CMAT Newsletter: December 2009

Wolfgang M. Hartmann

December 2009

Contents

1 General Remarks

A number of excellent methods for analyzing time series data, like EKG or EEG data, are based on the excellent *Oxford MFE Toolbox* written for Matlab users by Kevin Sheppard. We have modified some of his methods and used some of his ideas, especially for the reparametrization of constraints with the GARCH methods. With his permission we also use Kevin's tables of critical values for the (augmented) Dickey-Fuller testing.

1.1 New Functions

arhetero Heterogeneous Auto-Regressive modeling (arhetero);

arma Auto-Regressive Moving-Average modeling (arma);

armafore AutoRegressive Moving-Average forecasting (armafore);

affrma RMA (Robust Multichip Average) by Bolstad et.al (2003) for the normalization of microarray data

berkow implements Berkowitz test of time series and cross sectional data for many data distributions (berkow);

garch implements (G)ARCH, TARCH, AVARCH, ZARCH, EGARCH, APARCH, AGARCH, NAGARCH, IGARCH, FIGARCH time series modeling (based on MFE Toolbox) (garch);

jarbera implements the Jarque-Bera test for normality based on the skewness and kurtosis of the data (jarbera);

mucomp comparison of means, confirmatory ANOVA similar to Kuiper, Klugkist, & Hoijtink (2007) (mucomp);

shapwilk implements Shapiro-Wilk and Shapiro-Francia tests for normal distributed data (shapwilk).

1.2 Fixed Bugs

2 Modifications of Features

2.1 Modification of the `kstest` Function

The old version was able only for testing uniform and normal distributions of data. The new version permits also testing of a large variety of other distributions, optional delivers some printed output, and returns additional arguments. Here is the new document for this function:

---

prob = kstest(y<,optn<,"dist"<,...>>)

< prob,stat > = kstest(y<,optn<,"dist"<,...>>)

Purpose: Different forms of the Cramer-vonMises-Kolmogorov-Smirnov test for a large variety of data distributions, including uniform and normal, are implemented.

Input: y The first input object y must be either a n vector or a $n \times nc$ matrix. If y is a vector the scalar value of the probability of the CDF is returned, if y is a matrix then a vector of nc probabilities is returned by using each column of y separately.

optn this is a vector of options which should be initialized with missing values for defaults:

1. (int) amount of printed output (default is no output);
2. (real) probability for test (default is 0.05), must be in $[0, 1]$;
3. (int) version number of algorithm (default is 0):

0 the default version, for all implemented distributions (see below), returning all three results, probability, statistics, and hypothesis value.

1 Anderson-Darling approximative, only for uniform distribution (i.e. data in $[0, 1]$) which returns the probability only.

2 Marsaglia-Tsang-Wang exact (JSS,2003), only for uniform distribution (i.e. data in $[0, 1]$) which returns the probability only.

3 the old implementation of the KS-Lilliefors method, only for normal distribbution with $\mu = 0$ and $\sigma = 1$ and returning the probability only.

"dist" The following string options are valid:

("bern",x,prob) Bernoulli distribution

- `x` is the upper limit for cumulation $0 \leq x \leq 1$.
- `p` is the probability for success of 1, $0 \leq prob \leq 1$.

("beta",x,a,b<, <l>, <u>>) Beta distribution

- `x` is the upper limit for integration $0 \leq x \leq 1$.
- `a` is the first shape parameter $a \geq 0$.
- `b` is the second shape parameter $b \geq 0$.

("bin",s,prob,n) Binomial distribution

- `s` is the number of successes, $0 \leq s \leq n$.
- `prob` is the probability for success, $0 \leq prob \leq 1$.
- `n` is the number of independend trials, $n > 0$.

("cau",x<, < μ >, < σ >>) Cauchy distribution

- `x` is the upper limit for integration $x > 0$.
- μ location parameter $mu \geq 0$.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("chis",x,df<,nc>) (noncentral) ChiSquare distribution

- `x` is the upper limit for integration $x > 0$.
- `df` degrees of freedom $df > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("expo",x<,σ >) Exponential distribution

- `x` is the upper limit for integration $0 \leq x \leq 1$.
- σ scale parameter ($\sigma > 0$).

