phi_2 z^2-\cdots-\phi_p z^p .$$

If

$$\theta_1,\theta_2,\cdots,\theta_q$$

are initial estimates for an invertible series then all (complex) roots of the polynomial

$$1-\theta_1 z-\theta_2 z^2-\cdots-\theta_q z^q$$

will fall outside the unit circle.

Initial values for the AR and MA coefficients can be supplied by vectors `init_ar` and `init_ma`. Otherwise, estimates are computed internally by the method of moments. `imsls_f_max_arma` computes the roots of the associated polynomials. If the AR estimates represent a non-stationary series, `imsls_f_max_arma` issues a warning message and replaces `init_ar` with initial values that are stationary. If the MA estimates represent a non-invertible series, `imsls_f_max_arma` issues a terminal error, and new initial values have to be sought.

`imsls_f_max_arma` also validates the final estimates of the AR coefficients to ensure that they too represent a stationary series. This is done to guard against the possibility that the internal log-likelihood optimizer converged to a non-stationary solution. If non-stationary estimates are encountered, `imsls_f_max_arma` issues a fatal error message.

For model selection, the ARMA model with the minimum value for *AIC* might be preferred,

$$AIC=\texttt{log\_likeli}+2(\mathrm{p}+\mathrm{q})$$

Function `imsls_f_max_arma` can also handle white noise processes, i.e. ARMA(0,0) Processes.

## auto_uni_ar

Automatic selection and fitting of a univariate autoregressive time series model. The lag for the model is automatically selected using Akaike's information criterion (AIC). Estimates of the autoregressive parameters for the model with minimum AIC are calculated using method of moments, method of least squares, or maximum likelihood.

### *Synopsis*

*float* `*imsls_f_auto_uni_ar` (*int* `n_obs`, *float* `z[]`, *int* `maxlag`,
*int* `*p`,…,0)

The type *double* function is `imsls_d_auto_uni_ar`.

### *Required Arguments*

*int* `n_obs` (Input)
Number of observations in the time series.

*float* `z[]` (Input)
Array of length `n_obs` containing the stationary time series.

*int* `maxlag` (Input)
Maximum number of autoregressive parameters requested. It is required that
1≤ `maxlag` ≤ `n_obs`/2.

*int* `*p` (Output)
Number of autoregressive parameters in the model with minimum AIC.

### *Return Value*

Vector of length 1+ `maxlag` containing the estimates for the constant and the autoregressive parameters in the model with minimum AIC. The estimates are located in the first 1+ `p` locations of this array.

### *Synopsis with Optional Arguments*

*float* `*imsls_f_auto_uni_ar` (*int* `n_obs`, *float* `z[]`, *int* `maxlag`, *int* `*p`,
`IMSLS_PRINT_LEVEL`, *int* `iprint`,
`IMSLS_MAX_ITERATIONS`, *int* `maxit`,
`IMSLS_METHOD`, *int* `method`,

```
IMSLS_VAR_NOISE, float *avar,
IMSLS_AIC, float *aic,
IMSLS_MEAN_ESTIMATE, float *z_mean,
IMSLS_RETURN_USER, float *constant, float ar[],
0)
```

## *Optional Arguments*

`IMSLS_PRINT_LEVEL`, *int* `iprint` (Input)

Printing option:

0 — No printing.

1 — Prints final results only.

2 — Prints intermediate and final results.

Default: `iprint` = 0

`IMSLS_MAX_ITERATIONS`, *int* `maxit` (Input)

Maximum number of estimation iterations.

Default: `maxit` = 300

`IMSLS_METHOD`, *int* `method` (Input)

Estimation method option:

0 — Method of moments

1 — Method of least squares realized through Householder transformations

2 — Maximum likelihood

Default: `method` = 1

`IMSLS_VAR_NOISE`, *float* `*avar` (Output)

Estimate of innovation variance.

`IMSLS_AIC`, *float* `*aic` (Output)

Minimum AIC.

`IMSLS_MEAN_ESTIMATE`, *float* `*z_mean` (Input/Output)

Estimate of the mean of the time series `z`. On return, `z_mean` contains an update of the mean.

Default: Time series `z` is centered about its sample mean.

IMSLS_RETURN_USER*, float* *constant*, float* ar[] (Output)

If specified, constant is the constant parameter estimate, ar is an array of length maxlag containing the final autoregressive parameter estimates in its first p locations.

### *Description*

Function auto_uni_ar automatically selects the order of the AR model that best fits the data and then computes the AR coefficients. The algorithm used in auto_uni_ar is derived from the work of Akaike, H., et. al (1979) and Kitagawa and Akaike (1978). This code was adapted from the UNIMAR procedure published as part of the TIMSAC-78 Library.

The best fit AR model is determined by successively fitting AR models with 0, 1, 2, ..., maxlag autoregressive coefficients. For each model, Akaike's Information Criterion (AIC) is calculated based on the formula

$$AIC = -2\ln(likelihood) + 2(\mathrm{p}+1)$$

Function auto_uni_ar uses the approximation to this formula developed by Ozaki and Oda (1979),

$$AIC = (\mathtt{n\_obs} - \mathtt{maxlag})\ln(\hat{\sigma}^2) + 2(\mathrm{p}+1) + (\mathtt{n\_obs} - \mathtt{maxlag})(\ln(2\pi)+1),$$

where $\hat{\sigma}^2$ is an estimate of the residual variance of the series, commonly known in time series analysis as the innovation variance. By dropping the constant

$$(\mathtt{n\_obs} - \mathtt{maxlag})(\ln(2\pi)+1),$$

the calculation is simplified to

$$AIC = (\mathtt{n\_obs} - \mathtt{maxlag})\ln(\hat{\sigma}^2) + 2(\mathrm{p}+1)$$

The best fit model is the model with minimum AIC. If the number of parameters in this model is equal to the highest order autoregressive model fitted, i.e., p=maxlag, then a model with smaller AIC might exist for larger values of maxlag. In this case, increasing maxlag to explore AR models with additional autoregressive parameters might be warranted.

If method = 0, estimates of the autoregressive coefficients for the model with minimum AIC are calculated using method of moments. If method =1, the coefficients are determined by the method of least squares applied in the form described by Kitagawa and Akaike (1978). Otherwise, if method =2, the coefficients are estimated using maximum likelihood.

# ts_outlier_identification

Detects and determines outliers and simultaneously estimates the model parameters in a time series whose underlying outlier free series follows a general seasonal or nonseasonal ARMA model.

## *Synopsis*

*float* *imsls_f_ts_outlier_identification (*int* n_obs, *int* model[],
  *float* w[],…,0)

The type *double* function is imsls_d_ts_outlier_identification.

## *Required Arguments*

*int* n_obs (Input)
Number of observations in the time series.

*int* model[] (Input)
Vector of length 4 containing the numbers *p*, *q*, *s*, *d* of the ARIMA $(p,0,q)\times(0,d,0)_s$ model the outlier free series is following.

*float* w[] (Input)
An array of length n_obs containing the time series.

## *Return Value*

Pointer to an array of length n_obs containing the outlier free time series.
If an error occurred, NULL is returned.

## *Synopsis with Optional Arguments*

*float* *imsls_f_ts_outlier_identification (*int* n_obs,
  *int* model[], *float* w[],

```
IMSLS_RETURN_USER, float x[],
IMSLS_DELTA, float delta,
IMSLS_CRITICAL, float critical,
IMSLS_EPSILON, float epsilon,
IMSLS_RELATIVE_ERROR, float relative_error,
IMSLS_RESIDUAL, float **residual,
IMSLS_RESIDUAL_USER, float residual[],
IMSLS_RESIDUAL_SIGMA, float *res_sigma,
IMSLS_NUM_OUTLIERS, int *num_outliers,
IMSLS_OUTLIER_STATISTICS, int **outlier_stat,
IMSLS_OUTLIER_STATISTICS_USER, int outlier_stat[],
IMSLS_TAU_STATISTICS, float **tau_stat,
IMSLS_TAU_STATISTICS_USER, float tau_stat[],
IMSLS_OMEGA_WEIGHTS, float **omega,
IMSLS_OMEGA_WEIGHTS_USER, float omega[],
IMSLS_ARMA_PARAM, float **parameters,
IMSLS_ARMA_PARAM_USER, float parameters[],
IMSLS_AIC, float *aic,
0)
```

### *Optional Arguments*

`IMSLS_RETURN_USER`, *float* `x[]` (Output)

A user supplied array of length `n_obs` containing the outlier free series.

`IMSLS_DELTA`, *float* `delta` (Input)

The dampening effect parameter used in the detection of a Temporary Change Outlier (TC), 0<`delta` < 1.

Default: `delta` = 0.7

`IMSLS_CRITICAL`, *float* `critical` (Input)

Critical value used as a threshold for outlier detection, `critical` > 0.

Default: `critical` = 3.0

`IMSLS_EPSILON`, *float* `epsilon` (Input)

Positive tolerance value controlling the accuracy of parameter estimates during outlier detection.

Default: `epsilon` = 0.001

`IMSLS_RELATIVE_ERROR`, *float* `relative_error` (Input)

Stopping criterion for the nonlinear equation solver used in function `imsls_f_arma`.
Default: `relative_error` = $10^{-10}$.

`IMSLS_RESIDUAL`, *float* `**residual` (Output)

Address of a pointer to an internally allocated array of length `n_obs` containing the residuals for the outlier free series.

`IMSLS_RESIDUAL_USER`, *float* `residual[]` (Output)

Storage for array `residual` is provided by the user. See `IMSLS_RESIDUAL`.

`IMSLS_RESIDUAL_SIGMA`, *float* `*res_sigma` (Output)

Residual standard error of the outlier free series.

`IMSLS_NUM_OUTLIERS`, *int* `*num_outliers` (Output)

The number of outliers detected.

`IMSLS_OUTLIER_STATISTICS`, *int* `**outlier_stat` (Output)

Address of a pointer to an internally allocated array of length `num_outliers` × 2 containing outlier statistics. The first column contains the time at which the outlier was observed (`t`=1,2,...,`n_obs`) and the second column contains an identifier indicating the type of outlier observed.
Outlier types fall into one of five categories:

0 Innovational Outliers (IO)

1 Additive outliers (AO)

2 Level Shift Outliers (LS)

3 Temporary Change Outliers (TC)

4 Unable to Identify (UI).

Use `IMSLS_NUM_OUTLIERS` to obtain `IMSLS_NUM_OUTLIERS`, the number of detected outliers.
If `num_outliers = 0`, `NULL` is returned.

`IMSLS_OUTLIER_STATISTICS_USER`, *int* `outlier_stat[]` (Output)

A user allocated array of length `n_obs` × 2 containing outlier statistics in the first `num_outliers` locations. See `IMSLS_OUTLIER_STATISTICS`.

If `num_outliers = 0`, `outlier_stat` stays unchanged.

`IMSLS_TAU_STATISTICS`, *float* `**tau_stat` (Output)

Address of a pointer to an internally allocated array of length `num_outliers` containing the *t* value for each detected outlier.

If `num_outliers =` 0, `NULL` is returned.

`IMSLS_TAU_STATISTICS_USER`, *float* `tau_stat[]` (Output)

A user allocated array of length `n_obs` containing the *t* value for each detected outlier in its first `num_outliers` locations.

If `num_outliers = 0`, `tau_stat` stays unchanged.

`IMSLS_OMEGA_WEIGHTS`, *float* `**omega` (Output)

Address of a pointer to an internally allocated array of length `num_outliers` containing the computed $\omega$ weights for the detected outliers.

If `num_outliers = 0`, `NULL` is returned.

`IMSLS_OMEGA_WEIGHTS_USER` *float* `omega[]` (Output)

A user allocated array of length `n_obs` containing the computed $\omega$ weights for the detected outliers in its first `num_outliers` locations.

If `num_outliers = 0`, `omega` stays unchanged.

`IMSLS_ARMA_PARAM`, *float* `**parameters` (Output)

Address of a pointer to an internally allocated array of length 1+*p*+*q* containing the estimated constant, AR and MA parameters.

`IMSLS_ARMA_PARAM_USER` *float* `parameters[]` (Output)

A user allocated array of length 1+*p*+*q* containing the estimated constant, AR and MA parameters.

`IMSLS_AIC`, *float* `*aic` (Output)

Akaike's information criterion (AIC).

