

**A STUDY OF THE EFFICIENCY OF THE
FOREIGN EXCHANGE MARKET THROUGH
ANALYSIS OF ULTRA-HIGH FREQUENCY DATA**

**Ludwig Kanzler
Christ Church
University of Oxford**

**D.Phil. Thesis
Sub-Faculty of Economics
Michaelmas Term 1998**

***Appendix C:
Software (MATLAB Code)***

Also downloadable in ASCII format from:

<http://users.ox.ac.uk/~econlrk>

See the first pages of this volume for a table of contents.

Programme 3.1: Standing-Quote Aggregation of Unevenly Spaced Ticks

```

function [aggticks, spacing] = aggreg (series, interval, span, tickmin)

%AGGREG Standing-quote aggregation of unevenly spaced ticks into regular x-min. spacing
%
% [AGGTICKS, SPACING] = AGGREG (SERIES, INTERVAL, SPAN, TICKMIN)
% * aggregates vector SERIES of unevenly spaced ticks in MATLAB DATENUM format
% * into evenly spaced periods of length INTERVAL in minutes.
%
% * AGGTICKS is a vector of indices corresponding to those ticks of SERIES onto which
% the SERIES was aggregated;
% * SPACING is a vector of evenly spaced ticks in DATENUM format corresponding to the
% aggregated values.
%
% * SPAN is a two-element vector defining the starting point (default = SERIES(1)) and
% the closing point (default=SERIES(END)) of the aggregated series in DATENUM format.
% * TICKMIN defines the starting point for the aggregation within SERIES, e.g. assuming
% the first tick in SERIES to be DATENUM('01-Oct-1992 00:00:14') and INTERVAL to be 5
% (default), then for TICKMIN=1, SPACING starts with DATENUM('01-Oct-1992 00:01:00'),
% for 2 -> DATENUM('01-Oct-1992 00:02:00'), etc.
% TICKMIN must be between 0 and INTERVAL.
%
% By varying TICKMIN, different aggregated samples can be drawn from the same data set,
% thus allowing maximum use of the information contained in the data.
%
% This function can be used not only for the purpose of deriving an aggregated time
% series, but also to compute the frequency of observations in any interval by
% differencing AGGTICKS (see also the author's OBSFREQ.M file).
%
% Another distinguishing feature of this algorithm is that its main part is free of loops
% and hence very fast even on large data sets. A large number of gaps in the data (i.e.
% intervals in which the quote does not change from the previous one) will prolong the
% runtime significantly.
%
% EXAMPLES on a PII-233 192MB 60ns RAM Windows NT 4.0 system with O&A's HFDF-I dataset:
%
% Aggregation      1,472,241 DM/$ quotes      567,759 ¥/$ quotes
% 60 min.          46 sec.s                19 sec.s
% 30 min.          52 sec.s                22 sec.s
% 10 min.          77 sec.s                44 sec.s
% 5 min.           2.5 min.s               2.0 min.s
% 2 min.           10.7 min.s              9.8 min.s
% 1 min.           36.9 min.s              38.4 min.s
% and              115 MB RAM              65 MB RAM
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
% Copyright (c) 11 April 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.22 $ $ Date: 3 October 1998 $

% Check function arguments and assign default values if necessary:

if nargin < 3
    span = [series(1), series(end)];
    if nargin < 2
        interval = 5;
    end
end
if ~exist('tickmin', 'var')
    tickmin = interval;
end
if (tickmin > interval | tickmin <= 0)
    disp('Error! Inadmissible value assigned to TICKMIN!')
    return
end

```

```

% Generate the series of evenly spaced ticks:

spacing = datenum([datestr(span(1),1), ' ', datestr(span(1)+datenum('00:00:29'),15)]) +...
          tickmin/1440 : interval/1440 : span(2);

% Cat the evenly spaced series to the unevenly-spaced series and sort the total series;
% BTS (BothToSorted) is a vector of indices such that SORTED = BOTH (BTS), and
% STB (SortedToBoth) is a vector of indices such that BOTH = SORTED(STB).
% (If the same value occurs twice in BOTH (i.e. an original tick occurs on even tick
% time), these values are left in the same order by MATLAB.)

both      = [series(:)' spacing];
[sorted, bts] = sort(both);
stb(bts)   = 1:length(bts);

% So, STB(length(series)+1:end) are the indices of all the evenly spaced observations in
% BOTH, so all the index numbers preceding each of these index numbers point to the
% observations in BOTH which are the ticks immediately preceding or falling onto evenly-
% spaced ticks. Therefore, the corresponding indices in BOTH are given by

aggticks = bts (stb(length(series)+1 : end) - 1);

% If there is at least one unique tick for each interval, the above AGGTICKS is complete.
% But if there are "holes" in the data, i.e. if there are instances of ticks which are not
% replaced for subsequent intervals, then some of the indices given by AGGTICKS will point
% to the evenly spaced series SPACING in BOTH, so they need to be replaced by the tick
% indices preceding them until no such instances are left:

repeated = find(aggticks > length(series(:)));
while repeated
    aggticks(repeated) = aggticks(repeated-1);
    repeated          = find(aggticks > length(series(:)));
end

% End of file.

```

Programme 4.1: Variance-Ratio Test with Overlapping Observations and Heteroskedasticity Consistency

```
function [vr, psi, psisig] = varratio (P, Q)

%VARRATIO Heteroskedasticity-consistent variance ratio using overlapping observations
%
% [VR, PSI, PSISIG] = VARRATIO (P, Q) performs heteroskedasticity-consistent variance-
% ratio tests using overlapping observations for each spacing q defined in Q on time
% series P, and returns the corresponding result vectors
%
% VR      = the variance ratios
% PSI     = the test statistics, and
% PSISIG  = the significance levels at which each H0: VR = 1 (indicating a random walk)
%           is rejected against H1: VR <> 1.
%           (Assumes NaN if the MATLAB Statistics Toolbox is not installed).
%
% P must be strictly positive and in levels, and
% Q must be a vector of integers (in any order) greater than 1 (default is 2).
% (See also the NOTE at the bottom of the script.)
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
%           Copyright (c) 23 March 1998 by Ludwig Kanzler
%           Department of Economics, University of Oxford
%           Postal: Christ Church, Oxford OX1 1DP, U.K.
%           E-mail: ludwig.kanzler@economics.oxford.ac.uk
%           Homepage:      http://users.ox.ac.uk/~econlrk
%           $ Revision: 1.11 $$ Date: 27 September 1998 $

% Check input arguments/assign default:

if sum(P <= 0)
    error('All elements of the time series must be strictly positive.')
end

if nargin < 2
    Q = 2;
elseif Q~= round(Q) | sum(Q < 2)
    error('Spacing can only be specified by integers greater than 1.')
end

% The idea of the variance-ratio test is to check whether the variance of random-walk
% increments can be described as a linear function of the time interval (the null
% hypothesis). These increments are simply defined as continuously compounded "returns"
% on the price series (i.e. a time series in levels):

p   = log(P(:));
r   = p(2:end) - p(1:end-1);
obs = length(p);

% Compute the variance-ratio statistics for each q contained in Q:

i = 0;
for q = Q
    i = i + 1;

    % Define n such that (the total number of price) obs. = nq + 1
    % (so n defines the number of non-overlapping q-spaced returns):

    n = fix((obs-1)/q);

    % The unbiased sample mean (also maximum likelihood) estimator of r is given by:

    mu = (p(n*q+1) - p(1)) / (n*q);

    % The unbiased sample variance (also maximum likelihood) estimator of r is defined as:

    ssa = r(1:n*q)' * r(1:n*q);
    sa  = ssa / (n*q-1);
end
end
```

```

% The alternative unbiased estimator of the sample variance of r using the sum of
% q-period over-lapping returns is computed as follows:

dev = p(q+1:n*q+1) - p(1:n*q-q+1) - q*mu*ones(n*q-q+1,1);
sc = dev' * dev / (q*(n*q-q+1)*(1-1/n));

% And one thus obtains the variance ratio:

vr(i) = sc / sa;

% The heteroskedasticity-consistent estimator of the variance of VR is th/(n*q):

th = 0;
for k = 1 : q-1
    th = th + 4*(1-k/q)^2 * n*q / ssa^2 * ...
        ((r(k+1:n*q) - mu*ones(n*q-k, 1)).^2)' * (r(1:n*q-k) - mu*ones(n*q-k, 1)).^2;
end

% Then, the standardised test statistic is given by:

psi(i) = sqrt(n*q) * (vr(i) - 1) / sqrt(th);
end

% Since the test statistic follows a N(0,1) distribution, its level of significance is
% easily evaluated:
if exist('normcdf.m','file') & nargout == 3
    psisig = min(normcdf(psi,0,1), 1-normcdf(psi,0,1))*2;
elseif nargout == 3
    psisig(1:i) = NaN;
end

% End of function.

% NOTE:
%
% This script's procedure uses the theoretical exposition of Campbell, Lo & MacKinlay,
% 1997, pp. 48-55, which unsurprisingly follows Lo & MacKinlay (1988, 1989). The code
% adopts their notation as far as possible, but as the lowest index number in MATLAB is,
% of course, 1 (not 0), all indices are shifted up by 1. Also, while Campbell et al.
% (1997) arbitrarily restrict possible values of q to multiples of the smallest value 2
% (presumably for computational convenience), q can take the value of any integer greater
% than 1 in this implementation. Of course, this requires not only sc and psi to be
% functions of q, but also mu and sa (so their computation is part of the loop).

% REFERENCES:
%
% Campbell, John, Andrew Lo & Craig MacKinlay (1997), "The Econometrics of Financial
% Markets", Princeton University Press, Princeton, New Jersey
%
% Lo, Andrew & Craig MacKinlay (1988), "Stock Market Prices Do Not Follow Random Walks:
% Evidence from a Simple Specification Test", Review of Financial Studies, vol. 1,
% no. 1, pp. 41-66
%
% Lo, Andrew & Craig MacKinlay (1989), "The Size and Power of the Variance Ratio Test in
% Finite Samples: A Monte Carlo Investigation", Journal of Econometrics, vol. 40,
% pp. 203-238

% End of file.

```

Programme 4.2: Box-Pierce Q Test with Ljung-Box Finite-Sample Correction

```

function [q, qsig] = qstat (series, m)

%QSTAT Box-Pierce (1970) Q test using Ljung & Box's (1978) finite-sample correction
%
% [Q, QSIG] = QSTAT (SERIES, M) returns (row vector) Q, the Ljung & Box corrected Q
% statistics of serial correlation in (vector) SERIES for each lag order specified by
% (vector) M, and (row vector) QSIG, the levels of significance at which the associated
% null hypotheses of no correlation are rejected.
%
% SERIES should be a vector, or else it is transformed into a vector column by column.
%
% M can be a scalar (default is 1) or a vector of integers in any order. For example,
% if M = 5, results are only returned for the null hypothesis of no 5th order serial
% correlation, but if M = [3 1 5], Q and QSIG will be three-point vectors carrying the
% results for the test at lag orders 3, 1 and 5 (in this order).
%
% QSIG assumes NaN values if the MATLAB Statistics Toolbox is not installed.
%
% The cost of computation depends on the LARGEST lag order alone as specified in M.
% See the source code comments for an explanation of how the test is conducted and the
% bottom of the script for a list of references.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.

%
% Copyright (c) 6 April 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlkr
% $ Revision: 1.21 $$ Date: 15 September 1998 $

% Some preparations:

if nargin < 2, m = 1; end
m = m(:)';
maxm = max(m);
P(1:maxm) = 0;
series = series(:);
n = length(series);

% First, compute the parts of the Q statistic which differ for each j:

for j = 1 : maxm
    P(j) = (series(1:end-j)'*series(j+1:end))^2 / (n-j)^3;
end

% Then compute the cumulative sum and multiply by the part of the Q statistic which does
% not depend on j, thus obtaining vector Q which comprises the Q statistic for each j up
% to the highest (last) value of M:

Q = cumsum(P) * n^3*(n+2)/(series'*series)^2;

% But the Q statistic does not need to be returned for every j, only for all M:

q = Q(m);

% Lastly, evaluate the null hypothesis of no serial correlation at lag order m by
% computing the level of significance at which H0 is to be rejected:

if exist('chi2cdf.m','file') & nargin == 2
    qsig = 1 - chi2cdf(q, m);
elseif nargin == 2
    qsig = NaN*m;
end

% End of function.

```

```
% NOTE:
%
% This script builds on the theoretical foundations described, for example, in Campbell
% et al., 1997, p. 47, Harvey, 1990, pp. 212-213, and Intriligator et al., 1996, p. 204.
%
% However, in order to save CPU time by computing Q and QSIG for many m in one integrated
% algorithm, this implementation of the Q test is structured quite differently from any of
% the explicit equations contained in the above references. Specifically, all constant
% multiplicative terms have been pulled out of the summation (loop) and are applied only
% at the end; moreover, the summation is done only once.
%
% One criticism levelled against the Q test is that it is not clear which lag order to
% choose for the computation. This implementation addresses the problem by allowing
% computation of the test for ANY lag order without compromising speed, thus enabling the
% researcher to investigate the results for all possible m.

% REFERENCES:
%
% Box, G.E.P & David Pierce (1970), "Distribution of Residual Autocorrelations in
% Autoregressive-Integrated Moving Average Times Series Models", Journal of the
% American Statistical Association, vol. 65, no. 332 (December), pp. 1509-1526
%
% Campbell, John, Andrew Lo & Craig MacKinlay (1997), "The Econometrics of Financial
% Markets", Princeton University Press, Princeton, New Jersey
%
% Harvey, Andrew (1990), "The Econometric Analysis of Time Series", 2nd edition, MIT
% Press, Cambridge, Massachusetts
%
% Intriligator, Michael, Ronald Bodkin & Cheng Hsiao (1996), "Econometric Models,
% Techniques, and Applications", 2nd edition, Prentice Hall, Upper Saddle River, New
% Jersey
%
% Ljung, G.M. & G.E.P. Box (1978), "On a Measure of Lack of Fit in Time Series Models",
% Biometrika, vol. 65, no. 2, pp. 297-303

% End of file.
```