Default is $\sigma = 1$.

("f",x,ndf,ddf<,nc>) (noncentral) F distribution

- `x` is the upper limit for integration $x > 0$.
- `ndf` numerator degrees of freedom $ndf > 0$.
- `ddf` denominator degrees of freedom $ddf > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("gam",x,shape<,scale>)

- `x` is the upper limit for integration $x > 0$.
- `shape` shape parameter $shape > 0$.
- `scale` is the scale parameter $scale > 0$.

Default is $scale = 1$.

("gaus",x<,< μ >,< σ >>) Gauss (Normal) distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("geom",m,p)

- `m` is the upper limit for cumulation $m > 0$.
- `p` is the probability for success of 1, $0 \leq prob \leq 1$.

("hypg",x,m,k,n<,r>) Hypergeometric distribution

- `x` is the upper limit for cumulation $x > 0$.
- `m` is the population size ($m \geq 1$.
- `k` is the number of items ($0 \leq k \leq m$.
- `n` is the sample size ($0 \leq n \leq m$.
- `r` is an optional odds ratio ($r > 0$.

Default is $r = 1$.

("igau",x,d) Inverse Gauss (Wald) distribution

- `x` is the upper limit for integration $x > 0$.
- `d` shape parameter $d > 0$.

("lapl",x<,< μ >,< σ >>) Laplace distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("logi",x<,< μ >,< σ >>) • `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("logn",x<,< μ >,< σ >>) LogNormal distribution
- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("negb",x,prob,n) • `x` is the number of failures, $0 \leq f \leq n$.
- `prob` is the probability for success, $0 \leq prob \leq 1$.
- `n` is the number of successes, $n > 0$.

("norm",x<,< μ >,< σ >>) Normal (Gauss) distribution
- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

Using `optn[3]` two different versions may be specified:

0 : default algorithm allowing the specification of additional arguments for μ and σ and which returns probability, statistics and hypothesis value.

3 : the old implementation of the KS-Lilliefors method which assumes $\mu = 0$ and $\sigma = 1$ and returns the probability only.

("pare",x,a<,k>) Pareto distribution
- `x` is the upper limit for integration $x > 0$.
- `a` shape parameter ($a > 0$.
- `k` scale parameter ($k > 0$).

Default is $k = 1$.

("pois",n,λ) Poisson distribution
- `x` is the upper limit for cumulation $x > 0$.
- λ shape parameter.

("t",t,df<,nc>) (noncentral) t distribution
- `t` is chosen from t distribution.
- `df` degrees of freedom $df > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("unif",x<,<l>,<u>>) • `x` is the upper limit for integration $x > 0$.
- `l` left location.
- `u` right location ($u > l$).

Default is $l = 0$ and $u = 1$.

Using `optn[3]` three different versions may be specified:

0 : default algorithm: which returns probability, statistics, and hypothesis value.

1 : Anderson-Darling approximative, which returns the probability only.

2 : Marsaglia-Tsang-Wang exact (JSS,2003), which returns the probability only.

("wald",x,d) • `x` is the upper limit for integration $x > 0$.

• `d` shape parameter $d > 0$.

("weib",x,shape<,scale>) Weibull distribution

• `t` is the upper limit for integration $x > 0$.

• `shape` shape parameter $shape > 0$.

• `scale` scale parameter $scale > 0$.

Default is $scale = 1$.

... There may be a variable number of additional input arguments specifying non-default distributional parameters.