## *Description*

Consider a univariate time series $\{Y_t\}$ that can be described by the following multiplicative seasonal ARIMA model of order $(p,0,q)\times(0,d,0)_s$:

$$Y_t - \mu = \frac{\theta(B)}{\Delta_s^d \phi(B)} a_t,\ t = 1,\ldots,n.$$

Here, $\Delta_s^d = (1-B^s)^d$, $\theta(B) = 1-\theta_1 B - \ldots - \theta_q B^q$, $\phi(B) = 1-\phi_1 B - \ldots - \phi_p B^p$. $B$ is the lag operator, $B^k Y_t = Y_{t-k}$, $\{a_t\}$ is a white noise process, and $\mu$ denotes the mean of the series $\{Y_t\}$.

In general, $\{Y_t\}$ is not directly observable due to the influence of outliers. Chen and Liu (1993) distinguish between four types of outliers: innovational outliers (IO), additive outliers (AO), temporary changes (TC) and level shifts (LS). If an outlier occurs as the last observation of the series, then Chen and Liu's algorithm is unable to determine the outlier's classification. In `imsls_f_ts_outlier_identification`, such an outlier is called a UI (unable to identify) and is treated as an innovational outlier.

In order to take the effects of multiple outliers occurring at time points $t_1, t_2, \ldots, t_m$ into account, Chen and Liu consider the following model:

$$Y_t^* - \mu = \sum_{j=1}^m \omega_j L_j(B) I_t(t_j) + \frac{\theta(B)}{\Delta_s^d \phi(B)} a_t.$$

Here, $\{Y_t^*\}$ is the observed outlier contaminated series, and $\omega_j$ and $L_j(B)$ denote the magnitude and dynamic pattern of outlier $j$, respectively. $I_t(t_j)$ is an indicator function that determines the temporal course of the outlier effect, $I_{t_j}(t_j) = 1$, $I_t(t_j) = 0$ otherwise. **Note** that $L_j(B)$ operates on $I_t$ via $B^k I_t = I_{t-k}, k = 0,1,\ldots$.

The last formula shows that the outlier free series $\{Y_t\}$ can be obtained from the original series $\{Y_t^*\}$ by removing all occurring outlier effects:

$$Y_t = Y_t^* - \sum_{j=1}^{m} \omega_j L_j(B) I_t(t_j) \, .$$

The different types of outliers are charaterized by different values for $L_j(B)$:

1. $L_j(B) = \dfrac{\theta(B)}{\Delta_s^d \phi(B)}$ for an innovational outlier,
2. $L_j(B) = 1$ for an additive outlier,
3. $L_j(B) = (1-B)^{-1}$ for a level shift outlier *and*
4. $L_j(B) = (1-\delta B)^{-1}, 0 < \delta < 1,$ for a temporary change outlier.

Function `imsls_f_ts_outlier_identification` is an implementation of Chen and Liu's algorithm. It determines the coefficients in $\phi(B), \theta(B)$ and the outlier effects in the model for the observed series jointly in three stages. The magnitude of the outlier effects is determined by least squares estimates. Outlier detection itself is realized by examination of the maximum value of the standardized statistics of the outlier effects. For a detailed description, see Chen and Liu's original paper (1993).

Intermediate and final estimates for the coefficients in $\phi(B)$ and $\theta(B)$ are computed by functions `imsls_f_arma` and `imsls_f_max_arma`. If the roots of $\phi(B)$ or $\theta(B)$ lie on or within the unit circle, then the algorithm stops with an appropriate error message. In this case, different values for *p* and *q* should be tried.

## difference

Differences a seasonal or nonseasonal time series.

### *Synopsis*

*float* `*imsls_f_difference` (*int* `n_observations`, *float* `z[]`, *int* `n_differences`, *int* `periods[]`, ..., 0)

The type *double* function is imsls_d_difference.

### Required Arguments

*int* n_observations (Input)
Number of observations.

*float* z[] (Input)
Array of length n_observations containing the time series.

*int* n_differences (Input)
Number of differences to perform. Argument n_differences must be greater than or equal to 1.

*int* periods[] (Input)
Array of length n_differences containing the periods at which z is to be differenced.

### Return Value

Pointer to an array of length n_observations containing the differenced series.

### Synopsis with Optional Arguments

*float* *imsls_f_difference (*int* n_observations, *float* z[], *int* n_differences, *int* periods[],
IMSLS_ORDERS, *int* orders[],
IMSLS_LOST, *int*v*n_lost,
IMSLS_EXCLUDE_FIRST, *or*
IMSLS_SET_FIRST_TO_NAN,
IMSLS_RETURN_USER, *float* w[],
0)

### Optional Arguments

IMSLS_ORDERS, *int* orders[] (Input)
Array of length n_differences containing the order of each difference given in periods. The elements of orders must be greater than or equal to 0.

IMSLS_LOST, *int* *n_lost (Output)
Number of observations lost because of differencing the time series z.

`IMSLS_EXCLUDE_FIRST`, *or*

`IMSLS_SET_FIRST_TO_NAN`

If `IMSLS_EXCLUDE_FIRST` is specified, the first `n_lost` are excluded from `w` due to differencing. The differenced series `w` is of length `n_observations` − `n_lost`. If `IMSLS_SET_FIRST_TO_NAN` is specified, the first `n_lost` observations are set to NaN (Not a Number). This is the default if neither `IMSLS_EXCLUDE_FIRST` nor `IMSLS_SET_FIRST_TO_NAN` is specified.

`IMSLS_RETURN_USER`, *float* `w[]` (Output)

If specified, `w` contains the differenced series. If `IMSLS_EXCLUDE_FIRST` also is specified, `w` is of length `n_observations`. If `IMSLS_SET_FIRST_TO_NAN` is specified or neither `IMSLS_EXCLUDE_FIRST` nor `IMSLS_SET_FIRST_TO_NAN` is specified, `w` is of length `n_observations` − `n_lost`.

## Description

Function `imsls_f_difference` performs *m* = `n_differences` successive backward differences of period *s*i = `periods` [*i* – 1] and order di = `orders` [*i* – 1] for *i* = 1, ..., *m* on the *n* = `n_observations` observations {*Z*t} for *t* = 1, 2, ..., *n*.

Consider the backward shift operator *B* given by

$$B^k Z_t = Z_{t-k}$$

for all *k*. Then, the *backward difference operator* with period *s* is defined by the following:

$$\Delta_s Z_t = (1 - B^s) Z_t = Z_t - Z_{t-s} \quad \text{for} \quad s > 0 \, .$$

Note that $B^s Z_t$ and $\Delta^s Z_t$ are defined only for *t* = (*s* + 1), ..., *n*. Repeated differencing with period *s* is simply

$$\Delta_s^d Z_t = \left(1 - B^s\right)^d Z_t = \sum_{j=0}^{d} \frac{d\,!}{j\,!\left(d - j\right)!} \left(-1\right)^j B^{sj} Z_t$$

where *d* ≥ 0 is the order of differencing. Note that

$$\Delta_s^d Z_t$$

is defined only for $t = (sd + 1), \ldots, n$.

The general difference formula used in the function `imsls_f_difference` is given by

$$W_t = \begin{cases} \text{NaN} & \text{for } t = 1, \ldots, n_L \\ \Delta_{s_1}^{d_1} \Delta_{s_2}^{d_2} \ldots \Delta_{s_m}^{d_m} Z_t & \text{for } t = n_L + 1, \ldots, n \end{cases}$$

where $n$L represents the number of observations "lost" because of differencing and NaN represents the missing value code. Note that

$$n_L = \sum_j s_j d_j$$

A homogeneous, stationary time series can be arrived at by appropriately differencing a homogeneous, nonstationary time series (Box and Jenkins 1976, p. 85). Preliminary application of an appropriate transformation followed by differencing of a series can enable model identification and parameter estimation in the class of homogeneous stationary autoregressive moving average models.

### *Fatal Errors*

| | |
|---|---|
| `IMSLS_PERIODS_LT_ZERO` | "period[#]" = #. All elements of "period" must be greater than 0. |
| `IMSLS_ORDER_NEGATIVE` | "order[#]" = #. All elements of "order" must be nonnegative. |
| `IMSLS_Z_CONTAINS_NAN` | "z[#]" = NaN; "z" can not contain missing values. There may be other elements of "z" that are equal to NaN. |

## box_cox_transform

Performs a forward or an inverse Box-Cox (power) transformation.

### *Synopsis*

*float* `*imsls_f_box_cox_transform` (*int* `n_observations`, *float* `z[]`, *float* `power`, ..., 0)

The type *double* function is imsls_d_box_cox_transform.

## *Required Arguments*

*int* n_observations (Input)
Number of observations in z.

*float* z[] (Input)
Array of length n_observations containing the observations.

*float* power (Input)
Exponent parameter in the Box-Cox (power) transformation.

## *Return Value*

Pointer to an internally allocated array of length n_observations containing the transformed data. To release this space, use imsls_free. If no value can be computed, then NULL is returned.

## *Synopsis with Optional Arguments*

*float* *imsls_f_box_cox_transform (*int* n_observations, *float* z[], *float* power,
IMSLS_SHIFT, *float* shift,
IMSLS_INVERSE_TRANSFORM,
IMSLS_RETURN_USER, *float* x[],
0)

## *Optional Arguments*

IMSLS_SHIFT, *float* shift (Input)
Shift parameter in the Box-Cox (power) transformation. Parameter shift must satisfy the relation min ($z(i)$) + shift > 0.
Default: shift = 0.0.

IMSLS_INVERSE_TRANSFORM
If IMSLS_INVERSE_TRANSFORM is specified, the inverse transform is performed.

IMSLS_RETURN_USER, *float* x[] (Output)
User-allocated array of length n_observations containing the transformed data.

### *Description*

Function `imsls_f_box_cox_transform` performs a forward or an inverse Box-Cox (power) transformation of *n* = `n_observations` observations {*Z*t} for *t* = 1, 2, ..., *n*.

The forward transformation is useful in the analysis of linear models or models with nonnormal errors or nonconstant variance (Draper and Smith 1981, p. 222). In the time series setting, application of the appropriate transformation and subsequent differencing of a series can enable model identification and parameter estimation in the class of homogeneous stationary autoregressive-moving average models. The inverse transformation can later be applied to certain results of the analysis, such as forecasts and prediction limits of forecasts, in order to express the results in the scale of the original data. A brief note concerning the choice of transformations in the time series models is given in Box and Jenkins (1976, p. 328).

The class of power transformations discussed by Box and Cox (1964) is defined by

$$X_t = \begin{cases} \dfrac{\left(Z_t + \xi\right)^{\lambda} - 1}{\lambda} & \lambda \neq 0 \\ \ln\left(Z_t + \xi\right) & \lambda = 0 \end{cases}$$

where *Z*t + ξ > 0 for all *t*. Since

$$\lim_{\lambda \to 0} \frac{\left(Z_t + \xi\right)^{\lambda} - 1}{\lambda} = \ln\left(Z_t + \xi\right)$$

the family of power transformations is continuous.

Let λ = `power` and ξ = `shift`; then, the computational formula used by `imsls_f_box_cox_transform` is given by

$$X_t = \begin{cases} \left(Z_t + \xi\right)^{\lambda} & \lambda \neq 0 \\ \ln\left(Z_t + \xi\right) & \lambda = 0 \end{cases}$$

where $Z_t + \xi > 0$ for all $t$. The computational and Box-Cox formulas differ only in the scale and origin of the transformed data. Consequently, the general analysis of the data is unaffected (Draper and Smith 1981, p. 225).

The inverse transformation is computed by

$$X_t = \begin{cases} Z_t^{1/\lambda} - \xi & \lambda \neq 0 \\ exp\left(Z_t\right) - \xi & \lambda = 0 \end{cases}$$

where $\{Z_t\}$ now represents the result computed by `imsls_f_box_cox_transform` for a forward transformation of the original data using parameters λ and ξ.