Programme 4.3: Geweke-Porter-Hudak Test of Long-Range Dependence

```

function [d, nobs, tasy, sigasy, tols, sigols] = gph (series, incl, excl)

%GPH Geweke & Porter-Hudak (1983) estimation of the fractional differencing parameter d
%
% [D, NOBS, TASY, SIGASY, TOLS, SIGOLS] = GPH (SERIES, INCL, EXCL) returns
% - GPH frequency domain estimator D of (differenced) time series in vector SERIES,
% - the number of observations NOBS used in the frequency domain regression,
% - asymptotic t-ratio TASY and associated level of significance SIGASY (when the known
% theoretical variance of the residuals (pi^2/6) is imposed on the calculation), and
% - OLS t-ratio TOLS and corresponding significance level SIGOLS (when the standard
% errors are estimated from standard OLS regression output).
%
% SIGASY and SIGOLS result from evaluating the null hypothesis H0: d = 0 against the
% two-sided alternative H1: d ~= 0. If TCDF.M of the MATLAB Statistics Toolbox is not
% on the path, SIGASY and SIGOLS assume NaN.
%
% Parameters INCL and EXCL are used to select the relevant range of frequencies:
% - the GPH regression excludes the first N^EXCL-1 frequencies
%   (default is EXCL = 0, so all lower frequencies are actually included), and
% - it includes the remaining lower frequencies up to the N^INCL th frequency
%   (default is INCL = 0.5, so the first SQRT(N) frequencies are included).
% Either INCL or EXCL, but not both, can be vector input, in which case the estimation
% results returned are all vectors corresponding to the varying values of INCL/EXCL.
%
% The objective of the GPH estimator is to capture any fractal structure in the LOWER
% frequencies, i.e. a correlation structure that is neither I(0) nor I(1) but I(d)
% where d is the fractional differencing parameter to be estimated here. These LOWER
% frequencies are covered EXCLUSIVELY (i.e. no specification of a short-memory process
% is required), because only a fraction of the first frequencies is used - hence the
% INCL restriction on the number of observations in the regression. Including too many
% frequencies would blur the analysis through higher-frequency cycles, while including
% too few frequencies would make the OLS regression less reliable. 0.5 is the standard
% value suggested by Kunsch (1986) and usually used in the literature. As argued by
% Kunsch (1986), also frequencies around the origin should be excluded to get a
% consistent estimator of D; yet, this is not always done in practice. Note also that
% if a seasonally-adjusted time series is used, another adjustment would be required to
% eliminate seasonal frequencies before running the GPH regression (see Ooms & Hassler,
% 1997). See also the bottom of the script for a literature list.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
% Copyright (c) 10 March 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.31 $$ Date: 15 September 1998 $

% Check input arguments and assign default values, if necessary:

if nargin < 1
    error('The GPH regression requires a time series input.')
elseif prod(size(series)) == 1
    error('The GPH regression cannot be run on a scalar.')
end

if nargin < 2
    incl = 0.5;
elseif isempty(incl)
    incl = 0.5;
end

if nargin < 3
    excl = 0;
elseif isempty(excl)
    excl = 0;
end

```

```

if incl >= 1
    error('The regression can only be run on fewer frequencies than there are observations.')
end

if excl < 0
    error('Input argument EXCL cannot be negative.')
end

lincl = length(incl);
lexcl = length(excl);

if lincl > 1 & lexcl > 1
    error('Only one of input arguments INCL and EXCL can be a vector, not both.')
elseif sum(excl >= incl)
    error('There are more frequencies excluded than there are included.')
end

incl(1:lincl*lexcl) = incl; % If one of EXCL and INCL is a vector, this transforms the
excl(1:lincl*lexcl) = excl; % other into a vector of equal values for each element.

% Take the Discrete Fourier Transform, thus obtaining the complex vector DFT (see also
% bottom of script):

dft = fft(series(:));
n = length(dft);
dft(1) = [];

% The following FOR loop runs the GPH regression for each parameter in EXCL or INCL:

for i = 1 : lincl*lexcl
    first(i) = ceil(n^excl(i)); % first frequency (lowest) of the regression series
    last(i) = floor(n^incl(i)); % last frequency (highest) of the regression series
    nobis(i) = last(i) - first(i) + 1; % number of observations of the regression series

    % The periodogram (a plot of power versus frequency) is defined by...

    % ... the Fourier frequencies (harmonic ordinates) of the sample, defined as (see GPH):

    freq = 2*pi*(first(i):last(i)) / n;

    % ... and the corresponding "power" observations of the periodogram, i.e. the squares
    % of the magnitude of DFT:

    power = abs(dft(first(i):last(i))).^2;

    % The spectral regression on a transformation of the periodogram (see GPH) produces the
    % the vector of slope coefficients BETA:

    y = log(power);
    X = [ones(nobis(i),1), log(4*sin(freq'/2).^2)];
    beta = X \ y;

    % A consistent estimate of fractional differencing parameter D is provided by the
    % negative slope coefficient on the frequency variable:

    d(i) = - beta(2);

    % The known theoretical variance of the residuals pi^2/6 is used to compute the ASYMPTO-
    % TIC standard errors; the OLS standard errors are computed from the actual residuals:

    seasy = sqrt(diag(inv(X'*X)) * pi^2/6 );
    tasy(i) = d(i)/seasy(2);

    if nargout > 4
        resid = y - X*beta;
        seols = sqrt(diag(inv(X'*X)) * resid'*resid/(nobis(i)-2));
        tols(i) = d(i)/seols(2);
    end
end
end

```

```

% If TCDF.M exists, evaluate H0 against the 2-sided alternative for all t-ratios at once:

if exist('tcdf.m','file')
    sigasy = min(1-tcdf(tasy, nobs-1), tcdf(tasy, nobs-1))*2;
    if nargout > 4
        sigols = min(1-tcdf(tols, nobs-1), tcdf(tols, nobs-1))*2;
    end
else
    sigasy(1:lincl*lexcl) = NaN;
    if nargout > 4
        sigols(1:lincl*lexcl) = NaN;
    end
end

% End of function.

% The following function is not actually used by the main function and only included here
% for the benefit of someone interested in translating this script into a programming
% language which does not have a built-in fast Fourier transform function. It may also be
% helpful to understand what the explicit Fourier transform equation computes and to check
% whether the built-in function FFT really produces the same results (it does! - to try
% simply replace FFT in the above code by SFT).

function dft = sft (series, n)

% SFT "Slow" Fourier Transform
%
% SFT (SERIES) is the discrete Fourier transform (DFT) of vector SERIES, computed using
% the explicit Fourier equation rather than using MATLAB's built-in function FFT.
%
% SFT (SERIES, N) returns the n-point DFT. If the length of SERIES is less than N,
% SERIES is padded with trailing zeros to length N. If the length of SERIES is greater
% than N, the sequence SERIES is truncated.
%
% Please note that unlike FFT, SFT only accepts vector input.
%
% Copyright (c) 1 April 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, England
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% $ Revision: 1.11 $ $ Date: 29 April 1998 $

N = length(series);
series = series(:);

if length(series) ~= N
    error('This Fourier transformation can only be performed on vector input.')
end

if nargin == 1
    n = N;
end

series(N+1:n) = 0;
dft(1:n,:) = 0;

% Computation for each element K in DFT (i.e. k from 1 to n):
for k = 1 : n
    dft(k) = exp(-i*2*pi*(k-1)*(0:n-1)/n) * series(1:n);
end

% NOTE: The matrix multiplication used above avoids another loop for summation; it is
% equivalent to the following routine, which is easier to understand, but much slower:
%
% for k = 1 : n
%     dftk = 0;
%     for l = 1 : n
%         dftk = dftk + series(l)*exp(-i*2*pi*(k-1)*(l-1)/n);
%     end
%     dft(k) = dftk;
% end

% End of sub-function.

```

% RELATED LITERATURE:

- %
% Baillie, Richard (1996), "Long Memory Processes and Fractional Integration in
% Econometrics", Journal of Econometrics, vol. 73, no. 1 (July), pp. 5-59
%
% Diebold, Francis & Marc Nerlove (1990), "Unit Roots in Economic Time Series: A
% Selective Survey", in George Rhodes Jr. & Thomas Fomby, eds., "Advances in
% Econometrics: A Research Annual", vol. 8 ("Co-integration, Spurious Regressions, and
% Unit Roots"), JAI Press, Greenwich, Connecticut, pp. 3-69
%
% Geweke, John & Susan Porter-Hudak (1983), "The Estimation and Application of Long Memory
% Time Series Models", Journal of Time Series Analysis, vol. 4, no. 4, pp. 221-238
%
% Granger, Clive & Roselyne Joyeux (1980), "An Introduction to Long-Memory Time Series
% Models and Fractional Differencing", Journal of Time Series Analysis, vol. 1, no. 1,
% pp. 15-29
%
% Harvey, Andrew (1993), "Time Series Models", 2nd edition, Harvester Wheatsheaf, Hemel
% Hempstead, England, Chapter 6 ("The Frequency Domain"), pp. 166-232
%
% Hosking, J.R.M. (1981), "Fractional Differencing", Biometrika, vol. 68, no. 1, pp.
% 165-176
%
% Kunsch, H. (1986), "Discrimination between Monotonic Trends and Long-Range Dependence",
% Journal of Applied Probability, vol. 23, pp. 1025-1030
%
% Mandelbrot, Benoît (1977), "Fractals: Form, Chance, and Dimension", Free Press,
% New York
%
% Ooms, Marius & Uwe Hassler (1997), "On the Effects of Seasonal Adjustment on the
% Periodogram Regression", Economics Letters, vol. 56, no. 2, pp. 135-141

% This MATLAB script draws on a procedure used (and described) in the following papers:

- %
% Barkoulas, John, Christopher Baum & Mustafa Caglayan (1998), "Persistent Dependence in
% Foreign Exchange Rates?", Journal of International Money and Finance, forthcoming,
% revised version of Boston College Working Papers in Economics, no. 377
%
% Barkoulas, John, Christopher Baum & Mustafa Caglayan (1998), "Long Memory or Level
% Shifts: Can Either Explain Nonstationary Real Exchange Rates Under the Current
% Float?", Boston College Working Papers in Economics, no. 380
%
% Barkoulas, John, Christopher Baum & Nickolaos Travlos (1998), "Long Memory in the Greek
% Stock Market", Journal of Applied Financial Economics, forthcoming, revised version
% of Boston College Working Papers in Economics, no. 356 (August 1997)
%
% Baum, Christopher & John Barkoulas (1996), "Testing for Fractal Structure in Stock
% Returns", Boston College Working Papers in Economics, no. 314 (April)
%
% Berg, Lennart & Johan Lyhagen (1996), "Short and Long Run Dependence in Swedish Stock
% Returns", Uppsala University, Department of Economics, working paper, no. 1996:19
% (September)

% ACKNOWLEDGEMENT:

% The author is grateful to Christopher Baum for a discussion of the GPH procedure.

% End of file.

Programme 4.4: Best-Fitting (Augmented) Dickey-Fuller Unit-Root Regression

```

function [adf, adfresid, df, dfresid] = unitroot (series)

%UNITROOT (Augmented) Dickey-Fuller and Phillips-Perron tests of the unit-root hypothesis
%
% [ADF, ADFRESID, DF, DFRESID] = UNITROOT (SERIES) tests the null hypothesis of the
%   existence of a unit root in SERIES and returns
%
%
%   - matrix ADF with the results for the Augmented Dickey-Fuller regression with the
%     highest number of augmented terms (dlags), if any, significant at the 5% level,
%
%   - vector ADFRESID with the residuals of the ADF regression,
%
%   - matrix DF with the results for the Dickey-Fuller regression, and
%
%   - vector DFRESID with the residuals of the DF regression.
%
% REQUIRES the MATLAB Statistics Toolbox (function files NORMCDF.M and TCDF.M) and
% the author's function files ADFREG.M, DFCRIT.M, DURBINH.M, DWATSON.M, and PHILLIPS.M.
%
%
% The general form of the regressions is:
%
% dseries = beta_0 + beta_1*series(-1) {+ beta_2*dseries(-1) + beta_3*dseries(-2) + ...}
%                                             + resid;
%
% The matrices returned are structured as follows:
%
%   {a}df = [ sigma      dw      beta_0  beta_1  {beta_2  ... }
%            tpp      dh      t_0     t_1     {t_2    ... }
%            tppsig   dhsig   NaN     tsig_1  {tsig_2  ... } ]
%
% where sigma = the estimated standard error of the residuals
%      dw = the Durbin-Watson statistics of the residuals
%      dh = the Durbin h statistic of the residuals
%      dhsig = the level of significance at which the (two-sided) null hypothesis
%              of no (first-order) autocorrelation in the residuals is rejected
%
% beta_0, beta_1 {,beta_2,...} = the estimated values of the coefficients (as above)
%
% t_0,    t_1    {,t_2,    ...} = the (uncorrected) t-ratios on the coefficients
% tpp                                = the Phillips-Perron corrected t-ratio on beta_1
%
%      tsig_1,
%      tppsig                                = the levels at which tsig_1 and tpp are statistically
%                                              significantly, using Dickey-Fuller critical values
%      {tsig_2, ... = the level at which t_2,... are statistically significant,
%                  using t-table critical values}
%
%
% So, reject a unit root,
%   - if t_1 in the ADF regression is statistically significant, e.g. tsig_1 <= 0.1,
%     AND the residuals are not correlated (otherwise the test statistic is inefficient),
%   - or if tpp (in any regression) is statistically significant (- or both).
% (Reject random walk, if unit root is rejected, or some dlags are significant, or both.)
%
% See also the NOTE section at the bottom of the script and all the files by the same
% author called by this script.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
% Copyright (c) 17 March 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.33 $$ Date: 10 September 1998 $

```

```

obs = length(series);

% First add dlags/augterms one by one as long as they are significant at the 5% level;
% as soon as the dlag added last is found not to be significant, stop the procedure and
% save the previous lag order (see ADFREG.M for details).

for augterms = 1 : obs-3
    [beta, se, t, tsig] = adfreg (series, augterms);
    if tsig(2+augterms) > 0.05
        augterms = augterms - 1;
        break
    end
end

% Then run and evaluate the Dickey-Fuller and Augmented Dickey Fuller regressions (this is
% done by the below sub-function):

[adf, adfresid] = evaluate (series, obs, augterms);
if nargout > 2 & augterms
    [df, dfresid] = evaluate (series, obs, 0);
elseif nargout > 2
    df = adf;
    dfresid = adfresid;
end

% End of main function.

function [results, resid] = evaluate (timeseries, nobs, dlags)

% THE (AUGMENTED) DICKEY-FULLER REGRESSION (see ADFREG.M for the details):

[beta, se, t, tsig, resid, rss, sigma] = adfreg (timeseries, dlags);

% Compute the DURBIN-WATSON d STATISTIC to check for first-order serial correlation in the
% residuals. Adding a sufficient number of dlags, as done in the above procedure, should
% produce "correlation-free" residuals so that the ADF regression estimators become
% unbiased; the Durbin-Watson statistic offers one means of checking this.
% NOTE that the DW statistic is biased towards 2.00 (indicating zero autocorrelation)
% when differenced lags are present, but can nevertheless be used to gauge the degree
% of autocorrelation (see DWATSON.M for details).

[dw, dwsigup, dwsiglow] = dwatson(resid);

% The Durbin-h statistic is robust to the inclusion of any lagged terms and is here
% evaluated in a two-sided test, i.e. against the alternative hypothesis of positive OR
% negative autocorrelation (see DWATSON.M for details):

[dh, dhsig] = durbinh (dw, se(2), nobs-dlags-1, 2);

% An alternative method to the above approach of finding the best-specification ADF
% regression and evaluating the unit-root hypothesis only on that regression is to follow
% Phillips (1987) and Phillips & Perron (1988) and compute the PHILLIPS-PERRON T-RATIO
% "corrected" for potential autocorrelation and/or heteroskedasticity (see PHILLIPS.M for
% details).

[tpp, tppsig] = phillips (se(2), t(2), resid, rss, sigma, nobs);

% COLLECT ALL RELEVANT STATISTICS in a matrix:

results = [ sigma    dw    beta'
            tpp     dh    t'
            tppsig  dhsig  tsig' ];

% End of sub-function.

% REFERENCES can be found in the reference sections of the author's function files called
% by this script.

% End of file.