Output: prob The result is either a scalar or an object $p = Prob(d < D_n)$, where

$$D_n = max(x_1 - \frac{0}{n}, x_2 - frac1n, \ldots, x_n - fracn - 1n, \frac{1}{n} - x_1, \frac{2}{n} - x_2, \ldots, \frac{n}{n} - x_n$$

The Anderson-Darling version of the Cramer-von Mises-Kolmogorov-Smirnov test for univariate distribution uses

$$A_n = -n - \frac{1}{n}[ln(x_1 z_1) + 3ln(x_2 z_2) + 5ln(x_3 z_3) + \ldots + (2n-1)ln(x_n z_n)]$$

with $z_i = 1 - x_{n+1-i}$.

stat The `stat` result has the same size as `prob` and returns the test statistics (only for `optn[3]=0`).

Restrictions: 1. Missing value are returned if the input y contains missing data.

2.

Relationships: ksprob(), berkow(), jarbera(), shapwilk()

Examples: 1. Test different Distributions:

The following are $N = 1000$ uniform random generated data:

```
   options NOECHO;
#include "..\\tdata\\matlb.dat"
   options ECHO;
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* version=cdf */
< p31,s31 > = kstest(uf1000,optn,"unif");
print "KSTEST P: UNIF",p31;
print "KSTEST S: UNIF",s31;
```

```
     Hypothesis H0: Data are uniformally distributed versus
       Hypothesis H1: Data are not uniformally distributed
     H0 accepted: Probability=0.212528 Statistics=0.0333409
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* version=cdf */
< p32,s32 > = kstest(uf1000,optn,"norm");
print "KSTEST P: NORM",p32;
print "KSTEST S: NORM",s32;
```

```
       Hypothesis H0: Data are normally distributed versus
        Hypothesis H1: Data are not normally distributed
     H0 rejected: Probability=1.42564e-218 Statistics=0.4992
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* version=cdf */
< p32,s32 > = kstest(uf1000,optn,"expo");
print "KSTEST P: NORM",p32;
print "KSTEST S: NORM",s32;
```

```
     Hypothesis H0: Data are Exponential distributed versus
       Hypothesis H1: Data are not Exponential distributed
    H0 rejected: Probability=2.26383e-118 Statistics=0.367064
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* version=cdf */
< p32,s32 > = kstest(uf1000,optn,"f");
print "KSTEST P: NORM",p32;
print "KSTEST S: NORM",s32;
```

```
          Hypothesis H0: Data are F distributed versus
```

```
            Hypothesis H1: Data are not F distributed
          H0 rejected: Probability=0 Statistics=0.999001
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* version=cdf */
< p32,s32 > = kstest(uf1000,optn,"logi");
print "KSTEST P: NORM",p32;
print "KSTEST S: NORM",s32;
```

```
       Hypothesis H0: Data are Logistic distributed versus
        Hypothesis H1: Data are not Logistic distributed
    H0 rejected: Probability=1.65613e-218 Statistics=0.499126
```

2. Compare with other versions for uniform and normal distribution:
   Note, that there is only one return argument for versions $\neq 0$. The default version=0 computed for uniform: `Probability=0.212528`:

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           1 ]; /* version=and */
p31 = kstest(uf1000,optn,"unif");
print "KSTEST: UNIF",p31;
```

```
KSTEST: UNIF 0.8380
```

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           2 ]; /* version=mtw */
p32 = kstest(uf1000,optn,"unif");
print "KSTEST: UNIF",p32;
```

```
KSTEST: UNIF 0.8117
```

   The default version=0 computed for normal: `Probability=2.26383e-118` :

```
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           3 ]; /* version=ori */
p33 = kstest(uf1000,optn,"norm");
print "KSTEST: NORM",p33;
```

```
Number of Data=1000 Mean=0.488832 Standard Deviation=0.283241
     Hypothesis H0: Data are normally distributed versus
      Hypothesis H1: Data are not normally distributed
    KS-Lilliefors-Test on Normal Distribution: D= 0.154522
              Critical Values for d (two-sided)
                  10%      5%      2%      1%
               0.0261  0.0284  0.0313  0.0332
                 H0 rejected: Probability=0
```