### *Fatal Errors*

| | |
|---|---|
| `IMSLS_ILLEGAL_SHIFT` | "shift" = # and the smallest element of "z" is "z[#]" = #. "shift" plus "z[#]" = #. "shift" + "z[i]" must be greater than 0 for *i* = 1, ..., "n_observations". "n_observations" = #. |
| `IMSLS_BCTR_CONTAINS_NAN` | One or more elements of "z" is equal to NaN (Not a number). No missing values are allowed. The smallest index of an element of "z" that is equal to NaN is #. |
| `IMSLS_BCTR_F_UNDERFLOW` | Forward transform. "power" = #. "shift" = #. The minimum element of "z" is "z[#]" = #. ("z[#]"+ "shift") ^ "power" will underflow. |
| `IMSLS_BCTR_F_OVERFLOW` | Forward transformation. "power" = #. "shift" = #. The maximum element of "z" is "z[#]" = #. ("z[#]" + "shift") ^ "power" will overflow. |
| `IMSLS_BCTR_I_UNDERFLOW` | Inverse transformation. "power" = #. The minimum element of "z" is "z[#]" = #. exp("z[#]") will underflow. |
| `IMSLS_BCTR_I_OVERFLOW` | Inverse transformation. "power" = #. The maximum element of "z[#]" = #. exp("z[#]") will overflow. |
| `IMSLS_BCTR_I_ABS_UNDERFLOW` | Inverse transformation. "power" = #. The element of "z" with the smallest absolute value is "z[#]" = #. "z[#]" ^ (1/ "power") will underflow. |

| | |
|---|---|
| IMSLS_BCTR_I_ABS_OVERFLOW | Inverse transformation. "power" = #. The element of "z" with the largest absolute value is "z[#]" = #. "z[#]" ^ (1/ "power") will overflow. |

# autocorrelation

Computes the sample autocorrelation function of a stationary time series.

### *Synopsis*

*float* `*imsls_f_autocorrelation` (*int* `n_observations`, *float* `x[]`,
  *int* `lagmax`, ...0)

The type *double* function is `imsls_d_autocorrelation`.

### *Required Arguments*

*int* `n_observations` (Input)
Number of observations in the time series `x`. `n_observations` must be greater than or equal to 2.

*float* `x[]` (Input)
Array of length `n_observations` containing the time series.

*int* `lagmax` (Input)
Maximum lag of autocovariance, autocorrelations, and standard errors of autocorrelations to be computed. `lagmax` must be greater than or equal to 1 and less than `n_observations`.

### *Return Value*

Pointer to an array of length `lagmax` + 1 containing the autocorrelations of the time series `x`. The *0*-th element of this array is 1. The *k*-th element of this array contains the autocorrelation of lag *k* where *k* = 1, ..., `lagmax`.

### *Synopsis with Optional Arguments*

*float* `*imsls_f_autocorrelation` (*int* `n_observations`, *float* `x[]`,
  *int* `lagmax`,
  `IMSLS_PRINT_LEVEL`, *int* `iprint`,

```
IMSLS_X_MEAN_IN, float x_mean_in,
IMSLS_X_MEAN_OUT, float *x_mean_out,
IMSLS_ACV, float **autocovariances,
IMSLS_ACV_USER, float autocovariances[],
IMSLS_SEAC, float **standard_errors, int se_option,
IMSLS_SEAC_USER, float standard_errors[], int se_option,
IMSLS_RETURN_USER, float autocorrelations[],
0)
```

## *Optional Arguments*

`IMSLS_PRINT_LEVEL`, *int* `iprint` (Input)

Printing option.

Default = 0.

| **Iprint** | **Action** |
|---|---|
| 0 | No printing is performed. |
| 1 | Prints the mean and variance. |
| 2 | Prints the mean, variance, and autocovariances. |
| 3 | Prints the mean, variance, autocovariances, autocorrelations, and standard errors of autocorrelations. |

`IMSLS_X_MEAN_IN`, *float* `x_mean_in` (Input)

User input the estimate of the time series `x`.

`IMSLS_X_MEAN_OUT`, *float* `*x_mean_out` (Output)

If specified, `x_mean_out` is the estimate of the mean of the time series `x`.

`IMSLS_ACV`, *float* `**autocovariances` (Output)

Address of a pointer to an array of length `lagmax` + 1 containing the variance and autocovariances of the time series `x`. The *0*-th element of this array is the variance of the time series `x`. The *k*th element contains the autocovariance of lag *k* where *k* = 1, ..., `lagmax`.

`IMSLS_ACV_USER`, *float* `autocovariances[]` (Output)

If specified, `autocovariances` is an array of length `lagmax` + 1 containing the variance and autocovariances of the time series `x`.

See `IMSLS_ACV`.

`IMSLS_SEAC`, *float* `**standard_errors`, *int* `se_option` (Output)

Address of a pointer to an array of length `lagmax` containing the standard errors of the autocorrelations of the time series `x`.

Method of computation for standard errors of the autocorrelations is chosen by `se_option`.

| **se_option** | **Action** |
|---|---|
| 1 | Compute the standard errors of `autocorrelations` using Barlett's formula. |
| 2 | Compute the standard errors of autocorrelations using Moran's formula. |

`IMSLS_SEAC_USER`, *float* `standard_errors[]`, *int* `se_option` (Output)

If specified, `autocovariances` is an array of length `lagmax` containing the standard errors of the autocorrelations of the time series `x`.

See `IMSLS_SEAC`.

`IMSLS_RETURN_USER`, *float* `autocorrelations[]` (Output)

If specified, autocorrelations is an array of length `lagmax` + 1 containing the autocorrelations of the time series x. The *o*th element of this array is 1. The *k*th element of this array contains the autocorrelation of lag $k$ where $k$ = 1, ..., `lagmax`.

### *Description*

Function `imsls_f_autocorrelation` estimates the autocorrelation function of a stationary time series given a sample of $n$ = `n_observations` observations {*X*t} for $t$ = 1, 2, …, $n$.

Let

$$\hat{\mu} = \texttt{x mean}$$

be the estimate of the mean μ of the time series {*X*t} where

$$\hat{\mu} = \begin{cases} \mu, & \mu \text{ known} \\ \dfrac{1}{n}\displaystyle\sum_{t=1}^{n} X_t & \mu \text{ unknown} \end{cases}$$

The autocovariance function σ(k) is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu})(X_{t+k} - \hat{\mu}), \quad k = 0, 1, \ldots, K$$

where *K* = `lagmax`. Note that

$$\hat{\sigma}(0)$$

is an estimate of the sample variance. The autocorrelation function ρ(*k*) is estimated by

$$\hat{\rho}(k) = \frac{\hat{\sigma}(k)}{\hat{\sigma}(0)}, \qquad k = 0, 1, \ldots, K$$

Note that

$$\hat{\rho}(0) \equiv 1$$

by definition.

The standard errors of the sample autocorrelations may be optionally computed according to argument `se_option` for the optional argument `IMSLS_SEAC`. One method (Bartlett 1946) is based on a general asymptotic expression for the variance of the sample autocorrelation coefficient of a stationary time series with independent, identically distributed normal errors. The theoretical formula is

$$\operatorname{var}\{\hat{\rho}(\mathrm{k})\} = \frac{1}{n}\sum_{i=-\infty}^{\infty}\left[\rho^2(i) + \rho(i-k)\rho(i+k) - 4\rho(i)\rho(k)\rho(i-k) + 2\rho^2(i)\rho^2(k)\right]$$

where

$$\hat{\rho}(k)$$

assumes μ is unknown. For computational purposes, the autocorrelations r(k) are replaced by their estimates

$$\hat{\rho}(k)$$

for |k| ≤ K, and the limits of summation are bounded because of the assumption that r(k) = 0 for all k such that |k| > K.

A second method (Moran 1947) utilizes an exact formula for the variance of the sample autocorrelation coefficient of a random process with independent, identically distributed normal errors. The theoretical formula is

$$\operatorname{var}\left\{\hat{\rho}(k)\right\} = \frac{n-k}{n(n+2)}$$

where μ is assumed to be equal to zero. Note that this formula does not depend on the autocorrelation function.

## partial_autocorrelation

Computes the sample partial autocorrelation function of a stationary time series.

### *Synopsis*

*float* `*imsls_f_partial_autocorrelation` (*int* `lagmax`, *int* `cf[]`, …, 0)

The type *double* function is `imsls_d_partial_autocorrelation`.

### *Required Arguments*

*int* lagmax (Input)

Maximum lag of partial autocorrelations to be computed.

*float* cf[] (Input)

Array of length lagmax + 1 containing the autocorrelations of the time series x.

### *Return Value*

Pointer to an array of length lagmax containing the partial autocorrelations of the time series x.

### *Synopsis with Optional Arguments*

*float* *imsls_f_partial_autocorrelation (*int* lagmax, *float* cf[],
IMSLS_RETURN_USER, *float* partial_autocorrelations[],
0)

### *Optional Arguments*

IMSLS_RETURN_USER, *float* partial_autocorrelations[] (Output)

If specified, the partial autocorrelations are stored in an array of length lagmax provided by the user.

### *Description*

Function imsls_f_partial_autocorrelation estimates the partial autocorrelations of a stationary time series given the *K* = lagmax sample autocorrelations

$$\hat{\rho}(k)$$

for *k* = 0, 1, …, *K*. Consider the AR(*k*) process defined by

$$X_t = \phi_{k1} X_{t-1} + \phi_{k2} X_{t-2} + \ldots + \phi_{kk} X_{t-k} + A_t$$

where $\phi$kj denotes the *j*-th coefficient in the process. The set of estimates

$$\left\{\hat{\phi}_{kk}\right\}$$

for $k = 1, \ldots, K$ is the sample partial autocorrelation function. The autoregressive parameters

$$\left\{\hat{\phi}_{kj}\right\}$$

for $j = 1, \ldots, k$ are approximated by Yule-Walker estimates for successive AR($k$) models where $k = 1, \ldots, K$. Based on the sample Yule-Walker equations

$$\hat{\rho}(j) = \hat{\phi}_{k1}\hat{\rho}(j-1) + \hat{\phi}_{k2}\hat{\rho}(j-2) + \ldots + \hat{\phi}_{kk}\hat{\rho}(j-k), \quad j = 1, 2, \ldots, k$$

a recursive relationship for $k = 1, \ldots, K$ was developed by Durbin (1960). The equations are given by

$$\hat{\phi}_{kk} = \begin{cases} \hat{\rho}(1) & k = 1 \\ \dfrac{\hat{\rho}(k) - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(k-j)}{1 - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(j)} & k = 2, \ldots, K \end{cases}$$

and

$$\hat{\phi}_{kk} = \begin{cases} \hat{\rho}(1) & k = 1 \\ \dfrac{\hat{\rho}(k) - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(k-j)}{1 - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}(j)} & k = 2, \ldots, K \end{cases}$$

This procedure is sensitive to rounding error and should not be used if the parameters are near the nonstationarity boundary. A possible alternative would be to estimate $\{\phi_{kk}\}$ for successive AR($k$) models using least or maximum likelihood. Based on the hypothesis that the true process is AR($p$), Box and Jenkins (1976, page 65) note

$$\text{var}\{\hat{\phi}_{kk}\} \simeq \frac{1}{n} \quad k \geq p+1$$

See Box and Jenkins (1976, pages 82–84) for more information concerning the partial autocorrelation function.

# lack_of_fit

Performs lack-of-fit test for a univariate time series or transfer function given the appropriate correlation function.

## *Synopsis*

*float* `*imsls_lack_of_fit` (*int* `n_observations`, *float* `cf[]`,
*int* `lagmax`, *int* `npfree`,..., 0)

## *Required Arguments*

*int* `n_observations` (Input)
Number of observations of the stationary time series.

*float* `cf[]` (Input)
Array of length `lagmax`+1 containing the correlation function.

*int* `lagmax` (Input)
Maximum lag of the correlation function.

*int* `npfree` (Input)
Number of free parameters in the formulation of the time series model. `npfree` must be greater than or equal to zero and less than `lagmax`. Woodfield (1990) recommends `npfree = p + q`.

## *Return Value*

Pointer to an array of length 2 with the test statistic, Q, and its *p*-value, *p*. Under the null hypothesis, Q has an approximate chi-squared distribution with `lagmax-lagmin+1-npfree` degrees of freedom.

## *Synopsis with Optional Arguments*

*#include* `<imsls.h>`

*float* `*imsls_f_lack_of_fit` (*int* `n_observations`, *float* `cf[]`, *int* `lagmax`,
*int* `npfree`,
`IMSLS_LAGMIN`, *int* `lagmin`,
`IMSLS_RETURN_USER`, *float* `stat[]`,
0)

### Optional Arguments

`IMSLS_LAGMIN`, *int* `lagmin` (Input)
Minimum lag of the correlation function. `lagmin` corresponds to the lower bound of summation in the lack of fit test statistic. Default value is 1.

`IMSLS_RETURN_USER`, *float* `stat[]` (Output)
User defined array for storage of lack-of-fit statistics.