```

Programme 4.5: (Augmented) Dickey-Fuller Unit-Root Regression

```

function [beta, se, t, tsig, resid, rss, sigma] = adfreg (series, dlags)

%ADFREG (Augmented) Dickey-Fuller regression
%
% [BETA, SE, T, TSIG, RESID, RSS, SIGMA] = ADFREG (SERIES, DLAGS) runs and evaluates a
% Dickey-Fuller (1979) type OLS regression on SERIES with the number of augmented terms
% (differenced lags) given by DLAGS (default is 0):
%
% dseries = beta_0 + beta_1*series(-1) [+ beta_2*dseries(-1) + beta_3*dseries(-2) + ...]
%
%
% returning: beta = the estimated regression coefficients [beta_0; beta_1; ...]
%
%           se = the estimated standard errors corresponding to beta
%
%           t = the t-ratios computed from beta and se
%
%           tsig = the level at which the corresponding t-ratios are statistically
%                 significantly, using Dickey-Fuller critical values for beta_1 and
%                 standard t-table critical values for beta_2, beta_3, etc.
%                 The significance level of the constant beta_0 is usually of little
%                 interest and since it follows neither a t-distribution nor a Dickey-
%                 Fuller distribution), it is not evaluated here and "NaN" is returned.
%
%           resid = the (vector) of residuals
%
%           rss = the residual sum of squares
%
%           sigma = the estimated standard error of the residuals
%
%
% (So, reject a unit root if t_1 in the (A)DF regression is statistically significant,
% e.g. tsig_1 <= 0.05, AND the residuals are not correlated (use, for example, the
% one of the Durbin or Box-Pierce statistics to check this), because otherwise the test
% is inefficient.)
%
%
% REQUIRES the MATLAB Statistics Toolbox and the author's DFCRIT m-function.
%
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
%           Copyright (c) 6 April 1998 by Ludwig Kanzler
%           Department of Economics, University of Oxford
%           Postal: Christ Church, Oxford OX1 1DP, U.K.
%           E-mail: ludwig.kanzler@economics.oxford.ac.uk
%           Homepage: http://users.ox.ac.uk/~econlkr
%           $ Revision: 1.01 $           $ Date: 20 May 1998 $

% Compute the total no. of observations in the sample & the series of first differences:
series = series(:);
obs = length(series);
dseries = series(2:obs) - series(1:obs-1);

% Create the matrices for the dependent and independent variables:
X = [series(dlags+1:obs-1)];

if dlags
    for i = 1 : dlags
        X = [X dseries(dlags+1-i:obs-1-i)];
    end
end

X = [ones(obs-dlags-1,1) X];
y = dseries(dlags+1:obs-1);

```

```
% "Run" the OLS REGRESSION, thus obtaining:

beta = X \ y; % vector of est. slope coeff.s (const., lag [dlags])
resid = y - X*beta ; % vector of residuals
rss = resid'*resid; % residual sum of squares
sgma2 = rss / (obs-2-dlags); % estimated variance of the residuals, sigma squared
sigma = sqrt(sgma2); % ... and sigma
se = sqrt(diag(sgma2*inv(X'*X))); % vector of est. standard errors of est. coeff.s
t = beta ./ se; % vector of corresponding t-ratios

% Evaluate the SIGNIFICANCE OF ALL T-RATIOS - SEE THE AUTHOR'S SEPARATE DFCRIT.M SCRIPT:

tsig(1:2,1) = [NaN; dfcrit(t(2), obs)];
if dlags > 0
    tsig(3:2+dlags) = min(1 - tcdf(t(3:end), obs-1-dlags), tcdf(t(3:end), obs-1-dlags))*2;
end

% End of function.

% REFERENCE:
%
% Dickey, David & Wayne Fuller (1979), "Distribution of the Estimators for Autoregressive
% Time Series With a Unit Root", Journal of the American Statistical Association, vol.
% 74, no. 366 (June), pp. 427-431

% End of file.
```

Programme 4.6: Phillips-Perron Test of the Unit-Root Hypothesis

```

function [tpp, tppsig] = phillips (se2, t2, resid, rss, sigma, obs)

%PHILLIPS Phillips-Perron test of the unit-root hypothesis in a Dickey-Fuller regression
%
% [TPP, TPPSIG] = PHILLIPS (SE2, T2, RESID, RSS, SIGMA, OBS)
%
% - computes the Phillips-Perron (Phillips, 1987, Phillips & Perron, 1988)
%   autocorrelation/heteroskedasticity corrected t-ratio TPP on the unit-root
%   coefficient in a Dickey-Fuller or an Augmented Dickey-Fuller (1979) regression, and
%
% - evaluates its significance level TPPSIG,
%
% using the following arguments from (A)DF the regression (which can be taken from the
% output of the author's ADFREG m-function):
%
% SE2   = the estimated standard errors of the unit-root coefficient
%
% T2    = the corresponding t-ratio
%
% RESID = the (row vector) of residuals
%
% RSS   = the residual sum of squares
%
% SIGMA = the estimated standard error of the residuals
%
% OBS   = the number of observations of the regression
%
% Usage: reject the null of a unit root, even in the presence of serial correlation
%        and/or heteroskedasticity, if tpp is statistically significant.
%
% REQUIRES the author's DFCRIT m-function to compute TPPSIG.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
%           Copyright (c) 6 April 1998 by Ludwig Kanzler
%           Department of Economics, University of Oxford
%           Postal: Christ Church, Oxford OX1 1DP, U.K.
%           E-mail: ludwig.kanzler@economics.oxford.ac.uk
%           Homepage:      http://users.ox.ac.uk/~econlrk
%           $ Revision: 1.03 $$ Date: 15 September 1998 $

% First determine the number of autocovariances of the residuals to be used, using the
% "rule of thumb" proposed by Schwert, 1987 (see also Diebold & Nerlove, 1990):

covs = fix(4*(obs/100)^0.25);

% Then compute the Newey & West (1987) estimator accordingly:

nw = rss;
for j = 1 : covs
    nw = nw + 2*(1-j/(covs+1))*(resid(1:end-j)'*resid(j+1:end));
end
nw = nw/obs;

% Finally, "apply" the NW estimator and some of the above regression output to the
% original t-ratio, thus obtaining the Phillips-Perron statistic, which follows the
% tabulated Dickey-Fuller distribution, and obtain its statistical significance:

tpp = sqrt(rss/obs / nw) * t2 - 1/2* (nw - rss/obs)/sqrt(nw) * obs*se2/sigma;
if nargout == 2
    tppsig = dfcrit (tpp, obs);
end

% End of function.

```

```
% REFERENCES:
%
% Dickey, David & Wayne Fuller (1979), "Distribution of the Estimators for Autoregressive
%   Time Series With a Unit Root", Journal of the American Statistical Association, vol.
%   74, no. 366 (June), pp. 427-431
%
% Diebold, Francis & Marc Nerlove (1990), "Unit Roots in Economic Time Series: A
%   Selective Survey", in George Rhodes Jr. & Thomas Fomby, eds., "Advances in
%   Econometrics: A Research Annual", vol. 8 ("Co-integration, Spurious Regressions, and
%   Unit Roots"), JAI Press, Greenwich, Connecticut, pp. 3-69
%
% Newey, Whitney & Kenneth West (1987), "A Simple, Positive Semi-definite,
%   Heteroskedasticity and Autocorrelation Consistent Covariance Matrix", Econometrica,
%   vol. 55, no. 3 (May), pp. 703-708
%
% Phillips, Peter (1987), "Time Series Regression with a Unit Root", Econometrica, vol.
%   55, no. 2 (March), pp. 277-301
%
% Phillips, Peter & Pierre Perron (1988), "Testing for a Unit Root in Time Series
%   Regression", Biometrika, vol. 75, no. 2 (June), pp. 335-346
%
% Schwert, William (1989), "Tests for Unit Roots: A Monte Carlo Investigation", Journal of
%   Business and Economic Statistics, vol. 7, no. 2 (April), pp. 147-159

% This implementation of the test follows the theoretical exposition in Hamilton, 1994,
% pp. 506-515, and Mills, 1993, pp. 54-55:
%
% Hamilton, James (1994), "Time Series Analysis", Princeton University Press, Princeton,
%   New Jersey
%
% Mills, Terence (1993), "The Econometric Modelling of Financial Time Series", Cambridge
%   University Press, Cambridge

% End of file.
```

Programme 4.7: Critical Dickey-Fuller Values and Level of Significance

```

function [sig, crit] = dfcrit (tratio, ssize, variant)

%DFCRIT Critical Dickey-Fuller values and level of significance, based on MacKinnon (1991)
%
% [SIG, CRIT] = DFCRIT (TRATIO, SSIZE, VARIANT) computes the critical values CRIT
%   of the Dickey-Fuller distribution for given sample size SSIZE and returns the
%   level SIG, if any, at which t-value TRATIO is significant.
%
%   Critical values are returned as a row vector for the 1%, 5% and 10% significance
%   levels of a one-sided test. SIG is the highest level at which TRATIO is significant;
%   1 indicates that TRATIO was not significant at the 10% level.
%
%   VARIANT should be set to 1, 2 or 3 in accordance to whether the Dickey-Fuller
%   regression contains
%
%       (1) no constant and no trend,
%       (2) a constant but no trend,
%       (3) a constant and a trend coefficient.
%
%   The relevant critical values CRIT are computed from a response surface developed by
%   MacKinnon (1991).
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.

%
%       Copyright (c) 23 April 1998 by Ludwig Kanzler
%       Department of Economics, University of Oxford
%       Postal: Christ Church, Oxford OX1 1DP, U.K.
%       E-mail: ludwig.kanzler@economics.oxford.ac.uk
%       Homepage:      http://users.ox.ac.uk/~econlkr
%       $ Revision: 1.0 $      $ Date: 23 April 1998 $

if nargin < 3
    variant = 2;
    if nargin < 2
        error('Insufficient number of arguments! Execution aborted.')
    end
elseif ~sum(variant == [1 2 3])
    error('Inadmissible value for VARIANT! Execution aborted.')
end

binf = [ -2.5658  -3.4335  -3.9638
         -1.9393  -2.8621  -3.4126
         -1.6156  -2.5671  -3.1279 ];

b1 = [ -1.960  -5.999  -8.353
        -0.398  -2.738  -4.039
        -0.181  -1.438  -2.418 ];

b2 = [ -10.04  -29.25  -47.44
        0.0     -8.36  -17.83
        0.0     -4.48  -7.58  ];

crit = binf(:, variant) + b1(:, variant)./ssize + b2(:, variant)./ssize^2;

switch sum(tratio <= crit)
    case 0
        sig = 1;
    case 1
        sig = 0.10;
    case 2
        sig = 0.05;
        if variant == 2
            if dftable (tratio, ssize) == 0.025;
                sig = 0.025;
            end
        end
    case 3
        sig = 0.01;
end

```

```

% End of main function.

function dfsig = dftable (tratio, ssize)

%DFTABLE Level of significance for Dickey-Fuller regression with constant, but no trend
%
%           Copyright (c) 13 March 1998 by Ludwig Kanzler
%           Department of Economics, University of Oxford
%           Postal: Christ Church, Oxford OX1 1DP, England
%           E-mail: ludwig.kanzler@economics.oxford.ac.uk
%           $ Revision: 1.03 $ $ Date: 15 September 1998 $

% DFCRIT lists the critical values for the Phillips-Perron Z(t) Test and for the Dickey-
% Fuller Test Based on Estimated OLS t Statistic, as tabulated in Fuller (1976) on p. 373,
% and reproduced, among others, in Hamilton (1994), p. 763, Table B.6, Case 2, and in
% Harvey (1990), p. 368, Table D, 2nd matrix.
%
% DFCRIT could probably be improved by using the extended tabulations of Guilkey & Schmidt
% (1989). Tables for the cases of no constant and of constant and trend could also be
% added here.
%
%
%                               sample
%                               size
dfcrit = [ -3.75 -3.33 -3.00 -2.63     25
           -3.58 -3.22 -2.93 -2.60     50
           -3.51 -3.17 -2.89 -2.58    100
           -3.46 -3.14 -2.88 -2.57    250
           -3.44 -3.13 -2.87 -2.57    500
           -3.43 -3.12 -2.86 -2.57    inf

           0.01  0.025 0.05  0.10   NaN ];

%           prob(tratio<=table entry)

% Find the entry matching sample size and t-ratio:

matchsize = find(ssize >= dfcrit(1:6,5));
matchlevel = find(tratio <= dfcrit(matchsize(end),1:4));

% Assign the corresponding level of significance, if any:

if matchlevel
    dfsig = dfcrit(7, matchlevel(1));
else
    dfsig = 1;
end

% End of sub-function.

% REFERENCES:
%
% Fuller, Wayne (1976), "Introduction to Statistical Time Series", John Wiley & Sons, New
%   York
%
% Guilkey, David & Peter Schmidt (1989), "Extended Tabulations for Dickey-Fuller Tests",
%   Economics Letters, vol. 31, no. 4, pp. 355-357
%
% Hamilton, James (1994), "Time Series Analysis", Princeton University Press, Princeton,
%   New Jersey
%
% Harvey, Andrew (1990), "The Econometric Analysis of Time Series", 2nd edition, MIT
%   Press, Cambridge, Massachusetts
%
% MacKinnon, James (1991), "Critical Values for Cointegration Tests", Chapter 13 in
%   Robert Engle & Clive Granger, eds., "Long-run Economic Relationships: Readings
%   in Cointegration", Oxford University Press, Oxford, pp. 267-276

% End of file.

```

Programme 4.8: Durbin-Watson Test for First-Order Serial Correlation

```

function [dw, dwsigup, dwsiglow] = dwatson (series)

%DWATSON Durbin-Watson d-statistic and significance level for the null hypothesis: DW = 2
%
% [DW, DWSIGUP, DWSIGLOW] = DWATSON(SERIES) computes
%
%   - the Durbin-Watson (1950, 1951) statistic d of serial correlation, and
%
%   - the significance level, if any, at which the null hypothesis DW = 2 is rejected
%     against either of the one-sided alternatives (but not both!), using both upper-
%     bound and lower-bound critical values. Possible values are 0.01, 0.05, 1, NaN.
%
%   This lookup is valid only for regressions with one explanatory variable & constant.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.

%
%           Copyright (c) 17 March 1998 by Ludwig Kanzler
%           Department of Economics, University of Oxford
%           Postal: Christ Church, Oxford OX1 1DP, U.K.
%           E-mail: ludwig.kanzler@economics.oxford.ac.uk
%           Homepage:   http://users.ox.ac.uk/~econlrk
%           $ Revision: 1.23 $$ Date: 15 September 1998 $

% Compute the Durbin-Watson statistic, which is defined as
% sum((series(2:end)-series(1:end-1)).^2)/sum((series(2:end)-mean(series(2:end))).^2),
% but which can be computed much quicker as follows:
series = series(:);
dseries = series(2:end)-series(1:end-1);
dw      = diag((dseries'*dseries)./(series'*series));

if nargin > 1
    % DWCRIT lists the critical values for the d-statistic for one explanatory variable
    % (excluding the constant) as published in Durbin & Watson (1951) and Savin & White
    % (1977), reproduced in Judge et al. (1988). Add entries if you need/know more values.
    %
    %
    %                               sample
    %                               size
    dwcrit = [ NaN   NaN   0.610 1.400   6
              NaN   NaN   0.700 1.356   7
              NaN   NaN   0.763 1.332   8
              NaN   NaN   0.824 1.320   9
              NaN   NaN   0.879 1.320  10
              NaN   NaN   0.927 1.324  11
              NaN   NaN   0.971 1.331  12
              NaN   NaN   1.010 1.340  13
              NaN   NaN   1.045 1.350  14
              0.81  1.07  1.077 1.361  15
              0.84  1.09  1.106 1.371  16
              0.87  1.10  1.133 1.381  17
              0.90  1.12  1.158 1.391  18
              0.93  1.13  1.180 1.401  19
              0.95  1.15  1.201 1.411  20
              0.97  1.16  1.221 1.420  21
              1.00  1.17  1.239 1.429  22
              1.02  1.19  1.257 1.437  23
              1.04  1.20  1.273 1.446  24
              1.05  1.21  1.288 1.454  25
              1.07  1.22  1.302 1.461  26
              1.09  1.23  1.316 1.469  27
              1.10  1.24  1.328 1.476  28
              1.12  1.25  1.341 1.483  29
              1.13  1.26  1.352 1.489  30
              1.15  1.27  1.363 1.496  31
              1.16  1.28  1.373 1.502  32
              1.17  1.29  1.383 1.518  33
              1.18  1.30  1.393 1.514  34
              1.19  1.31  1.402 1.519  35
              1.21  1.32  1.411 1.525  36