3. Artificial Examples: for normal distribution:

```
opt3 = [ .,.,3];
options nolapack;
s1 = [ .02  1.15  2.55  0.77 .33 .20 ];
pr1 = kstest(s1,opt3,"nor");
s2 = [ 5 7 9 8 5 0 ];
pr2 = kstest(s2,opt3,"nor");
print " 1: Prob for NOR:", pr1, pr2;
```

```
Prob for NOR: 0.10000 0.10000
```

```
opt3 = [ .,.,3];
options lapack;
s1 = [ .02  1.15  2.55  0.77 .33 .20 ];
pr1 = kstest(s1,opt3,"nor");
s2 = [ 5 7 9 8 5 0 ];
pr2 = kstest(s2,opt3,"nor");
print " 2: Prob for NOR:", pr1, pr2;
```

```
Prob for NOR: 0.6400 0.9900
```

```
nr = 5000; nc = 1;
mu = rand(nr,nc,'g',"duni");
mn = rand(nr,nc,'g',"nor");
mn2 = mn .** 2;
```

```
opt3 = [ .,.,3];
options nolapack;
p1 = kstest(mu,opt3,"nor");
print "Prob for UNIF:",p1;
p2 = kstest(mn,opt3,"nor");
print " 1: Prob for NORM:",p2;
```

```
Prob UNIF for UNIF: 0.0000
Prob NORM for NORM: 0.0000
```

```
opt3 = [ .,.,3];
options lapack;
p1 = kstest(mu,opt3,"nor");
print "Prob for UNIF:",p1;
p2 = kstest(mn,opt3,"nor");
print " 2: Prob for NORM:",p2;
```

```
Prob UNIF for NORM: 0.0000
Prob NORM for NORM: 0.0000
```

4. Comparison of Uniform Random Generators: Here the sample size N=1000 is selected.

```
nr = 1000; nc = 5;
opt1 = [ .,.,1];
opt2 = [ .,.,2];
options RANDUNI;
mu1 = rand(nr,nc,'g',"duni");
p11 = kstest(mu1,opt1,"uni");
print "AND: Prob UNIF for RANDUNI:",p11;
p12 = kstest(mu1,opt2,"uni");
print "MTW: Prob for RANDUNI:",p12;
print "RANDUNI:", p11[+]/nc, p12[+]/nc, ssq(p11 - p12);
```

```
AND: Prob UNIF for RANDUNI:
   |         1
-------------
 1 |   0.92352
 2 |   0.68155
 3 |   0.44995
 4 |   0.75229
 5 |   0.82055
```

```
MTW: Prob for RANDUNI:
   |         1
-------------
 1 |   0.82039
 2 |   0.61299
 3 |   0.56562
 4 |   0.82924
```

```
 5 |   0.71396


RANDUNI: 0.7256 0.7084 0.04600


 opt1 = [ .,.,1];
 opt2 = [ .,.,2];
 options RANDKIS;
 mu2 = rand(nr,nc,'g',"duni");
 p21 = kstest(mu2,opt1,"uni");
 print "AND: Prob UNIF for RANDKIS:",p21;
 p22 = kstest(mu2,opt2,"uni");
 print "MTW: Prob for RANDKIS:",p22;
 print "RANDKIS:", p21[+]/nc, p22[+]/nc, ssq(p21 - p22);


AND: Prob UNIF for RANDKIS:
   |         1
-------------
 1 |   0.63250
 2 |   0.31721
 3 |   0.51274
 4 |   0.56430
 5 |   0.52699


MTW: Prob for RANDKIS:
   |         1
-------------
 1 |   0.38966
 2 |   0.69173
 3 |   0.29469
 4 |   0.55523
 5 |   0.62822


RANDKIS: 0.5107 0.5119 0.2571


 opt1 = [ .,.,1];
 opt2 = [ .,.,2];
 options RANDLECU;
 mu3 = rand(nr,nc,'g',"duni");
 p31 = kstest(mu3,opt1,"uni");
 print "AND: Prob UNIF for RANDLECU:",p31;
 p32 = kstest(mu3,opt2,"uni");
 print "MTW: Prob for RANDLECU:",p32;
```

```
 print "RANDLECU:", p31[+]/nc, p32[+]/nc, ssq(p31 - p32);


AND: Prob UNIF for RANDLECU:
   |         1
--------------
 1 |   0.96694
 2 |   0.36281
 3 |   0.42487
 4 |   0.07408
 5 |   0.38351


MTW: Prob for RANDLECU:
   |         1
--------------
 1 |   0.98915
 2 |   0.21339
 3 |   0.32869
 4 |   0.05316
 5 |   0.65719


RANDLECU: 0.4424 0.4483 0.1074


 opt1 = [ .,.,1];
 opt2 = [ .,.,2];
options RANDXORWOW;
 mu4 = rand(nr,nc,'g',"duni");
 p41 = kstest(mu4,opt1,"uni");
 print "AND: Prob UNIF for RANDXORWOW:",p41;
 p42 = kstest(mu4,opt2,"uni");
 print "MTW: Prob for RANXORWOW:",p42;
 res = ssq(p41 - p42);
 print "RANDXORWOW:", p41[+]/nc, p42[+]/nc, ssq(p41 - p42);
 options RANDUNI;


AND: Prob UNIF for RANDXORWOW:
   |         1
--------------
 1 |   0.05877
 2 |   0.85525
 3 |   0.18640
 4 |   0.41601
 5 |   0.27656
```

```
MTW: Prob for RANXORWOW:
   |         1
--------------
 1 |   0.10718
 2 |   0.90353
 3 |   0.04704
 4 |   0.20441
 5 |   0.61463


RANDXORWOW: 0.3586 0.3754 0.1832
```