### Description

Routine `imsls_f_lack_of_fit` may be used to diagnose lack of fit in both ARMA and transfer function models. Typical arguments for these situations are:

| Model | LAGMIN | LAGMAX | NPFREE |
|---|---|---|---|
| ARMA (*p, q*) | 1 | $\sqrt{\mathrm{NOBS}}$ | $p + q$ |
| Transfer function | 0 | $\sqrt{\mathrm{NOBS}}$ | $r + s$ |

Function `imsls_f_lack_of_fit` performs a portmanteau lack of fit test for a time series or transfer function containing `n` observations given the appropriate sample correlation function

$$\hat{\rho}(k)$$

for $k = L, L + 1, \ldots, K$ where $L$ = `lagmin` and $K$ = `lagmax`.

The basic form of the test statistic $Q$ is

$$Q = n(n+2)\sum_{k=L}^{K}(n-k)^{-1}\hat{\rho}(k)$$

with $L = 1$ if

$$\hat{\rho}(k)$$

is an autocorrelation function. Given that the model is adequate, *Q* has a chi-squared distribution with *K* – *L* + 1 – *m* degrees of freedom where *m* = `npfree` is the number of parameters estimated in the model. If the mean of the time series is estimated, Woodfield (1990) recommends not including this in the count of the parameters estimated in the model. Thus, for an ARMA(*p*, *q*) model set `npfree`= *p* + *q* regardless of whether the mean is estimated or not. The original derivation for time series models is due to Box and Pierce (1970) with the above modified version discussed by Ljung and Box (1978). The extension of the test to transfer function models is discussed by Box and Jenkins (1976, pages 394–395).

# estimate_missing

Estimates missing values in a time series.

## *Synopsis*

*float* `*imsls_f_estimate_missing(`*int* `n_obs`, *int* `tpoints[]`,
*float* `z[]`, …, 0)

The type *double* function is `imsls_d_estimate_missing`.

## *Required Arguments*

*int* `n_obs` (Input)
Number of non-missing observations in the time series. The time series must not contain gaps with more than 3 missing values.

*int* `tpoints[]` (Input)
Vector of length `n_obs` containing the time points $t_1, \ldots, t_{n\_obs}$ at which the time series values were observed. The time points must be in strictly increasing order. Time points for missing values must lie in the open interval $(t_1, t_{n\_obs})$.

*float* `z[]` (Input)
Vector of length `n_obs` containing the time series values. The values must be ordered in accordance with the values in vector `tpoints`. It is assumed that the time series after estimation of missing values contains values at equidistant time points where the distance

between two consecutive time points is one. If the non-missing time series values are observed at time points $t_1, \ldots, t_{n\_obs}$, then missing values between $t_i$ and $t_{i+1}$, $i = 1, \ldots, \texttt{n\_obs} - 1$, exist if $t_{i+1} - t_i > 1$. The size of the gap between $t_i$ and $t_{i+1}$ is then $t_{i+1} - t_i - 1$. The total length of the time series with non-missing and estimated missing values is $t_{n\_obs} - t_1 + 1$, or `tpoints[n_obs-1]-tpoints[0]+1`.

### *Return Value*

Pointer to an array of length (`tpoints[n_obs-1]-tpoints[0]+1`) containing the time series together with estimates for the missing values. If an error occurred, `NULL` is returned.

### *Synopsis with Optional Arguments*

```
float *imsls_f_estimate_missing (int n_obs, int tpoints[], float z[],
      IMSLS_METHOD, int method,
      IMSLS_MAX_LAG, int maxlag,
      IMSLS_NTIMES, int *ntimes,
      IMSLS_MEAN_ESTIMATE, float mean_estimate,
      IMSLS_CONVERGENCE_TOLERANCE, float convergence_tolerance,
      IMSLS_RELATIVE_ERROR, float relative_error,
      IMSLS_MAX_ITERATIONS, int max_iterations,
      IMSLS_TIMES_ARRAY, int **times,
      IMSLS_TIMES_ARRAY_USER, int times[],
      IMSLS_MISSING_INDEX, int **missing_index,
      IMSLS_MISSING_INDEX_USER, int missing_index[],
      IMSLS_RETURN_USER, float u_z[],
      0)
```

### *Optional Arguments*

`IMSLS_METHOD`, *int* `method` (Input)

The method used for estimating the missing values:

0 — Use median.
1 — Use cubic spline interpolation.
2 — Use one-step-ahead forecasts from an AR(1) model.
3 — Use one-step-ahead forecasts from an AR(p) model.

Default: `method` = 3

If `method = 2` is chosen, then all values of gaps beginning at time points $t_1 + 1$ or $t_1 + 2$ are estimated by `method` 0. If `method` = 3 is chosen and the first gap starts at $t_1 + 1$, then the values of this gap are also estimated by `method` 0. If the length of the series before a gap, denoted `len`, is greater than 1 and less than `2·maxlag`, then maxlag is reduced to `len/2` for the computation of the missing values within this gap.

`IMSLS_MAX_LAG`, *int* `maxlag` (Input)

Maximum lag number when `method` = 3 was chosen.

Default: `maxlag` = 10

`IMSLS_NTIMES`, *int* `*ntimes` (Output)

Number of elements in the time series with estimated missing values. Note that `ntimes = tpoints[n_obs-1]-tpoints[0]+1`.

`IMSLS_MEAN_ESTIMATE`, *float* `mean_estimate` (Input)

Estimate of the mean of the time series.

`IMSLS_CONVERGENCE_TOLERANCE`, *float* `convergence_tolerance` (Input)

Tolerance level used to determine convergence of the nonlinear least squares algorithm used in method 2. Argument `convergence_tolerance` represents the minimum relative decrease in the sum of squares between two iterations required to determine convergence. Hence, `convergence_tolerance` must be greater than or equal to 0.

Default: $\max\{10^{-10}, \text{eps}^{2/3}\}$ for single precision and $\max\{10^{-20}, \text{eps}^{2/3}\}$ for double precision, where eps `=imsls_f_machine(4)` for single precision and eps `=imsls_d_machine(4)` for double precision.

`IMSLS_RELATIVE_ERROR`, *float* `relative_error` (Input)

Stopping criterion for use in the nonlinear equation solver used by method 2.

Default: `relative_error` = 100 × `imsls_f_machine(4)` for single precision, `relative_error` = 100 × `imsls_d_machine(4)` for double precision..

`IMSLS_MAX_ITERATIONS`, *int* `max_iterations` (Input)

Maximum number of iterations allowed in the nonlinear equations solver used by method 2.

Default: `max_iterations` = 200.

IMSLS_TIMES_ARRAY, *int* **times (Output)

Address of a pointer to an internally allocated array of length ntimes = tpoints[n_obs-1]-tpoints[0]+1 containing the time points of the time series with estimates for the missing values.

IMSLS_TIMES_ARRAY_USER, *int* times[] (Output)

Storage for array times is provided by the user. See IMSLS_TIMES_ARRAY.

IMSLS_MISSING_INDEX, *int* **missing_index (Output)

Address of a pointer to an internally allocated array of length (ntimes-n_obs) containing the indices for the missing values in array times. If ntimes-n_obs = 0, then no missing value could be found and NULL is returned.

IMSLS_MISSING_INDEX_USER, *int* missing_index[] (Output)

Storage for array missing_index is provided by the user. See IMSLS_MISSING_INDEX.

IMSLS_RETURN_USER, *float* u_z[] (Output)

If specified, u_z is a vector of length tpoints[n_obs-1]-tpoints[0]+1 containing the time series values together with estimates for missing values.

## *Description*

Traditional time series analysis as described by Box, Jenkins and Reinsel (1994) requires the observations made at equidistant time points $t_1, t_1+1, t_1+2, \ldots, t_n$. When observations are missing, the problem occurs to determine suitable estimates. Function imsls_f_estimate_missing offers 4 estimation methods:

Method 0 estimates the missing observations in a gap by the median of the last four time series values before and the first four values after the gap. If not enough values are available before or after the gap then the number is reduced accordingly. This method is very fast and simple, but its use is limited to stationary ergodic series without outliers and level shifts.

Method 1 uses a cubic spline interpolation method to estimate missing values. Here the interpolation is again done over the last four time series values before and the first four values after the gap. The missing values are estimated by the resulting interpolant. This method gives smooth transitions across missing values.

Method 2 assumes that the time series before the gap can be well described by an AR(1) process. If the last observation prior to the gap is made at time point $t_m$ then it uses the time series values at $t_1, t_1+1, \ldots, t_m$ to compute the one-step-ahead forecast at origin $t_m$. This value is taken as an estimate for the missing value at time point $t_{m+1}$. If the value at $t_{m+1}$ is also missing then the values at time points $t_1, t_1+1, \ldots, t_{m+1}$ are used to recompute the AR(1) model, estimate the value at $t_{m+2}$ and so on. The coefficient $\phi_1$ in the AR(1) model is computed internally by the method of least squares from routine `imsls_f_arma`.

Finally, method 3 uses an AR(p) model to estimate missing values by a one-step-ahead forecast . First, function `imsls_f_auto_uni_ar`, applied to the time series prior to the missing values, is used to determine the optimum `p` from the set {0, 1, …, `max_lag`} of possible values and to compute the parameters $\phi_1, \ldots, \phi_p$ of the resulting AR(p) model. The parameters are estimated by the least squares method based on Householder transformations as described in Kitagawa and Akaike (1978). Denoting the mean of the series $y_{t_1}, y_{t_1+1}, \ldots, y_{t_m}$ by μ the one-step-ahead forecast at origin $t_m$, $\hat{y}_{t_m}(1)$, can be computed by the formula

$$\hat{y}_{t_m}(1) = \mu\left(1 - \sum\nolimits_{j=1}^{p} \phi_j\right) + \sum\nolimits_{j=1}^{p} \phi_j y_{t_m+1-j} \ .$$

This value is used as an estimate for the missing value. The procedure starting with `imsls_f_auto_uni_ar` is then repeated for every further missing value in the gap. All four estimation methods treat gaps of missing values in increasing time order.

# Reference Material

## User Errors

IMSL functions attempt to detect user errors and handle them in a way that provides as much information to the user as possible. To do this, various levels of severity of errors are recognized, and the extent of the error in the context of the purpose of the function also is considered; a trivial error in one situation can be serious in another. IMSL attempts to report as many errors as can reasonably be detected. Multiple errors present a difficult problem in error detection because input is interpreted in an uncertain context after the first error is detected.

### What Determines Error Severity

In some cases, the user's input may be mathematically correct, but because of limitations of the computer arithmetic and of the algorithm used, it is not possible to compute an answer accurately. In this case, the assessed degree of accuracy determines the severity of the error. In cases where the function computes several output quantities, some are not computable but most are, an error condition exists. The severity of the error depends on an assessment of the overall impact of the error.

### Kinds of Errors and Default Actions

Five levels of severity of errors are defined in IMSL C/Stat/Library. Each level has an associated PRINT attribute and a STOP attribute. These attributes have default settings (YES or NO), but they may also be set by the user. The purpose of having multiple error types is to provide independent control of actions to be taken for errors of different levels of severity. Upon return from an IMSL function, exactly one error state exists. (A code 0 "error" is no error.) Even if more than one informational error occurs, only one message is printed (if the PRINT attribute is YES). Multiple errors for which no corrective action within the calling program is reasonable or necessary result in the printing of multiple messages (if the PRINT attribute for their severity level is YES). Errors of any of the severity levels except `IMSLS_TERMINAL` may be informational errors. The include file, *imsls.h*, defines each of `IMSLS_NOTE`, `IMSLS_ALERT`, `IMSLS_WARNING`, `IMSLS_FATAL`, `IMSLS_TERMINAL`, `IMSLS_WARNING_IMMEDIATE`, and `IMSLS_FATAL_IMMEDIATE` as enumerated data type *Imsls_error*.

`IMSLS_NOTE`. A *note* is issued to indicate the possibility of a trivial error or simply to provide information about the computations.
Default attributes: PRINT=NO, STOP=NO

`IMSLS_ALERT`. An *alert* indicates that a function value has been set to 0 due to underflow.
Default attributes: PRINT=NO, STOP=NO

`IMSLS_WARNING`. A *warning* indicates the existence of a condition that may require corrective action by the user or calling function. A warning error may be issued because the results are accurate to only a few decimal places; because some of the output may be erroneous, but most of the output is correct; or because some assumptions underlying the analysis technique are violated. Usually no corrective action is necessary, and the condition can be ignored.
Default attributes: PRINT=YES, STOP=NO

`IMSLS_FATAL`. A *fatal* error indicates the existence of a condition that may be serious. In most cases, the user or calling function must take corrective action to recover.
Default attributes: PRINT=YES, STOP=YES

`IMSLS_TERMINAL`. A *terminal* error is serious. It usually is the result of an incorrect specification, such as specifying a negative number as the number of equations. These errors can also be caused by various programming errors impossible to diagnose correctly in C. The resulting error message may be perplexing to the user. In such cases, the user is advised to compare carefully the actual arguments passed to the function with the dummy argument descriptions given in the documentation. Special attention should be given to checking argument order and data types.