```

```

1.22  1.32  1.419  1.530  37
1.23  1.33  1.427  1.535  38
1.24  1.34  1.435  1.540  39
1.25  1.34  1.442  1.544  40
1.29  1.38  1.475  1.566  45
1.32  1.40  1.503  1.585  50
1.36  1.43  1.528  1.601  55
1.38  1.45  1.549  1.616  60
1.41  1.47  1.567  1.629  65
1.43  1.49  1.583  1.641  70
1.45  1.50  1.598  1.652  75
1.47  1.52  1.611  1.662  80
1.48  1.53  1.624  1.671  85
1.50  1.54  1.635  1.679  90
1.51  1.55  1.645  1.687  95
1.52  1.56  1.654  1.694  100
NaN   NaN   1.720  1.746  150
NaN   NaN   1.758  1.778  200

0.01  0.01  0.05  0.05  NaN ]; % significance level for...
%     ...lower upper lower upper bounds

[r,c] = size(dwcrit);

% Find the entry matching sample size and dw statistic:
matchsize = find(length(series) >= dwcrit(1:r-1, c));
matchlow  = find(min(dw, 4-dw) <= dwcrit(matchsize(end),1:2:c-1));
matchup   = find(min(dw, 4-dw) <= dwcrit(matchsize(end),2:2:c-1));

% Assign the corresponding level of significance, if any:
if (matchlow | matchup)
    dwsiglow = dwcrit(r, matchlow(1));
    dwsigup  = dwcrit(r, matchup(1));
else
    if length(series) > 1.25*dwcrit(r-1, c)
        dwsiglow = NaN;
        dwsigup  = NaN;
    else
        dwsiglow = 1;
        dwsigup  = 1;
    end
end
end
end

% End of function.

% NOTE that there is a case for reporting the significance at the UPPER bound only:
%     the increase in the Type I error arising from ignoring the inconclusive area is
%     likely to be small when (see Harvey, 1990, p. 203)
%     - there is only one dependent variable (so k = 2),
%     - there are a large number of observations, and
%     - the data is in levels).

% REFERENCES:
%
% Durbin, James & G.S. Watson (1950), "Testing for Serial Correlation in Least Squares
% Regression I", Biometrika, vol. 37, pp. 409-428 [corrections in Durbin & Watson
% (1951), pp. 177-178]
%
% Durbin, James & G.S. Watson (1951), "Testing for Serial Correlation in Least Squares
% Regression II", Biometrika, vol. 38, pp. 159-178
%
% Harvey, Andrew (1990), "The Econometric Analysis of Time Series", 2nd edition,
% Press, Cambridge, Massachusetts, pp. 221-223
%
% Judge, George, Carter Hill, William Griffiths, Helmut Lütkepohl & Tsoung-Chao Lee
% (1988), "Introduction to the Theory and Practice of Econometrics", 2nd edition, John
% Wiley & Sons, New York, pp. 991-995, Table 5, k=2
%
% Savin, Eugene & Kenneth White (1977), "The Durbin-Watson Test for Serial Correlation
% with Extreme Sample Sizes or Many Regressors", Econometrica, vol. 45, no. 8
% (November), pp. 1989-1996

% End of file.

```

Programme 4.9: Durbin-H Test for First-Order Serial Correlation

```

function [dh, dhsig] = durbinh (dw, se2, n, h1)

%DURBINH Durbin h statistic and significance of the hypothesis of no serial correlation
%
% [DH, DHSIG] = DURBINH (DW, SE2, N, H1)
%
% - computes the Durbin h statistic DH (a statistic of autocorrelation which is robust to
%   the inclusion of lagged dependent variables in the regression, see Durbin, 1970), and
%
% - evaluates the associated null hypothesis of no serial correlation
%   * against the alternative of EITHER positive OR negative autocorrelation
%     if flag H1 is 1 (one-sided test), or
%   * against the alternative of ANY autocorrelation
%     if flag H1 is 2 (two-sided test, default),
%   and returns the level of significance DHSIG at which H0 is rejected.
%
% Required input arguments are
% - the Durbin-Watson statistic DW, computed from the residuals of the associated
%   regression, using, for example, the author's DWATSON m-function,
% - the est. standard error SE2 of the coefficient of the 1st lag in the regression, and
% - the number of residuals N (= OBS - DLAGS - 1).
%
% If the MATLAB Statistics Toolbox is not installed, DHSIG assumes NaN. If N*SE2^2 >= 1,
% the Durbin h statistic cannot be calculated, and so both DH and DHSIG assume NaN.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
% Copyright (c) 17 March 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.11 $$ Date: 15 September 1998 $
if nargin == 3
    h1 = 2;
end

dh = (1-dw/2) * sqrt(n/(1-n*se2^2));

if isreal(dh)
    if h1 == 1 & exist('normcdf.m','file')
        dhsig = 1-normcdf(dh,0,1);
    elseif exist('normcdf.m','file') & nargin == 2
        dhsig = min(normcdf(dh,0,1), 1-normcdf(dh,0,1))*2;
    elseif nargin == 2
        dhsig = NaN;
    end
else
    dh = NaN;
    dhsig = NaN;
end

% REFERENCE:
%
% Durbin, James (1970), "Testing for Serial Correlation in Least-Squares Regression When
%   Some of the Regressors are Lagged Dependent Variables", Econometrica, vol. 38, no. 3
%   (May), pp. 410-421
%
% SEE ALSO:
%
% Harvey, Andrew (1990), "The Econometric Analysis of Time Series", 2nd edition, MIT
%   Press, Cambridge, Massachusetts, pp. 275-277
%
% Judge, George, Carter Hill, William Griffiths, Helmut Lütkepohl & Tsoung-Chao Lee
%   (1988), "Introduction to the Theory and Practice of Econometrics", 2nd edition, John
%   Wiley & Sons, New York, p. 401

% End of file.

```

Programme 4.10: Engle's Test for ARCH

```

function [arch, archsig] = archtest (residuals, lags)

%ARCHTEST Engle's (1982) test for ARCH
%
% [ARCH, ARCHSIG] = ARCHTEST (RESIDUALS, LAGS) tests the null hypothesis of independently
% and identically distributed RESIDUALS against the alternative of linear dependence in
% conditional second moments (usually autoregressive conditional heteroskedasticity)
% for the lag order(s) specified by LAGS, returning ARCH, Engle's ARCH statistic(s),
% and ARCHSIG, the level(s) of significance at which H0 is rejected.
%
% RESIDUALS should be a vector of residuals obtained from some regression whose
% goodness of fit one wishes to evaluate with the ARCH test. Alternatively, this can
% also be a differenced time series.
%
% LAGS can be a scalar (default is 1) or a vector of integers in any order.
% For example, if LAGS = 5, results are only returned for the null hypothesis of no 5th
% order ARCH effects, but if LAGS = [3 1 5], ARCH and ARCHSIG will be three-point
% vectors carrying the results for the test at lag orders 3, 1 and 5 (in this order).
%
% ARCHSIG assumes NaN values if the MATLAB Statistics Toolbox is not installed.
%
% The cost of computation depends on the size of each lag order specified in LAGS.
% This programme requires far more processing power than QSTAT.M. See the source code
% comments for an explanation of how the test is conducted and the bottom of the
% script for a list of references.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.

%
% Copyright (c) 14 May 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.1 $ $ Date: 15 November 1998 $

% STEP 0: Set default value for LAGS, if the corresponding input argument is missing, and
% obtain information about the size of the input arguments:

if nargin < 2, lags = 1; end
nres = length(residuals(:));
lags = lags(:)';
nlags = length(lags);
maxlag = max(lags);

% STEP 1: Compute the squares of the residuals obtained from the original regression:
res2 = residuals(:).^2;

% STEP 2: Form a matrix of constant and lagged squared residuals (up to the highest lag
% order) on which to regress the squared residuals:

vars(1:nres,1:1+maxlag) = 1;
for i = 1 : maxlag
    vars(1:nres-i, i+1) = res2(1+i:nres);
end

% For each lag order, perform...

R2(1:nlags) = 0;
for i = 1 : nlags

    % STEP 3: ... "run" the regression and obtain the predicted values, and ...

    pred = vars(1:nres-lags(i), 1:1+lags(i)) * (vars(1:nres-lags(i), 1:1+lags(i))...
        \ res2(1:nres-lags(i)));

```

```

% STEP 4: ... compute the non-centred coefficient of determination R2.

R2(i) = sum((pred - sum(pred) / (nres-lags(i))).^2 )...
       / sum((res2(1:nres-lags(i)) - sum(res2(1:nres-lags(i)))) / (nres-lags(i)).^2 );
end

% STEP 5: Following Engle (1982), the test statistic is then given by

arch = nres .* R2;

% (Unfortunately, nobody appears to have checked yet whether Ljung & Box's (1978) small
% sample correction, here simply (nres+2)./(nres-lags).* applied to the above line
% improves the size of the test).

% STEP 6: Evaluate the level at which the test statistics are significant under the
% respective null hypotheses of no ARCH effects:

if exist('chi2cdf.m','file') & nargout == 2
    archsig = 1 - chi2cdf(arch, lags);
elseif nargout == 2
    archsig = NaN*lags;
end

% End of function.

% REFERENCES:
%
% Engle, Robert (1982), "Autoregressive Conditional Heteroskedasticity with Estimates of
% the Variance of United Kingdom Inflation", Econometrica, vol. 50, pp. 987-1007
%
% Ljung, Greta & George Box (1978), "On a Measure of Lack of Fit in Time Series Models",
% Biometrika, vol. 65, no. 2, pp. 297-303

% SOME USEFUL INFORMATION ABOUT THE ARCH TEST CAN ALSO BE FOUND AMONG THE FOLLOWING:
%
% Bollerslev, Tim, Ray Chou & Kenneth Kroner (1992), "ARCH Modeling in Finance: A Review
% of the Theory and Empirical Evidence", Journal of Econometrics, vol. 52, no. 1/2
% (April/May), pp. 5-59; revised version of earlier working paper with Narayanan
% Jayaraman entitled "ARCH Modeling in Finance: A Selective Review of the Theory and
% Empirical Evidence, with Suggestions for Future Research"
%
% Bollerslev, Tim, Robert Engle & Daniel Nelson (1994), "ARCH Models", in Robert Engle &
% Daniel McFadden, eds., "Handbook of Econometrics", vol. 4, North-Holland/Elsevier,
% Amsterdam
%
% Hamilton, James (1994), "Time Series Analysis", Princeton University Press, Princeton,
% New Jersey, pp. 664-665
%
% Harvey, Andrew (1990), "The Econometric Analysis of Time Series", 2nd edition, MIT
% Press, Cambridge, Massachusetts, pp. 221-223
%
% Mills, Terence (1993), "The Econometric Modelling of Financial Time Series", Cambridge
% University Press, Cambridge, p. 107

% End of file.

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Check and transformation of input arguments %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin < 5
    maxram = 150;
elseif maxram > 500
    disp('Are you sure you have so much memory available?')
    error('If so, you need to edit the code, otherwise try again with a lower value.')
end

if nargin < 4
    flag = 0;
elseif ~any(flag == [0 1])
    error('Unknown method for determining dimensional distance; try again with 0 or 1.')
end

if nargin < 3
    distance = 1.5;
elseif distance < 0
    error('The dimensional distance parameter must be positive.')
elseif flag == 1 & distance > 1
    error('The correlation integral cannot exceed 1.')
end

if nargin < 2
    maxdim = 2;
elseif maxdim < 1
    error('The dimension needs to be at least 1.');
```

```

end

if nargin < 1
    error('Cannot compute the BDS statistic on nothing.')
end

[rows,cols] = size(series);
if rows > 1 & cols == 1
    n = rows;
    series = series';
elseif cols > 1 & rows == 1
    n = cols;
elseif cols > 1 & rows > 1
    n = cols*rows;
    series = series(:)'; % transformation into a row vector
    disp(sprintf('\aTransformed matrix input into a single column.'))
else
    error('Cannot compute the BDS statistic on a scalar!')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Determination of and preparations for fastest method given MAXRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fastbuild    = 0.000016 * (1:52) .* pow2(1:52); % memory requirements
slowbuild    = 0.000045          * pow2(1:52); % for the various
holdinfo     = 0.000005          * pow2(1:52); % algorithms in
wordtable    = 0.000008 * n^2 ./ (1:52); % megabytes for
bitandop     = 0.000024 * n^2 ./ (1:52); % given N

[ram1, bits1] = min(fastbuild + holdinfo + wordtable + bitandop); % number of bits for
[ram2, bits2] = min(fastbuild + holdinfo + wordtable);           % which each of six
[ram3, bits3] = min(slowbuild + holdinfo + wordtable + bitandop); % methods uses minimum
[ram4, bits4] = min(slowbuild + holdinfo + wordtable);           % memory; this memory
[ram5, bits5] = min(                                             % is given by
    wordtable + bitandop);
[ram6, bits6] = min(                                             % ram1, ram2,..., ram6
    wordtable);

if ram1 < maxram | ram2 < maxram
    if ram1 < maxram
        method = 1;
        bits = bits1; ram = ram1;
    else
        method = 2;
        bits = bits2; ram = ram2;
        stepping = floor((maxram-ram)*bits/n/0.000024); % algorithm without exceeding MAXRAM
    end
end