2.2 Modifications of the `mvelps` Function

The `mvelps` function now has the form `v = mvelps(Cov,mu,U,c,optn)` which means that `Cov,mu,U,c` now are specified as single input arguments and no longer as entries of the option vector.
The example shows that the results are unchanged:

```
sigm = [ 1.,
         .5  1.,
         .5  .5  1. ];
mu = cons(1,3,0.);

u = [ .25 ,
       0.  .2,
       0.  0.  .1];
m = [ 1.  .5  .25 ];


optn = [ "ndim"         3,
         "bin"           ,
         "seed"       357,
         "ndf"         20,
         "randir"    1000,
         "nprel"       10,
         "e_rhs"       1. ];
ovec = mvelps(sigm,mu,u,m,optn);


      Number Dimensions                          3
      T Distribution DF                         20
      Preliminary Estimates                     10
      Number Random Directions               10000
      Value of Integral                  0.6792134
      Estimated Standard Error           5.07e-004
```

```
Initial Seed Value                                   357
Run Time (in seconds)                        2.000000000
```

3 Extensions to the Language

4 Extensions to Various Functions

4.1 Extension to the `tsmeas` Function

We have added a number of more functions:

"pacor" computes partial autocorrelations w.r.t. a scalar or vector of integer delays (lags) τ; in addition optional robust asymptotic standard errors are computed;

"ljung" Ljung-Box test for serial correlation for $\tau = 1, \ldots, nlag$ lags;

"lmtst" LM test for serial correlation for $\tau = 1, \ldots, nlag$ lags.

"nwcov" computing the Newey-West (Bartlett) covariance matrix for $\tau = 1, \ldots, nlag$ lags. This matrix is normally applied to the scores or an ARMA application.

"nwreg" ordinary least squares regression $\min \parallel \mathbf{X}\beta - y \parallel$ with t values based on the Newey-West covariance matrix for $\tau = 1, \ldots, nlag$ lags.

"difful" (augmented) Dickey-Fuller testing with specified lags or automatic lag selection (using critial values by K. Sheppard).

"grang" Granger causality testing: for a specified vector τ of lags; for homoskedastic and heteroskedastic with correlated or uncorrelated errors

"arvec" Vector AR estimation: for a specified vector τ of lags; for homoskedastic and heteroskedastic with correlated or uncorrelated errors

"iresp" Impulse Response modeling: for a specified vector τ of lags; for homoskedastic and heteroskedastic with correlated or uncorrelated errors.