A terminal error is not an informational error, because corrective action within the program is generally not reasonable. In normal use, execution is terminated immediately when a terminal error occurs. Messages relating to more than one terminal error are printed if they occur.
Default attributes: PRINT=YES, STOP=YES

`IMSLS_WARNING_IMMEDIATE`. An *immediate warning* error is identical to a warning error, except it is printed immediately.
Default attributes: PRINT=YES, STOP=NO

`IMSLS_FATAL_IMMEDIATE`. An *immediate fatal* error is identical to a fatal error, except it is printed immediately.
Default attributes: PRINT=YES, STOP=YES

The user can set PRINT and STOP attributes by calling function `imsls_error_options`.

# Product Support

## Contacting Visual Numerics Support

Users within support warranty may contact Visual Numerics regarding the use of the IMSL C Numerical Library. Visual Numerics can consult on the following topics:

- Clarity of documentation
- Possible Visual Numerics-related programming problems
- Choice of IMSL Libraries functions or procedures for a particular problem

Not included in these topics are mathematical/statistical consulting and debugging of your program.

**Refer to the following for Visual Numerics Product Support contact information:**

- http://www.vni.com/tech/imsl/phone.php

The following describes the procedure for consultation with Visual Numerics:

1. Include your Visual Numerics license number
2. Include the product name and version number: IMSL C Numerical Library Version 7.0
3. Include compiler and operating system version numbers
4. Include the name of the routine for which assistance is needed and a description of the problem

# Appendix A

## References


***Abe***

Abe, S. (2001) Pattern Classification: Neuro-Fuzzy Methods and their Comparison, Springer-Verlag.

***Abramowitz and Stegun***

Abramowitz, Milton and Irene A. Stegun (editors) (1964), *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, Washington.

***Afifi and Azen***

Afifi, A.A. and S.P. Azen (1979), *Statistical Analysis: A Computer Oriented Approach*, 2d ed., Academic Press, New York.

***Agresti, Wackerly, and Boyette***

Agresti, Alan, Dennis Wackerly, and James M. Boyette (1979), Exact conditional tests for cross-classifications: Approximation of attained significance levels, *Psychometrika*, **44**, 75-83.

***Aha***

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach.

In Proceedings of the Eighth International Workshop on Machine Learning (pp. 117--121). Evanston, ILL: Morgan Kaufmann.

***Ahrens and Dieter***

Ahrens, J.H. and U. Dieter (1974), Computer methods for sampling from gamma, beta, Poisson, and binomial distributions, *Computing,* **12**, 223–246.

Ahrens, J.H., and U. Dieter (1985), Sequential random sampling, *ACM Transactions on Mathematical Software*, **11**, 157−169.

### *Akaike*

Akaike, H., (1978), *Covariance Matrix Computation of the State Variable of a Stationary Gaussian Process*, Ann. Inst. Statist. Math. 30 , Part B, 499-504.

### *Akaike et al*

Akaike, H. , Kitagawa, G., Arahata, E., Tada, F., (1979), Computer Science Monographs No. 13, The Institute of Statistical Mathematics, Tokyo.

### *Anderberg*

Anderberg, Michael R. (1973), *Cluster Analysis for Applications*, Academic Press, New York.

### *Anderson*

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, John Wiley & Sons, New York.

Anderson, T. W. (1994) *The Statistical Analysis of Time Series*, John Wiley & Sons, New York.

### *Anderson and Bancroft*

Anderson, R.L. and T.A. Bancroft (1952), *Statistical Theory in Research*, McGraw-Hill Book Company, New York.

### *Asuncion and Newman*

Asuncion, A.and Newman, D.J. (2007), UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science.

### *Atkinson*

Atkinson, A.C. (1979), A Family of Switching Algorithms for the Computer Generation of Beta Random Variates, *Biometrika*, **66**, 141–145.

Atkinson, A.C. (1985), *Plots, Transformations, and Regression*, Claredon Press, Oxford.

### *Baker*

Baker, J. E. (1987), Reducing Bias and Inefficiency in the Selection Algorithm. *Genetic Algorithms and their Applications: Proceeding of the Second international Conference on Genetic Algorithms*, 14-21.

### *Barrodale and Roberts*

Barrodale, I., and F.D.K. Roberts (1973), An improved algorithm for discrete $L_1$ approximation, *SIAM Journal on Numerical Analysis*, **10**, 839−848.

Barrodale, I., and F.D.K. Roberts (1974), Solution of an overdetermined system of equations in the $l_1$ norm, *Communications of the ACM*, **17**, 319−320.

Barrodale, I., and C. Phillips (1975), Algorithm 495. Solution of an overdetermined system of linear equations in the Chebyshev norm, *ACM Transactions on Mathematical Software*, **1**, 264−270.

### *Bartlett, M. S.*

Bartlett, M.S. (1935), Contingency table interactions, *Journal of the Royal Statistics Society Supplement*, **2**, 248–252.

Bartlett, M. S. (1937) Some examples of statistical methods of research in agriculture and applied biology*, Supplement to the Journal of the Royal Statistical Society*, **4**, 137-183.

Bartlett, M. (1937), The statistical conception of mental factors, *British Journal of Psychology*, **28**, 97–104.

Bartlett, M.S. (1946), On the theoretical specification and sampling properties of autocorrelated time series, *Supplement to the Journal of the Royal Statistical Society*, **8**, 27–41.

Bartlett, M.S. (1978), *Stochastic Processes,* 3rd. ed., Cambridge University Press, Cambridge.

### *Bays and Durham*

Bays, Carter and S.D. Durham (1976), Improving a poor random number generator, *ACM Transactions on Mathematical Software*, **2**, 59–64.

### *Bendel and Mickey*

Bendel, Robert B., and M. Ray Mickey (1978), Population correlation matrices for sampling experiments, *Communications in Statistics*, **B7**, 163−182.

### *Berry*

Berry, M. J. A. and Linoff, G. (1997) *Data Mining Techniques*, John Wiley & Sons, Inc.

### *Best and Fisher*

Best, D.J., and N.I. Fisher (1979), Efficient simulation of the von Mises distribution, *Applied Statistics*, **28**, 152−157.

### *Bishop*

Bishop, C. M. (1995) *Neural Networks for Pattern Recognition*, Oxford University Press.

### *Bishop et al*

Bishop, Yvonne M.M., Stephen E. Feinberg, and Paul W. Holland (1975), *Discrete Multivariate Analysis: Theory and Practice*, MIT Press, Cambridge, Mass.

### *Bjorck and Golub*

Bjorck, Ake, and Gene H. Golub (1973), Numerical Methods for Computing Angles Between Subspaces, *Mathematics of Computation,* **27**, 579–594.

### *Blom*

Blom, Gunnar (1958), *Statistical Estimates and Transformed Beta-Variables*, John Wiley & Sons, New York.

### *Bosten and Battiste*

Bosten, Nancy E. and E.L. Battiste (1974), Incomplete beta ratio, *Communications of the ACM*, **17**, 156s–157.

### *Box and Jenkins*

Box, George E.P. and Gwilym M. Jenkins (1970) *Time Series Analysis: Forecasting and Control*, Holden-Day, Inc.

Box, George E.P. and Gwilym M. Jenkins (1976), *Time Series Analysis: Forecasting and Control*, revised ed., Holden-Day, Oakland.

### *Box and Pierce*

Box, G.E.P., and David A. Pierce (1970), Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, *Journal of the American Statistical Association*, **65**, 1509–1526.

### *Box and Tidwell*

Box, G.E.P. and P.W. Tidwell (1962), Transformation of the Independent Variables, *Technometrics*, **4**, 531–550.

### *Box et al.*

Box, George E.P., Jenkins,Gwilym M. and Reinsel G.C., (1994) *Time Series Analysis*, Third edition, Prentice Hall, Englewood Cliffs, New Jersey.

### *Boyette*

Boyette, James M. (1979), Random RC tables with given row and column totals, *Applied Statistics*, **28**, 329−332.

### *Bradley*

Bradley, J.V. (1968), *Distribution-Free Statistical Tests*, Prentice-Hall, New Jersey.

### *Breiman*

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984) *Classification and Regression Trees*, Chapman & Hall.  For the latest information on CART visit: http://www.salford-systems.com/cart.php.

***Breslow***

Breslow, N.E. (1974), Covariance analysis of censored survival data, *Biometrics*, **30**, 89–99.

***Bridel***

Bridle, J. S. (1990) Probabilistic Interpretation of Feedforward Classification Network Outputs, with relationships to statistical pattern recognition, in F. Fogelman Soulie and J. Herault (Eds.), *Neuralcomputing:  Algorithms, Architectures and Applications*, Springer-Verlag, 227-236.

***Brown***

Brown, Morton E. (1983), MCDP4F, two-way and multiway frequency tables-measures of association and the log-linear model (complete and incomplete tables), in *BMDP Statistical Software, 1983 Printing with Additions*, (edited by W.J. Dixon), University of California Press, Berkeley.

***Brown and Benedetti***

Brown, Morton B. and Jacqualine K. Benedetti (1977), Sampling behavior and tests for correlation in two-way contingency tables, *Journal of the American Statistical Association*, **42**, 309–315.

***Calvo***

Calvo, R. A. (2001) Classifying Financial News with Neural Networks, *Proceedings of the 6th Australasian Document Computing Symposium*.

***Chen and Liu***

Chen, C. and Liu, L., *Joint Estimation of Model Parameters and Outlier Effects in Time Series*, Journal of the American Statistical Association, Vol. 88, No.421, March 1993.

***Cheng***

Cheng, R.C.H. (1978), Generating beta variates with nonintegral shape parameters, *Communications of the ACM*, **21**, 317–322.

### *Chiang*

Chiang, Chin Long (1968), *Introduction to Stochastic Processes in Statistics*, John Wiley & Sons, New York.

### *Clarkson and Jenrich*

Clarkson, Douglas B. and Robert B Jenrich (1991), Computing extended maximum likelihood estimates for linear parameter models, submitted to *Journal of the Royal Statistical Society*, Series B, **53**, 417-426.

### *Coley*

Coley, D. A. (1999), *An Introduction to Genetic Algorithms for Scientists and Engineers*, World Scientific Publishing Co.

### *Conover*

Conover, W.J. (1980), *Practical Nonparametric Statistics*, 2d ed., John Wiley & Sons, New York.

### *Conover and Iman*

Conover, W.J. and Ronald L. Iman (1983), *Introduction to Modern Business Statistics*, John Wiley & Sons, New York.

### *Conover, W. J., Johnson, M. E., and Johnson, M. M*

Conover, W. J., Johnson, M. E., and Johnson, M. M. (1981) A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data, *Technometrics*, **23**, 351-361.

### *Cook and Weisberg*

Cook, R. Dennis and Sanford Weisberg (1982), *Residuals and Influence in Regression*, Chapman and Hall, New York.

***Cooper***

Cooper, B.E. (1968), Algorithm AS4, An auxiliary function for distribution integrals, *Applied Statistics*, **17**, 190–192.

***Cox***

Cox, David R. (1970), *The Analysis of Binary Data*, Methuen, London.

Cox, D.R. (1972), Regression models and life tables (with discussion), *Journal of the Royal Statistical Society*, Series B, *Methodology*, **34**, 187–220.

***Cox and Lewis***

Cox, D.R., and P.A.W. Lewis (1966), *The Statistical Analysis of Series of Events*, Methuen, London.

***Cox and Oakes***

Cox, D.R., and D. Oakes (1984), *Analysis of Survival Data*, Chapman and Hall, London.

***Cox and Stuart***

Cox, D.R., and A. Stuart (1955), Some quick sign tests for trend in location and dispersion, *Biometrika*, **42**, 80–95.

***Cranley and Patterson***

Cranley, R. and Patterson, T.N.L. (1976), Randomization of Number Theoretic Methods for Multiple Integration, *SIAM Journal of Numerical Analysis*, **13**, 904-914.

***D'Agostino and Stevens***

D'Agostino, Ralph B. and Michael A. Stevens (1986), *Goodness-of-Fit Techniques*, Marcel Dekker, New York.