```

```

% Vector BITINFO lists the number of bits set for each integer between 0 and 2^bits
% (corresponding to the indices of the vector shifted by 1). See Kanzler (1998) for an
% explanation.

bitinfo = uint8(sum(rem(floor((0:pow2(bits)-1)'*pow2(1-bits:0)),2),2));

elseif ram3 < maxram | ram4 < maxram
  if ram3 < maxram
    method = 3;
    bits = bits3; ram = ram3;
  else
    method = 4;
    bits = bits4; ram = ram4;
    stepping = floor((maxram - ram) * bits / n / 0.000024);
  end

  bitinfo(1:pow2(bits), :) = uint8(0); % the same as above, but created through
  for bit = 1 : bits % a loop, which consumes less memory
    bitinfo(1:pow2(bits)) = sum([bitinfo, ...
      kron(ones(pow2(bits-bit),1), [zeros(pow2(bit-1),1); ones(pow2(bit-1),1)])],2);
  end

elseif ram5 < maxram | ram6 < maxram
  if ram5 < maxram
    method = 5;
    bits = bits5; ram = ram5;
  else
    method = 6;
    bits = bits6; ram = ram6;
    stepping = floor((maxram - ram) * bits / n / 0.000024);
  end

else
  disp('Insufficient amount of memory. Allocate more memory to the system')
  disp('or reduce the number of observations, then try again.')
  error(' ')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Determination of dimensional distance EPSILON %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The empirical investigation by Kanzler (1998) shows that choosing EPSILON such that the
% first-order correlation integral is around 0.7 yields the most efficient estimation of
% low-dimensional BDS statistics. Hence the objective here is to choose EPSILON such that,
% say, 70% of all observations lie within distance EPSILON to each other. If desired, the
% programme first determines EPSILON as to fulfil this or a similar requirement.
%
% The conceptually simplest way of setting up the calculation of distance among all
% observations is to define a two-dimensional table D (for "distance") of length and width
% N and assign to each co-ordinate (x,y) the result of the problem ABS(x-y).
%
% In principle, the entire table could thus be created with the following one-line
% statement:
%       D = ABS( SERIES(ONES(1000,1),:) - SERIES(ONES(1000,1),:) )
%
% Since the lower triangle of the table only replicates the upper triangle and since the
% diagonal values represent own values (ones) which are not desired to be included in the
% calculation, only the upper triangle receives further attention.
%
% Unfortunately, sewing all the row vectors of the upper triangle together to form one
% single (row) vector makes indexing very messy. To aid understanding of the vector-space
% indexing used here (as well as in the optional sub-function further below), one may wish
% to refer to the following exemplary matrix table (N=7):
%
%
%           * * * *c o l u m n* * * *
%   I      1  2  3  4  5  6  7
%
% * 1      *  1  2  3  4  5  6
% * 2      .  *  7  8  9 10 11
% r 3      .  .  * 12 13 14 15
% o 4      .  .  .  * 16 17 18
% w 5      .  .  .  .  * 19 20
% * 6      .  .  .  .  .  * 21
% * 7      .  .  .  .  .  .  *

Using this example, it is easy to verify
that column vector I is defined by the
following indices in vector space:
      I+(0 : I-2)*N - CUMSUM(1 : I-1)

More generally, column vector I starting only
in row J is:
      I+(J-1 : I-2)*N - SUM(1:J-1)-CUMSUM(J : I-1)

Row vector I is given by indices:
      1+(I-1)*(N-1)-SUM(1:I-2) : I*(N-1)-SUM(1:I-1)

```

```

%
% (A formal derivation of the above formulae is beyond the scope of this script.)
%
% To calculate a percentile of the distribution of distance values, the row vector is
% sorted (unfortunately, this requires a lot of time and RAM in MATLAB).

if ~flag
    demeaned = series-sum(series)/n; % fastest algorithm for
    epsilon = distance * sqrt(demeaned*demeaned'/(n-1)); % computing the standard
    clear demeaned % to save memory % deviation of SERIES

elseif 0.000008 * 3 * sum(1:n-1) < maxram % check memory requirements for DIST and sorting
    dist(1:sum(1:n-1)) = 0;
    for i = 1 : n-1
        dist(1+(i-1)*(n-1)-sum(0:i-2):i*(n-1)-sum(1:i-1)) = abs(series(i+1:n)-series(i));
    end
    sorted = sort(dist);
    epsilon = sorted(round(distance*sum(1:n-1))); % DISTANCEth percentile of SORTED series
    clear dist sorted
else
    error('Insufficient RAM to compute EPSILON; allocate more memory or use METHOD = 1.')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation and storage of one-dimensional distance information %%%%%%%%%%%%%%%

% Similarly to the above, a two-dimensional table C (for "close") of length and width N
% can be defined by assigning to each co-ordinate (x,y) the result of the problem
% ABS(x-y) <= EPSILON; (x,y) assumes the value 1 if the statement is true and 0 otherwise.
%
% Formally, for given EPSILON:
%
%                               C(x,y) = 1 if ABS(x-y) <= EPSILON
%                               = 0 otherwise
%
% Once again, the resulting information needs to be stored in the most efficient way.
% In this implementation, this is done by chopping each row of the table into "words" of
% several bits, the precise number of bits per word being determined by the above
% algorithms. One "word" is thus represented by one integer. This slashes the size of the
% table by the number of bits. See Kanzler (1998) for more details.
%
% The below routine stores all rows of the upper triangle of the conceptual table
% (described in Kanzler, 1998) left-aligned and assigns zeros to all other elements.
%
% As will also be explained further below, the computation of parameter K requires the sum
% of each FULL row, i.e. each row including the elements in the lower triangle and on the
% diagonal. The "missing" bits correspond to the sums over each column in the upper
% triangle, and these sums are also computed and stored in the below loop. And to make
% matters simple, diagonal values are allocated to the column sums by initialising them
% with value 1. See also Kanzler (1998).

colsum(1:n) = 1;
rowsum(1:n) = 0;
nwords = ceil((n-1)/bits);
wrmtx(1:n-1,1:nwords) = 0; % initialisation of bit-word table

for row = 1 : n-1
    bitvec = abs(series(1+row:n) - series(row)) <= epsilon;
    rowsum(row) = sum(bitvec);
    colsum(1+row:n) = colsum(1+row:n) + bitvec;
    nwords = ceil((n-row)/bits);
    wrmtx(row,1:nwords) = (reshape([bitvec,zeros(1,nwords*bits-n+row)],...%transformation
        bits, nwords)' *pow2(0:bits-1)')'; %into bit-words
end
clear series bitvec

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of one-dimensional correlation estimates %%%%%%%%%%%%%%%

% C(1), the fraction (or estimated probability) of pairs in SERIES being "close" in the
% first dimension is just the average over ALL unique elements. C(1) is hence the most
% efficient estimator of C(1), and the resulting estimate is used in the computation of
% SIGMA(M) further below.
%
% However, for the difference term C(M) - C(1)^M of the BDS statistic (see further below)
% to follow SIGMA asymptotically, both C(M) and C(1) need to be estimated over the same
% length vector, and so MAXDIM different C1's need to be estimated here:

```

```

%
%
%           N      N
%   C1(M) = 2/(N-M+1)/(N-M) * SUM   SUM   B(S,T)
%                               S=M   T=S+1
%
% Each C1(M) is easily computed from the sum of all bits set in rows M to N-1 divided by
% the appropriate total number of bits.

bitsum(maxdim:-1:1) = cumsum([sum(rowsum(maxdim:n-1)), rowsum(maxdim-1:-1:1)]);
c1      (maxdim:-1:1) = bitsum(maxdim:-1:1) ./ cumsum([sum(1:n-maxdim), n-maxdim+1 : n-1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of parameter K %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% A parameter needed to estimate SIGMA(M) is K, which is defined as:
%
%           N      N      N
%   K = 6/N/(N-1)/(N-2)* SUM   SUM   SUM {C(T,S)*C(S,R) + C(T,R)*C(R,S) + C(S,T)*C(T,R)} / 3
%                       T=1   T=T+1 R=S+1
%
% As is readily apparent, a literary computation of the above would be very processing
% intensive, e.g.:
%
%           HT(1 : N) = 0;
%           FOR T = 1 : N
%               HS(1 : N-T) = 0;
%               FOR S = T+1 : N
%                   HR(1 : N-S) = 0;
%                   FOR R = S + 1 : N
%                       HR(R-S) = (C(T,S)*C(S,R) + C(T,R)*C(R,S) + C(S,T)*C(T,R)) / 3;
%                   END
%                   HS(S-T) = SUM(HR(1 : N-S));
%               END
%               HT(T) = SUM(HS(1 : N-T));
%           END
%           K = SUM(HT) * 6 / N/(N-1)/(N-2);
%
% To understand what k actually estimates, and how this estimation can be made
% computationally more efficient, see Kanzler (1998).
%
% The above FOR loop computes the sum over each row and over each column including the
% diagonal in the upper triangle. To compute K from this, the sum of the squares of the
% row and column sums needs to be adjusted as reasoned above, whereby the sum of all
% elements in table C is given by twice the sum of all vector elements plus the diagonal
% values.

fullsum = rowsum + colsum;
k      = (fullsum*fullsum' + 2*n - 3*(2*bitsum(1)+n)) / n/(n-1)/(n-2);
clear rowsum colsum fullsum bitsum

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of correlation estimates and SIGMA for higher dimensions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% C(M), the M-dimension correlation
% estimate, is defined as:
%
%           N      N      M-1
%   C(M) = 2/(N-M+1)/(N-M) * SUM   SUM   PROD B(S-J, T-J)
%                               S=M   T=S+1   J=0
%
% To see how C can be computed for M > 1, see Kanzler (1998).
%
% In practice, the required BITAND-operation can be performed on the entire table at once
% by replacing the entire table between rows M and N-1 with the result of the BITAND-
% operation between the table formed by rows M to N-1 and M-1 to N-2. But this works only
% if sufficient memory is available (methods 1, 3 and 5). Otherwise, the BITAND-operation
% has to be performed by looping BACKWARDS through the table, taking as many rows as
% possible at once (methods 2, 4 and 6).
%
% The number of bits set in rows M to N-1 (inclusive) is counted either in one go or
% through the above loop by looking up the number of bits set for each integer (LeBaron,
% 1997, uses a similar method), or, if memory was insufficient to create the required
% BITINFO array, by column-wise brute-force counting.
%
% MATLAB uses logarithms to compute powers, and this can result in minute deficiencies in
% accuracy. To avoid this, integer powers are computed by separate functions in this
% script (see further below). Otherwise, SIGMA would be calculated as follows:
%
%   sigma(m-1) = 2*sqrt(k^m + 2*k.^(m-(1:m-1))*(c1(1).^(2*(1:m-1)))'...
%               + (m-1)^2*c1(1)^(2*m) - m^2*k*c1(1)^(2*m-2));

```

```

for m = 2 : maxdim
    bitcount = 0;

    if sum(method == [1 3])
        wrdmtrx(m:n-1,:) = bitand(wrdmtrx(m:n-1,:), wrdmtrx(m-1:n-2,:)); % BITAND and bit
        bitcount = sum(sum(bitinfo(wrdmtrx(m:n-1,:)+1))); % count all at once

    elseif sum(method == [2 4])
        for row = n-stepping : -stepping : m+1 % BITAND
            wrdmtrx(row:row+stepping-1,:) = bitand(wrdmtrx(row:... % and bit
                row+stepping-1,:), wrdmtrx(row-1:row+stepping-2,:)); % count in
            bitcount=bitcount+sum(sum(bitinfo(wrdmtrx(row:row+stepping-1,:)+1))); % backward
        end % loops
        wrdmtrx(m:row-1,:) = bitand(wrdmtrx(m:row-1,:), wrdmtrx(m-1:row-2,:)); % through
        bitcount = bitcount + sum(sum(bitinfo(wrdmtrx(m:row-1,:)+1))); % the table

    elseif method == 5
        wrdmtrx(m:n-1,:) = bitand(wrdmtrx(m:n-1,:), wrdmtrx(m-1:n-2,:)); % BITAND at once...
        for col = 1 : ceil((n-1)/bits) % bit count
            bitcount = bitcount + sum(sum(rem(floor(wrdmtrx(m:... % by brute force
                n-1-(col-1)*bits, col) * pow2(1-bits:0)), 2))); % in loops
        end

    else
        for row = n-stepping : -stepping : m+1 % BITAND
            wrdmtrx(row:row+stepping-1,:) = bitand(wrdmtrx(row:... % opera-
                row+stepping-1,:), wrdmtrx(row-1:row+stepping-2,:)); % tions
        end % and brute-
        wrdmtrx(m:row-1,:) = bitand(wrdmtrx(m:row-1,:), wrdmtrx(m-1:row-2,:)); % force bit
        for col = 1 : ceil((n-1)/bits) % counting
            bitcount = bitcount + sum(sum(rem(floor(wrdmtrx(m:... % in loops
                n-1-(col-1)*bits, col) * pow2(1-bits:0)), 2)));
        end
    end

    c(m-1) = bitcount / sum(1:n-m); % indexing of
    sigma(m-1) = 2*sqrt(prod(ones(1,m)*k) + 2*ivp(k,m-(1:m-1),m-1)... % C and SIGMA
        *(ivp(c1(1),2*(1:m-1),m-1))' + (m-1)*(m-1)... % runs from 1
        *prod(ones(1,2*m)*c1(1)) - m*m*k*prod(ones(1,2*m-2)*c1(1))); % to MAXDIM-1
end
clear wrdmtrx

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of the BDS statistic and level of significance %%%%%%%%%%%%%%%

% Under the null hypothesis of independence, it is obvious that the time-series process
% has the property  $C(1)^M = C(M)$ . In finite samples,  $C(1)$  and  $C(M)$  are consistently
% estimated by  $C1(M)$  and  $C(M)$  as above. Also, Brock et al. (1996) show that the standard
% deviation of the difference  $C(M) - C1(M)^M$  can be consistently estimated by  $SIGMA(M)$ 
% divided by  $\sqrt{N-M+1}$ , where:
%
%
% 
$$SIGMA(M)^2 = 4 * [K^M + 2 * \sum_{J=1}^{M-1} \{K^{(M-J)} * C^{(2*J)}\} + (M-1)^2 * C^{(2*M)} - M^2 * K * C^{(2*M-2)}]$$

%
% and  $C = C1(1)$  and  $K$  as above.
%
% For given  $N$  and  $EPSILON$ , the BDS Statistic  $C(M) - C1(M)^M$ 
% is defined as the ratio of the two terms:  $W(M) = \sqrt{N-M+1} * \frac{C(M) - C1(M)^M}{SIGMA(M)}$ 
%
%
% Since it follows asymptotically the normal distribution with mean 0 and variance 1,
% hypothesis testing is straightforward. If available, this is done here using function
% NORMCDF of the MATLAB Statistics Toolbox.
%
% Integer powers are again calculated by a sub-routine which is more accurate than the
% MATLAB built-in power function; without using the sub-routine, the line for calculating
%  $W$  would be:  $w = \sqrt{n-(2:maxdim)+1} .* (c - c1(2:maxdim).^{(2:maxdim)}) ./ sigma;$ 

```



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation and storage of one-dimensional distance information %%%%%%%%%%%%%%%

% Recall that in the implementation of the main function above, table C is stored in bit-
% representation. When this is not possible or desirable, the second best method is to use
% one continuous vector of unassigned 8-bit integers (called UINT8). This, however,
% requires version 5.1 or higher, and a similar option may not be available in other high-
% level languages. Implementation does not depend on the ability to use unassigned low-bit
% integers and would work equally with double-precision integers, but the memory
% requirements would, of course, be higher. Using UINT8's is still a rather inefficient
% way of storing zeros and ones, which in principle require only a single bit each. On the
% PC, MATLAB actually requires "only" around 5 bytes for each UNIT8.

b(1:pairs) = uint8(0);
for i = 1 : n-1
    b(1+(i-1)*(n-1)-sum(0:i-2):i*(n-1)-sum(1:i-1)) = abs(series(i+1:n)-series(i))<=epsilon;
end
clear series

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of parameter K %%%%%%%%%%%%%%%

sums(1 : n) = 0;
for i = 1 : n
    sums(i) = sum(b(i+(0 : i-2)*n - cumsum(1 : i-1)))...           % sum over column I
            + 1 ...                                               % diagonal element
            + sum(b(1+(i-1)*(n-1)-sum(1:i-2) : i*(n-1)-sum(1:i-1))); % sum over row I
end
k = (sum(sums.^2) + 2*n - 3*(2*sum(b)+n)) / n/(n-1)/(n-2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of one-dimensional correlation estimates %%%%%%%%%%%%%%%

bitsum(1:maxdim) = sum(b(1+(maxdim-1)*(n-1)-sum(0:maxdim-2) : pairs));
for m = maxdim-1 : -1 : 1
    bitsum(m) = bitsum(m+1) + sum(b(1+(m-1)*(n-1)-sum(0:m-2):m*(n-1)-sum(1:m-1)));
end
c1(maxdim:-1:1) = bitsum(maxdim:-1:1) ./ cumsum([sum(1:n-maxdim), n-maxdim+1 : n-1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of correlation estimates and SIGMA for higher dimensions %%%%%%%%%%%%%%%

for m = 2 : maxdim

    % Indexing in vector space once again follows the rules set out above. Multiplication
    % is done by moving up column by column into north-west direction, so counter I runs
    % backwards in the below WHILE loop until the Mth column (from the left) is reached:
    i = n;
    while i - m