***Dallal and Wilkinson***

Dallal, Gerald E. and Leland Wilkinson (1986), An analytic approximation to the distribution of Lilliefor's test statistic for normality, *The American Statistician*, **40**, 294–296.

### *Davis and Rabinowitz*

Davis, P.J. and Rabinowitz, P. (1984), *Methods of Numerical Integration*, Academic Press, 482-483.

### *De Jong*

De Jong, K.  A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems. (Doctorial dissertation, Univ. of Michigan).  *Dissertation Abstracts International* 36(10), 5140B. (University Microfilms No. 76-9381).

### *Demiroz et al.*

Demiroz, G., H. A. Govenir, and N. Ilter (1988), “Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals”, Artificial Intelligence in Medicine.

### *Dennis and Schnabel*

Dennis, J.E., Jr. and Robert B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

### *Devore*

Devore, Jay L (1982), *Probability and Statistics for Engineering and Sciences*, Brooks/Cole Publishing Company, Monterey, Calif.

### *Doornik*

Doornik, J.A. (2005), An Improved Ziggurat Method to Generate Normal Random Samples, http://www.doornik.com/research/ziggurat.pdf., University of Oxford.

### *Draper and Smith*

Draper, N.R. and H. Smith (1981), *Applied Regression Analysis*, 2d ed., John Wiley & Sons, New York.

### *Dunnett and Sobel*

Dunnett, C. W. and Sobel, M. (1955),  Approximations to the Probability Integral and Certain Percentage Points of a Multivariate Analogue of Student’s *t*-distribution.  *Biometrika*, **42**, 258-260.

### *Durbin*

Durbin, J. (1960), The fitting of time series models, *Revue Institute Internationale de Statistics*, **28**, 233–243.

### *Efroymson*

Efroymson, M.A. (1960), Multiple regression analysis, *Mathematical Methods for Digital Computers*, Volume 1, (edited by A. Ralston and H. Wilf), John Wiley & Sons, New York, 191–203.

### *Ekblom*

Ekblom, Hakan (1973), Calculation of linear best $L_p$-approximations, *BIT*, **13**, 292−300.

Ekblom, Hakan (1987), The $L_1$-estimate as limiting case of an $L_p$ or Huber-estimate, in *Statistical Data Analysis Based on the $L_1$-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 109−116.

### *Elandt-Johnson and Johnson*

Elandt-Johnson, Regina C., and Norman L. Johnson (1980), *Survival Models and Data Analysis*, John Wiley & Sons, New York, 172–173.

### *Elman*

Elman, J. L. (1990)  Finding Structure in Time, *Cognitive Science,* 14, 179-211.

### *Emmett*

Emmett, W.G. (1949), Factor analysis by Lawless method of maximum likelihood, *British Journal of Psychology, Statistical Section*, **2**, 90–97.

### *Engle*

Engle, C. (1982), Autoregressive conditional heteroskedasticity with estimates of the variance of U.K. inflation, *Econometrica ,*  **50**, 987–1008.

### *Fisher*

Fisher, R.A. (1936), The use of multiple measurements in taxonomic problems, *The Annals of Eugenics*, **7**, 179–188.

### *Fishman*

Fishman, George S. (1978), *Principles of Discrete Event Simulation*, John Wiley & Sons, New York.

### *Fishman and Moore*

Fishman, George S. and Louis R. Moore (1982), A statistical evaluation of multiplicative congruential random number generators with modulus , *Journal of the American Statistical Association*, **77**, 129–136.

### *Forsythe*

Forsythe, G.E. (1957), Generation and use of orthogonal polynomials for fitting data with a digital computer, *SIAM Journal on Applied Mathematics*, **5**, 74–88.

### *Frey and Slate*

Frey, P. W. and D. J. Slate. (1991), “Letter Recognition using Holland-style Adaptive Classifiers”. (Machine Learning *Vol* 6 #2).

### *Fuller*

Fuller, Wayne A. (1976), *Introduction to Statistical Time Series*, John Wiley & Sons, New York.

### *Furnival and Wilson*

Furnival, G.M. and R.W. Wilson, Jr. (1974), Regressions by leaps and bounds, *Technometrics*, **16**, 499–511.

### *Fushimi*

Fushimi, Masanori (1990), Random number generation with the recursion $X_t = X_{t-3p} \oplus X_{t-3q}$, *Journal of Computational and Applied Mathematics*, **31**, 105−118.

### *Gentleman*

Gentleman, W. Morven (1974), Basic procedures for large, sparse or weighted linear least squares problems, *Applied Statistics*, **23**, 448–454.

***Genz***

Genz, A. (1992), Numerical Computation of Multivariate Normal Probabilities. *J. Comp. Graph Stat*., **1**, 141-149.

***Gibbons***

Gibbons, J.D. (1971), *Nonparametric Statistical Inference*, McGraw-Hill, New York.

***Girschick***

Girschick, M.A. (1939), On the Sampling Theory of Roots of Determinantal Equations, *Annals of Mathematical Statistics*, **10**, 203–224.

***Goldberg***

Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Co.

Goldberg, D. E. and Deb, K. (1991), A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In G. Rawlins, Ed., *Foundations of Genetic Algorithms*. Morgan Kaufmann.

***Golub and Van Loan***

Golub, Gene H. and Charles F. Van Loan (1983), *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md.

***Gonin and Money***

Gonin, Rene, and Arthur H. Money (1989), *Nonlinear* $L_p$*-Norm Estimation*, Marcel Dekker, New York.

***Goodnight***

Goodnight, James H. (1979), A tutorial on the SWEEP operator, *The American Statistician*, **33**, 149–158.

### *Graybill*

Graybill, Franklin A. (1976), *Theory and Application of the Linear Model*, Duxbury Press, North Scituate, Mass.

### *Griffin and Redish*

Griffin, R. and K.A. Redish (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*,
**13**, 54.

### *Gross and Clark*

Gross, Alan J., and Virginia A. Clark (1975), *Survival Distributions: Reliability Applications in the Biomedical Sciences*, John Wiley & Sons, New York.

### *Gruenberger and Mark*

Gruenberger, F., and A.M. Mark (1951), The $d^2$ test of random digits, *Mathematical Tables and Other Aids in Computation*, **5**, 109–110.

### *Guerra et al.*

Guerra, Victor O., Richard A. Tapia, and James R. Thompson (1976), A random number generator for continuous random variables based on an interpolation procedure of Akima, in *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, (edited by David C. Hoaglin and Roy E. Welsch), Prindle, Weber & Schmidt, Boston, 228−230.

### *Giudici*

Giudici, P. (2003)  *Applied Data Mining:  Statistical Methods for Business and Industry*, John Wiley & Sons, Inc.

### *Haldane*

Haldane, J.B.S. (1939), The mean and variance of  when used as a test of homogeneity, when expectations are small, *Biometrika*, **31**, 346.

***Hamilton***

Hamilton, James D., *Time Series Analysis*, Princeton University Press, Princeton (NewJersey), 1994.

***Harman***

Harman, Harry H. (1976), *Modern Factor Analysis*, 3d ed. revised, University of Chicago Press, Chicago.

***Hart et al***

Hart, John F., E.W. Cheney, Charles L. Lawson, Hans J. Maehly, Charles K. Mesztenyi, John R. Rice, Henry G. Thacher, Jr., and Christoph Witzgall (1968), *Computer Approximations*, John Wiley & Sons, New York.

***Hartigan***

Hartigan, John A. (1975), *Clustering Algorithms*, John Wiley & Sons, New York.

***Hartigan and Wong***

Hartigan, J.A. and M.A. Wong (1979), Algorithm AS 136: A *K*-means clustering algorithm, *Applied Statistics*, **28**, 100–108.

***Hayter***

Hayter, Anthony J. (1984), A proof of the conjecture that the Tukey-Kramer multiple comparisons procedure is conservative, *Annals of Statistics*, **12**, 61–75.

***Hebb***

Hebb, D. O. (1949) *The Organization of Behaviour: A Neuropsychological Theory*, John Wiley.

***Heiberger***

Heiberger, Richard M. (1978), Generation of random orthogonal matrices, *Applied Statistics*, **27**, 199−206.

***Hemmerle.***

Hemmerle, William J. (1967), *Statistical Computations on a Digital Computer*, Blaisdell Publishing Company, Waltham, Mass.

### *Herraman*

Herraman, C. (1968), Sums of squares and products matrix, *Applied Statistics*, **17**, 289–292.

### *Hill*

Hill, G.W. (1970), Student's *t*-distribution, *Communications of the ACM*, **13**, 617–619.

Hill, G.W. (1970), Student's *t*-quantiles, *Communications of the ACM*, **13**, 619–620.

### *Hinkelmann, K and Kemthorne*

Hinkelmann, K and Kemthorne, O (1994) *Design and Analysis of Experiments – Vol 1*, John Wiley.

### *Hinkley*

Hinkley, David (1977), On quick choice of power transformation, *Applied Statistics*, **26**, 67–69.

### *Hoaglin and Welsch*

Hoaglin, David C. and Roy E. Welsch (1978), The hat matrix in regression and ANOVA, *The American Statistician*, **32**, 17–22.

### *Hocking*

Hocking, R.R. (1972), Criteria for selection of a subset regression: Which one should be used?, *Technometrics*, **14**, 967–970.

Hocking, R.R. (1973), A discussion of the two-way mixed model, *The American Statistician*, **27**, 148−152.

Hocking, R.R. (1985), *The Analysis of Linear Models*, Brooks/Cole Publishing Company, Monterey, California.

***Hopfield***

Hopfield, J. J. (1987) Learning Algorithms and Probability Distributions in Feed-Forward and Feed-Back Networks, *Proceedings of the National Academy of Sciences*, 84, 8429-8433.

***Holland***

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

***Huber***

Huber, Peter J. (1981), *Robust Statistics*, John Wiley & Sons, New York.

***Hutchinson***

Hutchinson, J. M. (1994) *A Radial Basis Function Approach to Financial Timer Series Analysis*, Ph.D. dissertation, Massachusetts Institute of Technology.

***Hughes and Saw***

Hughes, David T., and John G. Saw (1972), Approximating the percentage points of Hotelling's generalized $T_0^2$ statistic, *Biometrika*, **59**, 224–226.

***Hwang***

Hwang, J. T. G. and Ding, A. A. (1997) Prediction Intervals for Artificial Neural Networks, *Journal of the American Statistical Society*, 92(438) 748-757.

***Iman and Davenport***

Iman, R.L., and J.M. Davenport (1980), Approximations of the critical region of the Friedman statistic, *Communications in Statistics*, **A9(6)**,
571–595.

### *Jacobs*

Jacobs, R. A., Jorday, M. I., Nowlan, S. J., and Hinton, G. E. (1991) Adaptive Mixtures of Local Experts, *Neural Computation*, 3(1), 79-87.

### *Jennrich and Robinson*

Jennrich, R.I. and S.M. Robinson (1969), A Newton-Raphson algorithm for maximum likelihood factor analysis, *Psychometrika*, **34**, 111–123.

### *Jennrich and Sampson*

Jennrich, R.I. and P.F. Sampson (1966), Rotation for simple loadings, *Psychometrika*, **31**, 313–323.

### *John*

John, Peter W.M. (1971), *Statistical Design and Analysis of Experiments*, Macmillan Company, New York.

### *Jöhnk*

Jöhnk, M.D. (1964), Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen, *Metrika*, **8**, 5–15.

### *Johnson and Kotz*

Johnson, Norman L., and Samuel Kotz (1969), *Discrete Distributions*, Houghton Mifflin Company, Boston.

Johnson, Norman L., and Samuel Kotz (1970a), *Continuous Univariate Distributions*-1, John Wiley & Sons, New York.

Johnson, Norman L., and Samuel Kotz (1970b), *Continuous Univariate Distributions*-2, John Wiley & Sons, New York.

### *Johnson and Kotz*

Johnson, N.L. and Kotz, S. (1972), *Distributions in Statistics: Continuous Multivariate Distributions*, John Wiley & Sons, Inc., New York.

### *Johnson and Welch*

Johnson, D.G., and W.J. Welch (1980), The generation of pseudo-random correlation matrices, *Journal of Statistical Computation and Simulation*, **11**,
55−69.

### *Jonckheere*

Jonckheere, A.R. (1954), A distribution-free *k*-sample test against ordered alternatives, *Biometrika*, **41**, 133–143.

### *Jöreskog*

Jöreskog, K.G. (1977), Factor analysis by least squares and maximum-likelihood methods, *Statistical Methods for Digital Computers*, (edited by Kurt Enslein, Anthony Ralston, and Herbert S. Wilf), John Wiley & Sons, New York,
125–153.

### *Kachitvichyanukul*

Kachitvichyanukul, Voratas (1982), *Computer generation of Poisson, binomial, and hypergeometric random variates*, Ph.D. dissertation, Purdue University, West Lafayette, Indiana.

### *Kaiser*

Kaiser, H.F. (1963), Image analysis, *Problems in Measuring Change*, (edited by C. Harris), University of Wisconsin Press, Madison, Wis.

### *Kaiser and Caffrey*

Kaiser, H.F. and J. Caffrey (1965), Alpha factor analysis, *Psychometrika*, **30**,
1–14.

### *Kalbfleisch and Prentice*

Kalbfleisch, John D., and Ross L. Prentice (1980), *The Statistical Analysis of Failure Time Data*, John Wiley & Sons, New York.

### *Keast*

Keast, P. (1973) Optimal Parameters for Multidimensional Integration, *SIAM Journal of Numerical Analysis*, **10**, 831-838.

### *Kemp*

Kemp, A.W., (1981), Efficient generation of logarithmically distributed pseudo-random variables, *Applied Statistics,* **30**, 249–253.

### *Kendall and Stuart*

Kendall, Maurice G. and Alan Stuart (1973), *The Advanced Theory of Statistics*, Volume 2: *Inference and Relationship*, 3d ed., Charles Griffin & Company, London.

Kendall, Maurice G. and Alan Stuart (1979), *The Advanced Theory of Statistics*, Volume 2: *Inference and Relationship*, 4th ed., Oxford University Press, New York.

### *Kendall et al.*

Kendall, Maurice G., Alan Stuart, and J. Keith Ord (1983), *The Advanced Theory of Statistics*, Volume 3: *Design and Analysis, and Time Series*, 4th. ed., Oxford University Press, New York.

### *Kennedy and Gentle*

Kennedy, William J., Jr. and James E. Gentle (1980), *Statistical Computing*, Marcel Dekker, New York.

### *Kohonen*

Kohonen, T. (1995), Self-Organizing Maps, Third Edition. Springer Series in Information Sciences., New York.

### *Kuehl, R. O.*

Kuehl, R. O. (2000) *Design of Experiments: Statistical Principles of Research Design and Analysis*, 2$^{nd}$ edition, Duxbury Press.

### *Kim and Jennrich*

Kim, P.J., and R.I. Jennrich (1973), Tables of the exact sampling distribution of the two sample Kolmogorov-Smirnov criterion $D_{mn}$ ($m < n$), in *Selected Tables in Mathematical Statistics*, Volume 1, (edited by H. L. Harter and D.B. Owen), American Mathematical Society, Providence, Rhode Island.

### *Kinderman and Ramage*

Kinderman, A.J., and J.G. Ramage (1976), Computer generation of normal random variables, *Journal of the American Statistical Association*, **71**, 893−896.

### *Kinderman et al.*

Kinderman, A.J., J.F. Monahan, and J.G. Ramage (1977), Computer methods for sampling from Student's *t* distribution, *Mathematics of Computation* **31**,
1009−1018.

### *Kinnucan and Kuki*

Kinnucan, P. and H. Kuki (1968), *A Single Precision INVERSE Error Function Subroutine*, Computation Center, University of Chicago.

### *Kirk*

Kirk, Roger E. (1982), *Experimental Design: Procedures for the Behavioral Sciences*, 2d ed., Brooks/Cole Publishing Company, Monterey, Calif.

### *Kitagawa and Akaike*

Kitagawa, G. and Akaike, H., A Procedure for the modeling of non-stationary time series, Ann. Inst. Statist. Math. 30 (1978), Part B, 351-363.

### *Konishi and Kitagawa*

Konishi, S. and Kitagawa, G (2008), Information Criteria and Statistical Modeling, Springer, New York.

### *Knuth*

Knuth, Donald E. (1981), *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, 2d ed., Addison-Wesley, Reading, Mass.

### *Kshirsagar*

Kshirsagar, Anant M. (1972), *Multivariate Analysis*, Marcel Dekker, New York.

### *Lachenbruch*

Lachenbruch, Peter A. (1975), *Discriminant Analysis*, Hafner Press, London.

### *Lai*

Lai, D. (1998a), Local asymptotic normality for location-scale type processes. *Far East Journal of Theorectical Statistics*, (in press).

Lai, D. (1998b), Asymptotic distributions of the correlation integral based statistics. *Journal of Nonparametric Statistics*, (in press).

Lai, D. (1998c), Asymptotic distributions of the estimated BDS statistic and residual analysis of AR Models on the Canadian lynx data. *Journal of Biological Systems*, (in press).

### *Laird and Oliver*

Laird, N.M., and D. Fisher (1981), Covariance analysis of censored survival data using log-linear analysis techniques, *JASA* **76**, 1231−1240.

### *Lawless*

Lawless, J.F. (1982), *Statistical Models and Methods for Lifetime Data*, John Wiley & Sons, New York.

### *Lawley and Maxwell*

Lawley, D.N. and A.E. Maxwell (1971), *Factor Analysis as a Statistical Method*, 2d ed., Butterworth, London.

### *Lawrence et al*

Lawrence, S., Giles, C. L, Tsoi, A. C., Back, A. D.  (1997) Face Recognition:  A Convolutional Neural Network Approach,  *IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition*, 8(1), 98-113.

***Learmonth and Lewis***

Learmonth, G.P. and P.A.W. Lewis (1973), *Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A*, Naval Postgraduate School, Monterey, Calif.

***Lee***

Lee, Elisa T. (1980), *Statistical Methods for Survival Data Analysis*, Lifetime Learning Publications, Belmont, Calif.

***Lehmann***

Lehmann, E.L. (1975), *Nonparametrics: Statistical Methods Based on Ranks*, Holden-Day, San Francisco.

***Levenberg***

Levenberg, K. (1944), A method for the solution of certain problems in least squares, *Quarterly of Applied Mathematics*, **2**, 164–168.

***Levene, H.***

Levene, H. (1960) In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, I. Olkin et al. editors, Stanford University Press,  278-292.

***Lewis et al.***

Lewis, P.A.W., A.S. Goodman, and J.M. Miller (1969), A pseudorandom number generator for the System/360, *IBM Systems Journal*, **8**, 136–146.

***Li***

Li, L. K.  (1992) Approximation Theory and Recurrent Networks, Proc. Int. Joint Conf. On Neural Networks, vol. II, 266-271.

### *Liffiefors*

Lilliefors, H.W. (1967), On the Kolmogorov-Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association*, **62**, 534–544.

### *Lippmann*

Lippmann, R. P. (1989) Review of Neural Networks for Speech Recognition, Neural Computation, I, 1-38.

### *Ljung and Box*

Ljung, G.M., and G.E.P. Box (1978), On a measure of lack of fit in time series models, *Biometrika*, **65**, 297–303.

### *Loh*

Loh, W.-Y. and Shih, Y.-S. (1997) Split Selection Methods for Classification Trees, *Statistica Sinica*, 7, 815-840. For information on the latest version of QUEST see: http://www.stat.wisc.edu/~loh/quest.html.

### *Longley*

Longley, James W. (1967), An appraisal of least-squares programs for the electronic computer from the point of view of the user, *Journal of the American Statistical Association*, **62**, 819–841.

### *Matsumoto and Nishimure*

Makoto Matsumoto and Takuji Nishimura, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, Pages 3–30.

### *Mandic*

Mandic, D. P. and Chambers, J. A. (2001) *Recurrent Neural Networks for Prediction*, John Wiley & Sons, LTD.

### *Manning*

Manning, C. D. and Schütze, H. (1999) *Foundations of Statistical Natural Language Processing,* MIT Press.

### *Marsaglia*

Marsaglia, George (1964), Generating a variable from the tail of a normal distribution, *Technometrics*, **6**, 101−102.

Marsaglia, G. (1968), Random numbers fall mainly in the planes, *Proceedings of the National Academy of Sciences*, **61**, 25−28.

Marsaglia, G. (1972), The structure of linear congruential sequences, in *Applications of Number Theory to Numerical Analysis*, (edited by S. K. Zaremba), Academic Press, New York, 249−286.

Marsaglia, George (1972), Choosing a point from the surface of a sphere, *The Annals of Mathematical Statistics*, **43**, 645−646.

### *Marsaglia and Tsang*

Marsaglia, G. and Tsang, W. W. (2000), The Ziggurat Method for Generating Random Variables, *Journal of Statistical Software*, **5-8**, 1−7.

### *McCulloch*

McCulloch, W. S. and Pitts, W. (1943), A Logical Calculus for Ideas Imminent in Nervous Activity, *Bulletin of Mathematical Biophysics*, 5, 115-133.

### *McKean and Schrader*

McKean, Joseph W., and Ronald M. Schrader (1987), Least absolute errors analysis of variance, in *Statistical Data Analysis Based on the $L_1$-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 297−305.

### *McKeon*

McKeon, James J. (1974), *F* approximations to the distribution of Hotelling's $T_0^2$, *Biometrika*, **61**, 381–383.

***McCullagh and Nelder***

McCullagh, P., and J.A. Nelder, (1983), *Generalized Linear Models*, Chapman and Hall, London.

***Maindonald***

Maindonald, J.H. (1984), *Statistical Computation*, John Wiley & Sons, New York.

***Marazzi***

Marazzi, Alfio (1985), Robust affine invariant covariances in ROBETH, ROBETH-85 document No. 6, Division de Statistique et Informatique, Institut Universitaire de Medecine Sociale et Preventive, Laussanne.

***Mardia et al.***

Mardia, K.V. (1970), Measures of multivariate skewness and kurtosis with applications, *Biometrics*, **57**, 519–530.

Mardia, K.V., J.T. Kent, J.M. Bibby (1979), *Multivariate Analysis*, Academic Press, New York.

***Mardia and Foster***

Mardia, K.V. and K. Foster (1983), Omnibus tests of multinormality based on skewness and kurtosis, *Communications in Statistics A, Theory and Methods*, **12**, 207–221.

***Marquardt***

Marquardt, D. (1963), An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics*, **11**, 431–441.

***Marsaglia***

Marsaglia, George (1964), Generating a variable from the tail of a normal distribution, *Technometrics*, **6**, 101−102.

***Marsaglia and Bray***

Marsaglia, G. and T.A. Bray (1964), A convenient method for generating normal variables, *SIAM Review*, **6**, 260−264.

***Marsaglia et al.***

Marsaglia, G., M.D. MacLaren, and T.A. Bray (1964), A fast procedure for generating normal random variables, *Communications of the ACM*, **7**, 4−10.

***Merle and Spath***

Merle, G., and H. Spath (1974), Computational experiences with discrete $L_p$ approximation, *Computing*, **12**, 315−321.

***Miller***

Miller, Rupert G., Jr. (1980), *Simultaneous Statistical Inference*, 2d ed., Springer-Verlag, New York.

***Milliken and Johnson***

Milliken, George A. and Dallas E. Johnson (1984), *Analysis of Messy Data*, *Volume 1: Designed Experiments*, Van Nostrand Reinhold, New York.

***Mitchell***

Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, MIT Press.

***Moran***

Moran, P.A.P. (1947), Some theorems on time series I, *Biometrika*, **34**,
281–291.

***Moré et al.***

Moré, Jorge, Burton Garbow, and Kenneth Hillstrom (1980), *User Guide for [4] MINPACK-1*, Argonne National Laboratory Report ANL-80_74, Argonne, Ill.

***Morrison***

Morrison, Donald F. (1976), *Multivariate Statistical Methods*, 2nd. ed. McGraw-Hill Book Company, New York.

***Muller***

Muller, M.E. (1959), A note on a method for generating points uniformly on N-dimensional spheres, *Communications of the ACM*, **2**, 19−20.

***Nelson***

Nelson, D. B.  (1991), Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, , **59**, 347–370.

***Nelson***

Nelson, Peter (1989), Multiple Comparisons of Means Using Simultaneous Confidence Intervals, *Journal of Quality Technology*, **21**, 232–241.

***Neter***

Neter, John (1983), *Applied Linear Regression Models*, Richard D. Irwin, Homewood, Ill.

***Neter and Wasserman***

Neter, John and William Wasserman (1974), *Applied Linear Statistical Models*, Richard D. Irwin, Homewood, Ill.