        % Multiplication is not defined on UINT8 variables and translating the columns
        % twice, once from UINT8 to DOUBLE integer and then back to UINT8, would be
        % inefficient, so it is better to sum entries (this operation - undocumented by
        % MATLAB - is defined, and even faster than the documented FIND function!) and
        % compare them against the value 2:
        b(i + (m-1 : i-2)*n - sum(1:m-1) - cumsum(m : i-1)) = ...
            sum([ b(i + (m-1 : i-2)*n - sum(1:m-1) - cumsum(m : i-1)); ...
                  b(i-1 + (m-2 : i-3)*n - sum(1:m-2) - cumsum(m-1 : i-2)) ]) == 2;

        % The sum over each column is computed immediately after that column has been
        % updated. To store the column sums, the vector SUMS already used above for the row
        % sums is recycled (this is more memory-efficient than clearing the above SUMS
        % vector and defining a new vector of the column sums, because in the latter case,
        % MATLAB's memory space will end up being fragmented by variables K and C added to
        % the memory in the meantime!):
        sums(i) = sum(b(i + (m-1 : i-2)*n - sum(1:m-1) - cumsum(m : i-1)));
        i = i - 1;
    end

    c(m-1) = sum(sums(m+1:n)) / sum(1:n-m);
    sigma(m-1) = 2*sqrt(k^m + 2*k.^(m-(1:m-1))*(c1(1).^(2*(1:m-1)))'... % could use above
                + (m-1)^2*c1(1)^(2*m) - m^2*k*c1(1)^(2*m-2)); % inter-power sub-
end
                                                                    % functions instead

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Computation of the BDS statistic and level of significance %%%%%%%%%%%%%%%

w = sqrt(n-(2:maxdim)+1) .* (c-c1(2:maxdim).^(2:maxdim)) ./ sigma; % or use sub-functions

```


Programme 5.2: Significance of the BDS Statistic in Small Samples

```

function sig = bdssig(w, n, m, eps)

%BDSSIG Significance level of the BDS statistic in small samples
%
% SIG = BDSSIG (W, N, M, EPS) evaluates the significance of BDS statistics W under the
% null hypothesis of iidness, using Kanzler's (1998) finite-sample quantile values.
%
% The significance levels can only assume 0.005, 0.010, 0.025, 0.050 or 1 (indicates
% failure to reject the null hypothesis) in value. These levels must be doubled to
% conduct what is normally a two-sided BDS test.
%
% N is the size of the sample on which the BDS statistics were computed.
%
% M can be either a scalar or a vector (corresponding to W) and represents
% the embedding dimension(s) for which the BDS statistics were computed.
% Only integers between 2 and 15 inclusive are permitted.
%
% EPS is the dimensional distance for which the BDS statistics were calculated.
% This is given in units of the standard deviation of approximately normally
% distributed samples, and only values 0.5, 1.0, 1.5 and 2.0 are allowed.
%
% See Kanzler (1998) on how estimation of the BDS statistic is to be correctly
% sized in normally as well as non-normally distributed samples. As the paper
% shows, EPS = 0.5 or 1.0 yield in many cases BDS distributions which are badly
% shaped, and while it is in principle possible to use this function to evaluate
% the significance of BDS statistics computed for either of these values, the
% results may not be very reliable, in particular if the number of observations
% is not close to one of the sample sizes for which the BDS distribution is
% tabulated in Kanzler (1998).
%
% Note also that estimation of any BDS statistics evaluated through this function
% must be based on an algorithm which makes use of the most efficient estimators
% of the various correlation integrals on which the BDS statistic is based. The
% author's own function BDS.M may be the only function for which this is true.
% See Kanzler (1998) on why this issue can be crucial to correctly sized estimation
% of the BDS statistic.
%
% Requires the MATLAB Statistics Toolbox.
%
% The author assumes no responsibility for errors or damage resulting from usage. All
% rights reserved. Usage of the programme in applications and alterations of the code
% should be referenced. This script may be redistributed if nothing has been added or
% removed and nothing is charged. Positive or negative feedback would be appreciated.
%
% Copyright (c) 15 Sept. 1998 by Ludwig Kanzler
% Department of Economics, University of Oxford
% Postal: Christ Church, Oxford OX1 1DP, U.K.
% E-mail: ludwig.kanzler@economics.oxford.ac.uk
% Homepage: http://users.ox.ac.uk/~econlrk
% $ Revision: 1.0 $ $ Date: 16 September 1998 $

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Check validity of input arguments %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mcases = 2 : 15;
epscases = [0.5 1.0 1.5 2.0];
ncases = [50 100 250 500 750 1000 2500];
siglevels = [0.005 0.010 0.025 0.050 1 0.050 0.025 0.010 0.005];

if nargin < 1
    error('This function needs some argument input!')
elseif length(w) ~= length(w(:))
    error('Cannot evaluate a matrix of BDS statistics; input must be a scalar or vector.')
elseif nargin < 2
    n = inf;
end

if nargin < 3
    m = 2 : length(w);
elseif unique([mcases, m(:)']) ~= 14
    error('Cannot handle embedding dimension other than integers between 2 and 15.')
elseif length(m) ~= length(w) & length(m) ~= 1

```

```

    error('Cannot handle a vector of embedding dimensions which is not as long as W.')
elseif length(m) == 1
    m = m * ones(1, length(w));
end

if nargin < 4
    eps = 1.5;
elseif ~sum(eps == epscases)
    error('Cannot handle a dimensional distance other than 0.5, 1.0, 1.5 or 2.0.')
end

if n <= 5000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Setup of the quantile look-up table %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The quantile values are taken from Kanzler (1998) and are found along dimension 1
% (with the corresponding values for N(0,1) in parenthesis):
%
%                                     < 0.5% (-2.58)
%                                     < 1.0% (-2.33)
%                                     < 2.5% (-1.96)
%                                     < 5.0% (-1.65)
%                                     > 95.0% ( 1.65)
%                                     > 97.5% ( 1.96)
%                                     > 99.0% ( 2.33)
%                                     > 99.5% ( 2.58)
%
% Embedding dimensions m = [2 3 4 5 6 7 8 9 10 11 12 13 14 15] are along dimension 2.
%
% Sample sizes n = [50 100 250 500 750 1000 2500] are along dimension 3.
%
% Dimensional distances in units of the standard deviation of a normally distributed
% sample eps = [0.5 1.0 1.5 2.0] are along dimension 4.

quants = NaN * ones(8, 14, 7, 4);

% n = 50, eps = 0.5 (c1 ~ 0.27)

quants(1:8, 1:12, 1, 1) = [...
-22.66 -27.66 -31.41 -40.37 -54.26 -48.20 -40.93 -34.41 -28.99 -26.20 -22.25 -22.19
-13.87 -16.90 -19.92 -24.97 -28.85 -25.56 -21.97 -18.59 -16.45 -15.29 -14.53 -14.65
-8.01 -9.87 -12.09 -14.93 -15.79 -13.73 -11.88 -10.34 -9.30 -8.68 -8.71 -8.58
-5.75 -6.94 -8.74 -10.85 -10.85 -9.39 -7.98 -6.89 -6.33 -5.95 -5.81 -5.80
 5.66  6.88  9.14 13.14 18.81 15.41 -0.68 -0.53 -0.40 -0.29 -0.21 -0.15
 8.27 10.05 13.92 20.89 31.30 36.99 -0.44 -0.41 -0.30 -0.22 -0.15 -0.10
13.99 16.09 23.76 34.03 56.16 74.44 73.59 -0.29 -0.22 -0.15 -0.10 -0.07
21.47 27.66 37.27 57.80 89.30 118.37 147.67 45.58 -0.18 -0.12 -0.08 -0.05];

quants(1:8, 13:14, 1, 1) = [...
-22.79 -24.73
-14.99 -16.43
-8.98 -9.75
-5.89 -6.29
-0.11 -0.07
-0.07 -0.05
-0.04 -0.03
-0.03 -0.02];

% n = 50, eps = 1.0 (c1 ~ 0.51)

quants(1:8, 1:14, 1, 2) = [...
-4.66 -5.12 -5.55 -6.05 -6.45 -6.99 -7.46 -7.65 -7.89 -7.72 -8.09 -8.60 -9.37 -10.87
-4.10 -4.42 -4.67 -5.18 -5.49 -5.89 -6.27 -6.45 -6.57 -6.65 -6.69 -6.95 -7.70 -8.25
-3.35 -3.55 -3.78 -4.10 -4.37 -4.58 -4.92 -5.06 -5.13 -5.13 -5.06 -5.25 -5.45 -5.83
-2.84 -2.99 -3.12 -3.32 -3.54 -3.75 -4.01 -4.12 -4.20 -4.16 -4.03 -4.06 -4.17 -4.28
 2.76  2.89  3.03  3.27  3.64  4.11  4.81  5.66  6.60  7.15  6.60  2.59 -0.18 -0.15
 3.50  3.70  3.94  4.35  4.98  5.74  6.81  8.33 10.21 11.97 13.38 12.07  4.40 -0.10
 4.52  4.90  5.25  5.97  6.78  8.21  9.89 12.32 15.44 20.26 24.66 27.84 26.77 16.38
 5.43  5.82  6.46  7.24  8.60 10.65 12.56 16.11 21.04 26.66 34.22 40.93 46.54 46.35];

% n = 50, eps = 1.5 (c1 ~ 0.71)

quants(1:8, 1:14, 1, 3) = [...
-4.09 -4.01 -4.14 -4.08 -4.15 -4.30 -4.38 -4.53 -4.98 -5.32 -5.64 -6.19 -7.05 -7.52
-3.69 -3.65 -3.75 -3.69 -3.75 -3.92 -3.98 -4.12 -4.37 -4.64 -4.85 -5.19 -5.76 -6.35

```

```

-3.08 -3.12 -3.19 -3.19 -3.25 -3.32 -3.36 -3.49 -3.63 -3.74 -3.94 -4.21 -4.46 -4.74
-2.63 -2.67 -2.73 -2.75 -2.80 -2.84 -2.88 -2.97 -3.05 -3.14 -3.24 -3.40 -3.55 -3.74
 2.46  2.46  2.47  2.46  2.44  2.47  2.53  2.60  2.67  2.76  2.85  2.95  3.07  3.20
 3.01  3.04  3.07  3.07  3.14  3.27  3.34  3.48  3.64  3.78  3.96  4.25  4.54  4.87
 3.64  3.69  3.76  3.91  4.05  4.26  4.43  4.62  4.92  5.22  5.65  6.13  6.54  7.09
 4.15  4.11  4.26  4.57  4.70  4.96  5.22  5.56  6.05  6.40  6.90  7.41  8.29  9.19];

% n = 50, eps = 2.0 (c1 ~ 0.84)

quants(1:8, 1:14, 1, 4) = [...
-4.86 -4.77 -4.74 -4.67 -4.83 -4.89 -4.86 -5.14 -5.33 -5.48 -5.73 -6.05 -6.45 -6.89
-4.42 -4.34 -4.35 -4.28 -4.40 -4.48 -4.52 -4.62 -4.78 -4.88 -5.11 -5.32 -5.65 -5.94
-3.69 -3.68 -3.67 -3.69 -3.78 -3.79 -3.86 -3.96 -4.02 -4.09 -4.26 -4.41 -4.63 -4.87
-3.02 -3.09 -3.11 -3.15 -3.23 -3.25 -3.33 -3.38 -3.44 -3.53 -3.60 -3.69 -3.82 -3.98
 2.83  2.80  2.79  2.76  2.72  2.72  2.71  2.71  2.67  2.68  2.65  2.62  2.61  2.62
 3.50  3.42  3.46  3.44  3.44  3.42  3.40  3.40  3.41  3.39  3.44  3.46  3.48  3.52
 4.23  4.26  4.19  4.22  4.27  4.22  4.25  4.24  4.30  4.40  4.48  4.53  4.55  4.74
 4.70  4.77  4.71  4.72  4.81  4.82  4.91  4.93  5.08  5.16  5.21  5.30  5.44  5.53];

% n = 100, eps = 0.5 (c1 ~ 0.27)

quants(1:8, 1:14, 2, 1) = [...
-5.33 -6.39 -7.91 -9.88 -11.20 -9.92 -8.48 -7.07 -6.10 -5.59 -5.07 -4.83 -4.58 -4.41
-4.66 -5.51 -6.74 -8.49 -9.70 -8.72 -7.42 -6.29 -5.49 -4.88 -4.48 -4.14 -3.90 -3.83
-3.78 -4.50 -5.56 -6.90 -8.11 -7.44 -6.27 -5.35 -4.59 -4.04 -3.66 -3.39 -3.20 -3.03
-3.18 -3.71 -4.57 -5.67 -6.93 -6.44 -5.45 -4.62 -3.98 -3.50 -3.14 -2.86 -2.64 -2.50
 3.24  3.83  4.77  6.61  9.80 14.90 20.07 -1.13 -0.96 -0.78 -0.63 -0.50 -0.41 -0.33
 4.14  4.90  6.21  8.75 13.58 21.82 34.20 19.50 -0.82 -0.67 -0.53 -0.43 -0.34 -0.27
 5.49  6.35  8.22 11.79 19.23 33.76 58.23 87.29 -0.61 -0.56 -0.45 -0.35 -0.28 -0.21
 6.49  7.74  9.72 14.34 23.76 43.12 79.06 130.73 137.15 -0.47 -0.39 -0.30 -0.24 -0.18];

% n = 100, eps = 1.0 (c1 ~ 0.52)

quants(1:8, 1:14, 2, 2) = [...
-3.16 -3.23 -3.32 -3.42 -3.66 -3.90 -4.20 -4.41 -4.57 -4.57 -4.40 -4.27 -4.14 -4.03
-2.88 -2.93 -3.00 -3.12 -3.27 -3.50 -3.72 -3.97 -4.10 -4.12 -4.01 -3.85 -3.73 -3.63
-2.47 -2.52 -2.58 -2.66 -2.80 -2.93 -3.14 -3.34 -3.49 -3.52 -3.42 -3.30 -3.18 -3.06
-2.12 -2.16 -2.21 -2.29 -2.40 -2.53 -2.69 -2.87 -3.03 -3.07 -3.03 -2.90 -2.76 -2.64
 2.16  2.20  2.28  2.41  2.60  2.80  3.13  3.61  4.18  5.01  6.01  7.12  7.81  7.05
 2.64  2.73  2.85  3.05  3.28  3.67  4.16  4.84  5.68  6.96  8.84 10.88 12.77 14.23
 3.25  3.37  3.56  3.87  4.18  4.77  5.42  6.40  7.74  9.92 12.57 16.04 21.06 26.29
 3.72  3.81  4.06  4.39  4.94  5.58  6.37  7.58  9.39 12.14 15.70 20.87 27.11 35.25];