***Noether***

Noether, G.E. (1956), Two sequential tests against trend, *Journal of the American Statistical Association*, **51**, 440–450.

***Owen***

Owen, D.B. (1962), *Handbook of Statistical Tables*, Addison-Wesley Publishing Company, Reading, Mass.

Owen, D.B. (1965), A Special Case of the Bivariate Non-central *t* Distribution, *Biometrika*, **52**, 437–446.

***Ozaki and Oda***

Ozaki, T and Oda H (1978) Nonlinear time series model identification by Akaike's information criterion. *Information and Systems, Dubuisson eds*, Pergamon Press. 83-91.

***Pao***

Pao, Y. (1989) *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing.

***Palm***

Palm, F. C. (1996), GARCH models of volatility. In *Handbook of Statistics*,
Vol. 14, 209-240. Eds: Maddala and Rao. Elsevier,New York.

***Parker***

Parker, D. B., (1985), *Learning Logic*. Technical Report TR-47, Cambridge, MA: MIT Center for Research in computational Economics and Management Science.

***Patefield***

Patefield, W.M. (1981), An efficient method of generating $R \times C$ tables with given row and column totals, *Applied Statistics*, **30**, 91−97.

***Patefield and Tandy***

Patefield, W.M. (1981), and Tandy D. (2000) Fast and Accurate Calculation of Owen's T-Function, *J. Statistical Software*, **5**, Issue 5.

***Peixoto***

Peixoto, Julio L. (1986), Testable hypotheses in singular fixed linear models, Communications in Statistics: *Theory and Methods*, **15**,
1957–1973.

### *Petro*

Petro, R. (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **13**, 624.

### *Pillai*

Pillai, K.C.S. (1985), Pillai's trace, in *Encyclopedia of Statistical Sciences*, *Volume 6*, (edited by Samuel Kotz and Norman L. Johnson), John Wiley & Sons, New York, 725–729.

### *Poli*

Poli, I. and Jones, R. D. (1994) A Neural Net Model for Prediction, *Journal of the American Statistical Society*, 89(425) 117-121.

### *Pregibon*

Pregibon, Daryl (1981), Logistic regression diagnostics, *The Annals of Statistics*, **9**, 705–724.

### *Prentice*

Prentice, Ross L. (1976), A generalization of the probit and logit methods for dose response curves, *Biometrics*, **32**, 761–768.

### *Priestley*

Priestley, M.B. (1981), *Spectral Analysis and Time Series*, Volumes 1 and 2, Academic Press, New York.

### *Quinlan*

Quinlan, J. R. (1993). C4.5 Programs for Machine Learning, Morgan Kaufmann. For the latest information on Quinlan's algorithms see http://www.rulequest.com/.

Quinlan (1987). *Simplifying Decision Trees.* Int J Man-Machine Studies 27, pp. 221-234.

### *Rao*

Rao, C. Radhakrishna (1973), *Linear Statistical Inference and Its Applications*, 2d ed., John Wiley & Sons, New York.

### *Reed*

Reed, R. D. and Marks, R. J. II (1999) *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, The MIT Press, Cambridge, MA.

### *Ripley*

Ripley, B. D. (1994) Neural Networks and Related Methods for Classification, *Journal of the Royal Statistical Society B*, 56(3), 409-456.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*, Cambridge University Press.

### *Robinson*

Robinson, Enders A. (1967), *Multichannel Time Series Analysis with Digital Computer Programs*, Holden-Day, San Francisco.

### *Rosenblatt*

Rosenblatt, F. (1958) The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychol. Rev.*, 65, 386-408.

### *Royston*

Royston, J.P. (1982a), An extension of Shapiro and Wilk's *W* test for normality to large samples, *Applied Statistics*, **31**, 115–124.

Royston, J.P. (1982b), The *W* test for normality, *Applied Statistics*, **31**, 176–180.

Royston, J.P. (1982c), Expected normal order statistics (exact and approximate), *Applied Statistics*, **31**, 161–165.

Royston, J. P. (1991), Approximating the Shapiro-Wilk W-test for non-normality, *Statistics and Computing*, **2**, 117-119.

### *Rumelhart*

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986) Learning Representations by Back-Propagating Errors, *Nature*, 323, 533-536.

Rumelhart, D. E. and McClelland, J. L. eds. (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1, 318-362, MIT Press.

***Sallas***

Sallas, William M. (1990), An algorithm for an $L_p$ norm fit of a multiple linear regression model, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 131−136.

***Sallas and Lionti***

Sallas, William M. and Abby M. Lionti (1988), *Some useful computing formulas for the nonfull rank linear model with linear equality restrictions*, IMSL Technical Report 8805, IMSL, Houston.

***Savage***

Savage, I. Richard (1956), Contributions to the theory of rank order statistics-the two-sample case, *Annals of Mathematical Statistics*, **27**, 590–615.

***Scheffe***

Scheffe, Henry (1959), *The Analysis of Variance*, John Wiley & Sons, New York.

***Schmeiser***

Schmeiser, Bruce (1983), Recent advances in generating observations from discrete random variates, *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, (edited by James E. Gentle), North-Holland Publishing Company, Amsterdam, 154–160.

***Schmeiser and Babu***

Schmeiser, Bruce W. and A.J.G. Babu (1980), Beta variate generation via exponential majorizing functions, *Operations Research*, **28**, 917–926.

***Schmeiser and Kachitvichyanukul***

Schmeiser, Bruce and Voratas Kachitvichyanukul (1981), *Poisson Random Variate Generation*, Research Memorandum 81–4, School of Industrial Engineering, Purdue University, West Lafayette, Ind.

### *Schmeiser and Lal*

Schmeiser, Bruce W. and Ram Lal (1980), Squeeze methods for generating gamma variates, *Journal of the American Statistical Association*, **75**, 679–682.

### *Searle*

Searle, S.R. (1971), *Linear Models*, John Wiley & Sons, New York.

### *Seber*

Seber, G.A.F. (1984), *Multivariate Observations*, John Wiley & Sons, New York.

### *Snedecor and Cochran*

Snedecor and Cochran (1967) *Statistical Methods*, 6th edition, Iowa State University Press.

### *Snedecor and Cochran*

Snedecor, George W. and Cochran, William G. (1967) *Statistical Methods*, 6th edition, Iowa State University Press, 296-298.

Snedecor, George W. and Cochran, William G. (1967) *Statistical Methods*, 6th edition, Iowa State University Press, 432—436.

### *Shampine*

Shampine, L.F. (1975), Discrete least-squares polynomial fits, *Communications of the ACM*, **18**, 179–180.

### *Siegal*

Siegal, Sidney (1956), *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York.

### *Singleton*

Singleton, R.C. (1969), Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **12**, 185–187.

***Smirnov***

Smirnov, N.V. (1939), Estimate of deviation between empirical distribution functions in two independent samples (in Russian), *Bulletin of Moscow University*, **2**, 3–16.

***Smith and Dubey***

Smith, H., and S. D. Dubey (1964), "Some reliability problems in the chemical industry", Industrial Quality Control, 21 (2), 1964, 64-70.

***Smith***

Smith, M. (1993) *Neural Networks for Statistical Modeling*, New York: Van Nostrand Reinhold.

***Snedecor and Cochran***

Snedecor, George W. and William G. Cochran (1967), *Statistical Methods*, 6th ed., Iowa State University Press, Ames, Iowa.

***Sposito***

Sposito, Vincent A. (1989), Some properties of $L_p$-estimators, in *Robust Regression*: *Analysis and Applications* (edited by Kenneth D. Lawrence and Jeffrey L. Arthur), Marcel Dekker, New York, 23−58.

***Spurrier and Isham***

Spurrier, John D. and Steven P. Isham (1985), Exact simultaneous confidence intervals for pairwise comparisons of three normal means, *Journal of the American Statistical Association*, **80**, 438–442.

***Stablein, Carter, and Novak***

Stablein, D.M, W.H. Carter, and J.W. Novak (1981), Analysis of survival data with nonproportional hazard functions, *Controlled Clinical Trials*, **2**, 149–159.

***Stahel***

Stahel, W. (1981), Robuste Schatzugen: Infinitesimale Opimalitat und Schatzugen von Kovarianzmatrizen, Dissertation no. 6881, ETH, Zurich.

***Steel and Torrie***

Steel and Torrie (1960) *Principles and Procedures of Statistics*, McGraw-Hill.

***Stephens***

Stephens, M.A. (1974), EDF statistics for goodness of fit and some comparisons, *Journal of the American Statistical Association*, **69**, 730–737.

***Stirling***

Stirling, W.D. (1981), Least squares subject to linear constraints, *Applied Statistics*, **30**, 204–212. (See correction, p. 357.)

***Stoline***

Stoline, Michael R. (1981), The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way ANOVA designs,
*The American Statistician*, **35**, 134–141.

***Strecok***

Strecok, Anthony J. (1968), On the calculation of the inverse of the error function, *Mathematics of Computation*, **22**, 144–158.

***Studenmund***

Studenmund, A. H. (1992) Using Economics:  A Practical Guide, New York:  Harper Collins.

***Swingler***

Swingler, K. (1996) Applying Neural Networks:  A Practical Guide, Academic Press.

***Tanner and Wong***

Tanner, Martin A., and Wing H. Wong (1983), The estimation of the hazard function from randomly censored data by the kernel method, *Annals of Statistics*, **11**, 989–993.

Tanner, Martin A., and Wing H. Wong (1984), Data-based nonparametric estimation of the hazard function with applications to model diagnostics and exploratory analysis, *Journal of the American Statistical Association*, **79**, 123–456.

### *Taylor and Thompson*

Taylor, Malcolm S., and James R. Thompson (1986), Data based random number generation for a multivariate distribution via stochastic simulation, *Computational Statistics & Data Analysis*, **4**, 93−101.

### *Tesauro*

Tesauro, G. (1990)  Neurogammon Wins Computer Olympiad, *Neural Computation*, 1, 321-323.

### *Tezuka*

Tezuka, S. (1995), *Uniform Random Numbers: Theory and Practice*. Academic Publishers, Boston.

### *Thompson*

Thompson, James R, (1989), *Empirical Model Building*, John Wiley & Sons, New York.

### *Tong*

Tong, Y. L. (1990), *The Multivariate Normal Distribution*, Springer-Verlag, New York.

### *Tucker and Lewis*

Tucker, Ledyard and Charles Lewis (1973), A reliability coefficient for maximum likelihood factor analysis, *Psychometrika*, **38**, 1–10.

### *Tukey*

Tukey, John W. (1962), The future of data analysis, *Annals of Mathematical Statistics,* **33**, 1–67.

### *Velleman and Hoaglin*

Velleman, Paul F. and David C. Hoaglin (1981), *Applications, Basics, and Computing of Exploratory Data Analysis*, Duxbury Press, Boston.

***Verdooren***

Verdooren, L. R. (1963), Extended tables of critical values for Wilcoxon's test statistic, *Biometrika*, **50**, 177–186.

***Wallace***

Wallace, D.L. (1959), Simplified Beta-approximations to the Kruskal-Wallis H-test, *Journal of the American Statistical Association*, **54**, 225–230.

***Warner***

Warner, B. and Misra, M. (1996) Understanding Neural Networks as Statistical Tools, *The American Statistician*, 50(4) 284-293.

***Weisberg***

Weisberg, S. (1985), *Applied Linear Regression*, 2d ed., John Wiley & Sons, New York.

***Werbos***

Werbos, P. (1974) *Beyond Regression:  New Tools for Prediction and Analysis in the Behavioral Science*, PhD thesis, Harvard University, Cambridge, MA.

Werbos, P. (1990) Backpropagation Through Time:  What It Does and How to do It, *Proc. IEEE*, 78, 1550-1560.

***Wetzel***

Wetzel, A. (1983), *Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization*, Unpublished manuscript, Univ. of Pittsburg, Pittsburg.

***Williams***

Williams, R. J. and Zipser, D. (1989) A Learning Algorithm for Continuously Running Fully Recurrent Neural Networks, *Neural Computation,* 1, 270-280.

***Witten***

Witten, I. H. and Frank, E. (2000) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers.

***Woodfield***

Woodfield, Terry J. (1990), Some notes on the Ljung-Box portmanteau statistic, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 155–160.

***Wu***

Wu, S-I (1995) Mirroring Our Thought Processes, *IEEE Potentials*, 14, 36-41.

***Yates, F.***

Yates, F. (1936) A new method of arranging variety trials involving a large number of varieties. *Journal of Agricultural Science*, **26**, 424-455.