% n = 100, eps = 1.5 (c1 ~ 0.71)

quants(1:8, 1:14, 2, 3) = [...
-3.15 -3.12 -3.15 -3.14 -3.10 -3.14 -3.08 -3.13 -3.15 -3.23 -3.21 -3.27 -3.38 -3.49
-2.88 -2.88 -2.87 -2.87 -2.86 -2.86 -2.86 -2.85 -2.89 -2.95 -2.95 -2.99 -3.03 -3.15
-2.45 -2.48 -2.44 -2.47 -2.49 -2.49 -2.48 -2.48 -2.50 -2.50 -2.53 -2.56 -2.60 -2.63
-2.09 -2.11 -2.13 -2.12 -2.15 -2.16 -2.17 -2.16 -2.16 -2.17 -2.19 -2.22 -2.25 -2.28
 2.02  2.02  2.00  2.03  2.02  2.05  2.06  2.13  2.18  2.24  2.31  2.39  2.50  2.62
 2.47  2.48  2.45  2.49  2.54  2.59  2.65  2.72  2.82  2.94  3.09  3.25  3.41  3.60
 2.98  3.01  3.03  3.07  3.16  3.23  3.31  3.48  3.64  3.87  4.10  4.35  4.64  5.02
 3.27  3.41  3.45  3.50  3.61  3.72  3.86  4.05  4.26  4.46  4.84  5.16  5.56  6.10];

% n = 100, eps = 2.0 (c1 ~0.84)

quants(1:8, 1:14, 2, 4) = [...
-3.66 -3.60 -3.56 -3.52 -3.43 -3.44 -3.43 -3.47 -3.46 -3.48 -3.48 -3.56 -3.56 -3.62
-3.28 -3.24 -3.25 -3.25 -3.17 -3.14 -3.15 -3.16 -3.18 -3.20 -3.16 -3.24 -3.26 -3.31
-2.73 -2.75 -2.77 -2.81 -2.76 -2.73 -2.74 -2.72 -2.74 -2.76 -2.76 -2.80 -2.81 -2.82
-2.28 -2.33 -2.37 -2.38 -2.37 -2.36 -2.37 -2.37 -2.38 -2.37 -2.39 -2.41 -2.43 -2.44
 2.26  2.22  2.17  2.17  2.17  2.15  2.14  2.14  2.14  2.13  2.13  2.13  2.14  2.14
 2.77  2.71  2.69  2.67  2.66  2.66  2.67  2.72  2.72  2.71  2.72  2.75  2.77  2.78
 3.34  3.35  3.32  3.32  3.32  3.34  3.37  3.39  3.40  3.44  3.48  3.56  3.59  3.60
 3.73  3.74  3.76  3.78  3.74  3.78  3.84  3.92  3.93  4.00  4.04  4.07  4.10  4.23];

% n = 250, eps = 0.5 (c1 ~ 0.27)

quants(1:8, 1:14, 3, 1) = [...
-3.25 -3.64 -4.32 -5.44 -6.76 -8.03 -7.35 -6.27 -5.29 -4.57 -4.01 -3.58 -3.22 -2.90
-2.94 -3.32 -3.91 -4.85 -6.19 -7.40 -6.95 -5.91 -5.02 -4.33 -3.77 -3.35 -3.02 -2.72
-2.54 -2.86 -3.29 -4.11 -5.19 -6.55 -6.30 -5.40 -4.59 -3.95 -3.46 -3.05 -2.71 -2.45
-2.18 -2.40 -2.79 -3.47 -4.45 -5.78 -5.83 -5.00 -4.25 -3.65 -3.17 -2.80 -2.48 -2.23

```

```

2.27 2.50 2.98 3.79 5.19 7.68 12.18 18.51 13.10 -1.69 -1.44 -1.23 -1.05 -0.90
2.81 3.13 3.75 4.78 6.63 10.09 16.49 28.55 43.94 -1.52 -1.34 -1.13 -0.96 -0.82
3.49 3.90 4.70 6.00 8.37 13.11 22.93 42.19 73.31 98.79 -1.20 -1.03 -0.87 -0.74
3.96 4.49 5.42 7.01 9.81 15.49 28.61 54.34 102.44 165.50 -1.08 -0.94 -0.82 -0.69];

% n = 250, eps = 1.0 (c1 ~ 0.52)

quants(1:8, 1:14, 3, 2) = [...
-2.63 -2.57 -2.58 -2.61 -2.67 -2.70 -2.81 -2.96 -3.12 -3.32 -3.51 -3.53 -3.44 -3.24
-2.42 -2.38 -2.39 -2.40 -2.44 -2.48 -2.59 -2.72 -2.87 -3.06 -3.28 -3.33 -3.24 -3.08
-2.09 -2.11 -2.09 -2.10 -2.13 -2.18 -2.24 -2.37 -2.51 -2.71 -2.91 -3.01 -2.99 -2.83
-1.81 -1.83 -1.83 -1.84 -1.86 -1.90 -1.97 -2.06 -2.20 -2.37 -2.57 -2.73 -2.76 -2.63
1.83 1.88 1.88 1.95 2.03 2.15 2.25 2.45 2.71 3.05 3.56 4.21 5.05 6.33
2.26 2.30 2.35 2.44 2.54 2.69 2.88 3.12 3.49 3.93 4.68 5.60 6.99 8.82
2.76 2.82 2.93 3.01 3.18 3.40 3.78 4.11 4.58 5.21 6.22 7.76 9.79 12.75
3.08 3.15 3.31 3.47 3.70 3.97 4.39 4.91 5.38 6.49 7.70 9.43 12.14 15.87];

% n = 250, eps = 1.5 (c1 ~ 0.71)

quants(1:8, 1:14, 3, 3) = [...
-2.72 -2.70 -2.67 -2.63 -2.60 -2.55 -2.51 -2.49 -2.49 -2.45 -2.46 -2.44 -2.43 -2.43
-2.48 -2.49 -2.46 -2.43 -2.40 -2.37 -2.32 -2.31 -2.29 -2.29 -2.27 -2.26 -2.25 -2.25
-2.13 -2.12 -2.14 -2.13 -2.10 -2.08 -2.04 -2.02 -2.03 -2.00 -1.98 -1.99 -1.99 -1.99
-1.81 -1.81 -1.85 -1.84 -1.82 -1.81 -1.79 -1.77 -1.77 -1.75 -1.76 -1.75 -1.75 -1.74
1.82 1.82 1.82 1.84 1.85 1.87 1.89 1.92 1.94 1.98 2.02 2.06 2.11 2.16
2.22 2.22 2.25 2.25 2.29 2.33 2.35 2.40 2.44 2.52 2.55 2.63 2.73 2.82
2.68 2.67 2.74 2.77 2.84 2.90 2.94 3.02 3.09 3.15 3.28 3.46 3.59 3.73
2.97 3.01 3.09 3.12 3.19 3.27 3.28 3.41 3.52 3.66 3.85 3.97 4.20 4.42];

% n = 250, eps = 2.0 (c1 ~ 0.84)

quants(1:8, 1:14, 3, 4) = [...
-2.90 -2.88 -2.87 -2.86 -2.89 -2.83 -2.77 -2.74 -2.76 -2.73 -2.68 -2.68 -2.66 -2.62
-2.62 -2.64 -2.63 -2.61 -2.59 -2.59 -2.56 -2.51 -2.52 -2.50 -2.49 -2.48 -2.45 -2.43
-2.25 -2.26 -2.24 -2.24 -2.24 -2.23 -2.22 -2.22 -2.19 -2.18 -2.18 -2.17 -2.15 -2.13
-1.91 -1.92 -1.93 -1.93 -1.93 -1.94 -1.93 -1.93 -1.91 -1.91 -1.89 -1.89 -1.89 -1.88
1.88 1.86 1.86 1.86 1.85 1.83 1.83 1.84 1.85 1.85 1.85 1.87 1.88 1.88
2.30 2.29 2.29 2.28 2.27 2.29 2.28 2.28 2.29 2.29 2.31 2.33 2.35 2.36
2.79 2.78 2.75 2.78 2.80 2.79 2.81 2.83 2.83 2.81 2.84 2.88 2.92 2.93
3.16 3.15 3.09 3.07 3.13 3.17 3.14 3.18 3.20 3.24 3.22 3.24 3.32 3.36];

% n = 500, eps = 0.5 (c1 ~ 0.28)

quants(1:8, 1:14, 4, 1) = [...
-2.88 -3.01 -3.35 -4.01 -5.05 -6.40 -7.60 -6.95 -5.99 -5.12 -4.43 -3.88 -3.43 -3.07
-2.63 -2.78 -3.04 -3.67 -4.52 -5.92 -7.22 -6.69 -5.74 -4.93 -4.28 -3.74 -3.31 -2.95
-2.22 -2.38 -2.62 -3.09 -3.87 -5.15 -6.55 -6.33 -5.44 -4.66 -4.03 -3.54 -3.11 -2.77
-1.89 -2.01 -2.24 -2.64 -3.32 -4.38 -5.78 -6.00 -5.20 -4.46 -3.85 -3.35 -2.95 -2.63
2.01 2.13 2.37 2.87 3.68 5.25 8.05 13.57 22.54 31.55 -2.28 -2.00 -1.73 -1.51
2.46 2.64 2.92 3.55 4.55 6.61 10.41 17.88 31.94 49.71 -2.08 -1.89 -1.64 -1.43
2.95 3.27 3.72 4.41 5.74 8.50 13.55 23.17 44.46 84.82 133.43 -1.73 -1.53 -1.33
3.32 3.61 4.14 5.03 6.58 9.75 15.80 27.76 55.17 109.55 198.94 264.86 -1.45 -1.27];

% n = 500, eps = 1.0 (c1 ~ 0.52)

quants(1:8, 1:14, 4, 2) = [...
-2.55 -2.52 -2.53 -2.51 -2.46 -2.46 -2.48 -2.54 -2.65 -2.80 -3.01 -3.24 -3.43 -3.47
-2.34 -2.33 -2.30 -2.30 -2.27 -2.27 -2.30 -2.36 -2.47 -2.61 -2.78 -3.01 -3.22 -3.31
-2.01 -2.00 -1.99 -1.96 -1.99 -2.00 -2.03 -2.07 -2.16 -2.28 -2.46 -2.66 -2.91 -3.04
-1.72 -1.72 -1.71 -1.70 -1.73 -1.75 -1.76 -1.81 -1.88 -1.98 -2.14 -2.35 -2.59 -2.80
1.77 1.77 1.77 1.81 1.87 1.93 2.01 2.12 2.25 2.47 2.76 3.16 3.64 4.34
2.14 2.15 2.20 2.26 2.34 2.43 2.55 2.71 2.95 3.21 3.57 4.06 4.74 5.71
2.58 2.62 2.64 2.76 2.89 3.04 3.23 3.47 3.79 4.15 4.58 5.23 6.31 7.75
2.89 2.95 3.00 3.10 3.28 3.50 3.68 4.02 4.41 4.84 5.51 6.24 7.34 9.28];

% n = 500, eps = 1.5 (c1 ~ 0.71)

quants(1:8, 1:14, 4, 3) = [...
-2.61 -2.54 -2.54 -2.54 -2.48 -2.45 -2.44 -2.41 -2.37 -2.33 -2.29 -2.29 -2.28 -2.25
-2.37 -2.35 -2.31 -2.30 -2.26 -2.25 -2.23 -2.22 -2.19 -2.17 -2.14 -2.12 -2.10 -2.09
-2.01 -2.02 -2.00 -1.98 -1.98 -1.96 -1.94 -1.93 -1.91 -1.89 -1.89 -1.88 -1.87 -1.85
-1.71 -1.71 -1.71 -1.71 -1.69 -1.69 -1.69 -1.67 -1.67 -1.66 -1.65 -1.65 -1.64 -1.63
1.74 1.72 1.73 1.74 1.74 1.74 1.76 1.76 1.79 1.82 1.85 1.89 1.94 1.98
2.10 2.09 2.12 2.11 2.13 2.15 2.17 2.20 2.23 2.28 2.33 2.40 2.45 2.51

```

```

2.50 2.54 2.52 2.57 2.60 2.63 2.65 2.71 2.75 2.85 2.88 2.98 3.09 3.21
2.80 2.86 2.86 2.94 2.93 2.99 3.04 3.05 3.14 3.25 3.33 3.46 3.53 3.68];

% n = 500, eps = 2.0 (c1 ~ 0.84)

quants(1:8, 1:14, 4, 4) = [...
-2.65 -2.65 -2.70 -2.63 -2.62 -2.63 -2.63 -2.64 -2.60 -2.55 -2.52 -2.49 -2.46 -2.46
-2.43 -2.43 -2.45 -2.43 -2.39 -2.40 -2.40 -2.39 -2.40 -2.37 -2.34 -2.31 -2.30 -2.28
-2.09 -2.08 -2.08 -2.09 -2.08 -2.07 -2.07 -2.06 -2.07 -2.05 -2.04 -2.02 -2.01 -1.98
-1.76 -1.78 -1.79 -1.79 -1.79 -1.78 -1.78 -1.78 -1.78 -1.78 -1.78 -1.76 -1.74 -1.73
1.78 1.76 1.74 1.73 1.74 1.75 1.75 1.74 1.74 1.74 1.75 1.77 1.77 1.78
2.13 2.14 2.14 2.13 2.12 2.11 2.10 2.10 2.12 2.15 2.16 2.18 2.19 2.20
2.58 2.57 2.58 2.58 2.55 2.57 2.57 2.56 2.60 2.63 2.66 2.69 2.71 2.75
2.91 2.87 2.88 2.87 2.88 2.87 2.88 2.88 2.92 2.92 2.95 2.99 2.99 3.06 3.10];

% n = 750, eps = 1.0 (c1 ~ 0.52)

quants(1:8, 1:14, 5, 2) = [...
-2.52 -2.53 -2.49 -2.48 -2.49 -2.45 -2.44 -2.47 -2.51 -2.61 -2.78 -2.96 -3.24 -3.46
-2.32 -2.31 -2.30 -2.28 -2.29 -2.25 -2.23 -2.28 -2.32 -2.40 -2.55 -2.76 -3.03 -3.24
-1.98 -1.97 -1.98 -1.97 -1.96 -1.94 -1.95 -1.97 -2.04 -2.11 -2.23 -2.42 -2.66 -2.91
-1.68 -1.69 -1.69 -1.69 -1.69 -1.69 -1.70 -1.73 -1.77 -1.83 -1.96 -2.13 -2.34 -2.59
1.72 1.74 1.76 1.79 1.80 1.87 1.91 2.01 2.13 2.29 2.49 2.74 3.13 3.70
2.08 2.13 2.15 2.21 2.27 2.35 2.43 2.52 2.68 2.90 3.17 3.55 4.03 4.71
2.48 2.59 2.65 2.69 2.78 2.88 3.00 3.21 3.46 3.64 4.02 4.53 5.16 6.21
2.78 2.91 2.99 3.09 3.16 3.27 3.50 3.73 3.94 4.22 4.70 5.22 6.06 7.30];

% n = 750, eps = 1.5 (c1 ~ 0.71)

quants(1:8, 1:14, 5, 3) = [...
-2.57 -2.56 -2.54 -2.53 -2.50 -2.45 -2.40 -2.39 -2.36 -2.34 -2.31 -2.28 -2.26 -2.23
-2.34 -2.32 -2.33 -2.30 -2.29 -2.27 -2.24 -2.21 -2.17 -2.16 -2.14 -2.10 -2.09 -2.07
-2.01 -1.99 -1.98 -1.98 -1.97 -1.96 -1.95 -1.93 -1.91 -1.90 -1.87 -1.85 -1.84 -1.82
-1.68 -1.69 -1.69 -1.69 -1.69 -1.71 -1.69 -1.68 -1.67 -1.65 -1.65 -1.63 -1.61 -1.60 -1.60
1.69 1.68 1.68 1.69 1.71 1.72 1.73 1.74 1.75 1.78 1.79 1.80 1.84 1.87
2.02 2.03 2.03 2.06 2.07 2.11 2.12 2.17 2.19 2.23 2.25 2.28 2.32 2.39
2.48 2.50 2.48 2.50 2.54 2.58 2.61 2.63 2.69 2.75 2.84 2.93 2.95 3.04
2.79 2.78 2.77 2.80 2.88 2.91 2.97 3.02 3.08 3.16 3.29 3.33 3.43 3.56];

% n = 750, eps = 2.0 (c1 ~ 0.84)

quants(1:8, 1:14, 5, 4) = [...
-2.66 -2.64 -2.61 -2.65 -2.58 -2.57 -2.55 -2.55 -2.53 -2.51 -2.51 -2.50 -2.49 -2.46
-2.40 -2.40 -2.40 -2.40 -2.37 -2.36 -2.36 -2.36 -2.35 -2.34 -2.31 -2.30 -2.29 -2.25
-2.03 -2.03 -2.05 -2.04 -2.03 -2.03 -2.03 -2.04 -2.04 -2.02 -2.00 -1.99 -1.97 -1.97
-1.72 -1.72 -1.74 -1.74 -1.73 -1.73 -1.73 -1.73 -1.74 -1.72 -1.72 -1.72 -1.71 -1.70
1.75 1.71 1.71 1.69 1.70 1.69 1.69 1.69 1.70 1.71 1.71 1.72 1.73 1.75
2.09 2.09 2.09 2.05 2.05 2.04 2.04 2.06 2.08 2.08 2.09 2.12 2.13 2.14
2.53 2.54 2.49 2.47 2.47 2.48 2.48 2.52 2.55 2.57 2.59 2.60 2.62 2.64
2.86 2.81 2.77 2.78 2.75 2.73 2.77 2.80 2.88 2.88 2.91 2.95 2.97 2.99];

% n = 1000, eps = 0.5 (c1 ~ 0.28)

quants(1:8, 1:14, 6, 1) = [...
-2.66 -2.72 -2.83 -3.17 -3.82 -4.92 -6.63 -8.03 -7.33 -6.36 -5.50 -4.79 -4.21 -3.73
-2.44 -2.48 -2.59 -2.91 -3.51 -4.49 -6.10 -7.69 -7.15 -6.19 -5.35 -4.66 -4.10 -3.62
-2.08 -2.11 -2.24 -2.50 -3.01 -3.90 -5.30 -7.04 -6.85 -5.97 -5.16 -4.49 -3.94 -3.49
-1.77 -1.79 -1.92 -2.16 -2.58 -3.31 -4.59 -6.15 -6.60 -5.76 -4.99 -4.34 -3.81 -3.37
1.87 1.95 2.10 2.36 2.83 3.81 5.53 8.69 14.19 24.40 36.34 -3.02 -2.66 -2.33
2.26 2.37 2.54 2.88 3.46 4.69 6.81 10.86 18.53 32.76 49.56 -2.85 -2.56 -2.25
2.72 2.86 3.12 3.54 4.31 5.75 8.59 13.50 24.27 46.74 89.25 137.59 -2.44 -2.16
3.04 3.25 3.52 4.03 4.82 6.53 9.84 16.19 28.67 56.34 115.13 190.18 -2.25 -2.10];

% n = 1000, eps = 1.0 (c1 ~ 0.52)

quants(1:8, 1:14, 6, 2) = [...
-2.52 -2.53 -2.47 -2.45 -2.43 -2.42 -2.40 -2.40 -2.44 -2.52 -2.65 -2.87 -3.07 -3.33
-2.28 -2.29 -2.26 -2.22 -2.20 -2.22 -2.21 -2.21 -2.25 -2.34 -2.48 -2.63 -2.86 -3.11
-1.96 -1.96 -1.93 -1.94 -1.92 -1.90 -1.92 -1.93 -1.97 -2.06 -2.15 -2.31 -2.51 -2.77
-1.68 -1.67 -1.66 -1.65 -1.65 -1.65 -1.65 -1.68 -1.71 -1.80 -1.88 -2.00 -2.20 -2.44
1.68 1.70 1.72 1.75 1.78 1.82 1.88 1.95 2.04 2.12 2.28 2.50 2.79 3.19
2.03 2.07 2.11 2.15 2.19 2.25 2.31 2.40 2.54 2.66 2.87 3.13 3.52 4.07
2.43 2.50 2.56 2.57 2.66 2.73 2.82 3.00 3.15 3.35 3.62 3.94 4.44 5.20
2.74 2.78 2.86 2.89 2.99 3.13 3.25 3.39 3.64 3.84 4.25 4.63 5.21 5.94];

```

```
% n = 1000, eps = 1.5 (c1 ~ 0.71)
```

```
quants(1:8, 1:14, 6, 3) = [...
-2.53 -2.54 -2.50 -2.50 -2.47 -2.46 -2.44 -2.41 -2.38 -2.34 -2.31 -2.27 -2.26 -2.23
-2.33 -2.34 -2.30 -2.28 -2.27 -2.24 -2.22 -2.21 -2.19 -2.17 -2.15 -2.12 -2.09 -2.07
-2.00 -1.99 -2.01 -1.99 -1.97 -1.95 -1.94 -1.92 -1.92 -1.91 -1.89 -1.87 -1.86 -1.83
-1.69 -1.71 -1.71 -1.71 -1.71 -1.71 -1.67 -1.67 -1.66 -1.66 -1.65 -1.63 -1.63 -1.62 -1.61
1.68 1.69 1.68 1.68 1.70 1.71 1.73 1.73 1.73 1.74 1.75 1.77 1.79 1.81 1.83
2.01 2.02 2.03 2.04 2.06 2.08 2.12 2.16 2.16 2.19 2.22 2.25 2.27 2.31
2.43 2.46 2.45 2.48 2.49 2.53 2.55 2.58 2.65 2.69 2.73 2.78 2.84 2.90
2.70 2.74 2.78 2.81 2.82 2.87 2.89 2.89 2.93 3.01 3.07 3.16 3.23 3.30];
```

```
% n = 1000, eps = 2.0 (c1 ~ 0.84)
```

```
quants(1:8, 1:14, 6, 4) = [...
-2.70 -2.65 -2.62 -2.59 -2.58 -2.57 -2.58 -2.60 -2.56 -2.53 -2.51 -2.49 -2.47 -2.46
-2.40 -2.41 -2.39 -2.37 -2.37 -2.37 -2.35 -2.36 -2.35 -2.33 -2.30 -2.28 -2.26 -2.25
-2.05 -2.04 -2.04 -2.05 -2.04 -2.04 -2.05 -2.04 -2.03 -2.02 -2.00 -2.00 -1.98 -1.96
-1.73 -1.72 -1.74 -1.74 -1.73 -1.74 -1.74 -1.74 -1.73 -1.72 -1.73 -1.73 -1.71 -1.71
1.70 1.68 1.69 1.69 1.69 1.69 1.69 1.69 1.69 1.70 1.71 1.72 1.73 1.73
2.04 2.04 2.04 2.05 2.04 2.04 2.06 2.07 2.07 2.07 2.08 2.08 2.09 2.11
2.45 2.47 2.45 2.47 2.46 2.47 2.51 2.51 2.50 2.52 2.52 2.55 2.57 2.58
2.77 2.76 2.80 2.79 2.77 2.76 2.80 2.82 2.83 2.85 2.88 2.87 2.90 2.93];
```

```
% n = 2500, eps = 0.5 (c1 ~ 0.28)
```

```
quants(1:8, 1:14, 7, 1) = [...
-2.51 -2.53 -2.59 -2.74 -3.00 -3.67 -4.78 -6.51 -8.74 -8.85 -7.73 -6.77 -5.93 -5.24
-2.31 -2.33 -2.37 -2.55 -2.77 -3.35 -4.41 -6.00 -8.28 -8.68 -7.63 -6.66 -5.84 -5.16
-1.97 -1.99 -2.06 -2.16 -2.39 -2.89 -3.82 -5.23 -7.22 -8.41 -7.45 -6.51 -5.71 -5.05
-1.69 -1.70 -1.74 -1.85 -2.04 -2.46 -3.21 -4.52 -6.35 -8.19 -7.29 -6.38 -5.60 -4.93
1.73 1.75 1.79 1.94 2.18 2.66 3.53 5.18 8.20 13.75 24.50 32.59 -4.41 -3.95
2.07 2.11 2.21 2.36 2.68 3.19 4.27 6.32 10.24 17.79 31.87 60.12 95.19 -3.85
2.45 2.54 2.69 2.93 3.22 3.93 5.28 7.70 12.65 22.49 43.30 83.54 131.83 -3.66
2.78 2.85 3.02 3.26 3.60 4.36 5.93 8.69 14.35 26.23 52.23 105.70 216.13 339.65];
```

```
% n = 2500, eps = 1.0 (c1 ~ 0.52)
```

```
quants(1:8, 1:14, 7, 2) = [...
-2.58 -2.52 -2.47 -2.46 -2.42 -2.40 -2.40 -2.38 -2.42 -2.44 -2.47 -2.50 -2.61 -2.81
-2.35 -2.27 -2.26 -2.24 -2.22 -2.20 -2.20 -2.22 -2.22 -2.22 -2.25 -2.31 -2.41 -2.63
-1.97 -1.94 -1.91 -1.91 -1.90 -1.91 -1.90 -1.89 -1.89 -1.90 -1.92 -1.97 -2.02 -2.11 -2.29
-1.66 -1.66 -1.64 -1.63 -1.63 -1.64 -1.64 -1.64 -1.63 -1.62 -1.65 -1.68 -1.74 -1.83 -1.99
1.67 1.70 1.69 1.71 1.75 1.76 1.78 1.79 1.83 1.88 1.95 2.07 2.21 2.39
2.02 2.06 2.08 2.10 2.12 2.16 2.18 2.20 2.24 2.31 2.39 2.53 2.72 2.97
2.47 2.50 2.53 2.55 2.54 2.57 2.66 2.74 2.83 2.88 3.02 3.17 3.35 3.77
2.78 2.77 2.81 2.88 2.89 2.91 3.04 3.13 3.23 3.25 3.43 3.68 3.99 4.31];
```

```
% n = 2500, eps = 1.5 (c1 ~ 0.71)
```

```
quants(1:8, 1:14, 7, 3) = [...
-2.55 -2.54 -2.53 -2.49 -2.50 -2.45 -2.43 -2.42 -2.40 -2.40 -2.39 -2.36 -2.36 -2.32
-2.34 -2.32 -2.31 -2.29 -2.27 -2.23 -2.21 -2.22 -2.20 -2.18 -2.16 -2.16 -2.15 -2.13
-1.98 -1.96 -1.95 -1.94 -1.94 -1.92 -1.92 -1.92 -1.90 -1.89 -1.88 -1.88 -1.86 -1.84
-1.65 -1.66 -1.66 -1.64 -1.64 -1.63 -1.64 -1.64 -1.63 -1.63 -1.63 -1.62 -1.61 -1.59
1.60 1.60 1.61 1.63 1.64 1.64 1.65 1.67 1.67 1.68 1.69 1.71 1.72 1.75
1.95 1.91 1.92 1.93 1.95 1.98 2.00 2.01 2.02 2.04 2.07 2.11 2.14 2.17
2.36 2.34 2.34 2.34 2.32 2.35 2.36 2.38 2.38 2.42 2.47 2.54 2.58 2.63
2.59 2.60 2.56 2.61 2.58 2.59 2.62 2.67 2.73 2.74 2.77 2.80 2.84 2.89];
```

```
% n = 2500, eps = 2.0 (c1 ~ 0.84)
```

```
quants(1:8, 1:14, 7, 4) = [...
-2.56 -2.57 -2.55 -2.54 -2.56 -2.56 -2.54 -2.53 -2.51 -2.51 -2.50 -2.47 -2.46 -2.44
-2.33 -2.33 -2.32 -2.30 -2.32 -2.33 -2.31 -2.29 -2.28 -2.25 -2.26 -2.26 -2.24 -2.24
-1.96 -2.00 -2.00 -1.99 -1.98 -1.98 -1.98 -1.98 -1.96 -1.96 -1.95 -1.94 -1.92 -1.91
-1.67 -1.70 -1.70 -1.70 -1.70 -1.70 -1.71 -1.71 -1.71 -1.69 -1.68 -1.67 -1.66 -1.65
1.69 1.70 1.69 1.68 1.68 1.69 1.69 1.69 1.70 1.71 1.71 1.71 1.70 1.71 1.72
2.06 2.08 2.03 2.01 2.03 2.03 2.04 2.07 2.07 2.07 2.06 2.07 2.07 2.07
2.47 2.44 2.45 2.45 2.44 2.42 2.45 2.42 2.44 2.45 2.45 2.47 2.51 2.49
2.72 2.71 2.76 2.74 2.74 2.69 2.70 2.71 2.70 2.72 2.76 2.81 2.78 2.79];
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Look-up of the appropriate quantiles %%%%%%%%%
% Determine in between which two sample sizes tabulated the current n lies:

lower =      sum(n >= ncases);
upper = 8 - sum(n <= ncases);

% Fix some special cases; for samples of less than 50, use the values for 50:

if lower == 0
    lower = 1;

% and since there are no tabulated values for n = 750, eps = 0.5, reference to the
% corresponding part of the quantile table must be avoided:

elseif eps == 0.5
    if lower == 5
        lower = 4;
    end
    if upper == 5
        upper = 6;
    end
end

% Determine the significance level in turn for each BDS statistic contained in W:
for i = 1 : length(w)

    % Find the eight quantile values each for the lower and upper sample sizes:

    lowerqus = reshape(quants(1:8, m(i)-1, lower, eps*2), 8, 1);

    if n <= 2500
        upperqus = reshape(quants(1:8, m(i)-1, upper, eps*2), 8, 1);

    else % i.e. approaching standard normality:
        upperqus = norminv(siglevels([1:4 6:9]));
        ncases = [ncases 5000];
    end

    % Interpolate the quantile values for the actual sample size from the quantile
    % values of the surrounding sample sizes; note that this method may slightly
    % increase the size of a type I error for sample sizes which are not close to one
    % of the tabulated cases; this problem could be mitigated by a response surface
    % yet to be developed.

    if lower ~= upper
        qus = lowerqus + (upperqus - lowerqus) * (n - ncases(lower)) /...
            (ncases(upper) - ncases(lower));
    else
        qus = lowerqus;
    end

    % Find the matching significance levels; at least one of the terms must be 1, or
    % both, so their product yields the overall one-sided significance level:

    sig(i) = siglevels(5 - sum(w(i)<=qus(1:4))) * siglevels(5 + sum(w(i)>=qus(5:8)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Otherwise use standard-normal look-up %%%%%%%%%
else
    qus = [norminv(siglevels(1:4)) norminv(1 - siglevels(6:9))];
    for i = 1 : length(w)
        sig(i) = siglevels(5 - sum(w(i)<=qus(1:4))) * siglevels(5 + sum(w(i)>=qus(5:8)));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% REFERENCES:
%
% Kanzler, Ludwig (1998), "Very Fast and Correctly Sized Estimation of the BDS Statistic",
% Oxford University, Department of Economics, working paper, available on
% http://users.ox.ac.uk/~econlrk

% End of file.

```