4.1.1 Partial Autocorrelations

< pacor,ase > = tsmeas(xdat,"pacor",tau<,optn>)

The following specific options could be specified:

3 indicates whether robust (=1) or nonrobust (=0) asymptotic standard errors are being computed. The default is 1.

Option `optn[3]` must be specified to zero for returning nonrobust asymptotic standard errors (which are constant). An additional vector `tau` specifies a set of lags.
If `xdat` has more than one column, each column is treated as separate time series.
Some examples follow:

```
   xd = [ 0.8147 0.9058 0.1270 0.9134 0.6324
          0.0975 0.2785 0.5469 0.9575 0.9649
          0.1576 0.9706 0.9572 0.4854 0.8003
          0.1419 0.4218 0.9157 0.7922 0.9595 ];


   optn = [ 2 ]; tau = [ 1 2 3 ];
   pcor = tsmeas(xd,"pacor",tau,optn);
   print "PCor=",pcor;


PCor=
   |         1
--------------
 1 |  -0.07778
 2 |  -0.27478
 3 |  -0.00080


   print "MFE Implementation with robust ASE";
   rob = 1; optn = [ 2 . rob ];
   tau = [ 1 2 3 ];
   < pcor,rase > = tsmeas(xd,"pacor",tau,optn);
   print "PCor=",pcor;
   print "RASE=",rase;


PCor=
   |         1         2         3
-----------------------------------
 1 |  -0.07778  -0.27478  -0.00080

RASE=
   |         1         2         3
-----------------------------------
 1 |   0.21846   0.20710   0.24579


   print "MFE implementation: NonRobust ASE";
   tau = 3; rob = 0;
   optn = [ 2 tau rob ];
   < parcor,nase > = tsmeas(xd,"paco2",optn);
   print "ParCor=",parcor;
   print "Nonrobust ASE=",nase;


ParCor=
```

```
   |         1         2         3
-----------------------------------
 1 |  -0.07778  -0.27478  -0.00080

Nonrobust ASE=
   |         1         2         3
-----------------------------------
 1 |   0.22361   0.22361   0.22361
```

4.1.2 Ljung-Box Test for Serial Correlation

< stat,prob > = tsmeas(xdat,"ljung"<,optn>)

The following specific options could be specified.

2 integer `ntau >= 1` specifying how many lags $\tau = 1, \ldots, ntau$ are to be considered. The default is 1.

If `xdat` has more than one column, each column is treated as separate time series.
The function returns two arguments, the test statistics and the probability for each of the `ntau` lags.
The following example illustrates the use of this function:

```
 xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
          0.0975 0.2785 0.5469 0.9575 0.9649
          0.1576 0.9706 0.9572 0.4854 0.8003
          0.1419 0.4218 0.9157 0.7922 0.9595
          0.6557 0.0357 0.8491 0.9340 0.6787
          0.7577 0.7431 0.3922 0.6555 0.1712
          0.7060 0.0318 0.2769 0.0462 0.0971
          0.8235 0.6948 0.3171 0.9502 0.0344 ]';
  xc = xd2 - univar(xd2,"ari");


    print "Ljung-Box test for serial correlation:";
    tau = 5;
    optn = [ 2 tau ];
    < stat,prob > = tsmeas(xc,"ljung",optn);
    print "LJUNG: stat=",stat;
    print "LJUNG: prob=",prob;


Ljung-Box test for serial correlation:
```

```
LJUNG: stat=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.03534   0.07382   0.57096    2.0989    3.9281


LJUNG: prob=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.85089   0.96376   0.90305   0.71757   0.55982
```

4.1.3 (Robust) LM Test for Serial Correlation

< stat,prob > = tsmeas(xdat,"lmtst"<,optn>)

The following specific options could be specified.

2 an integer `ntau >= 1` specifying how many lags $\tau = 1, \ldots, ntau$ are to be considered. The default is 1.

3 indicates whether a robust (=1) or a nonrobust (=0) statistic is being computed. The default is 1.

If `xdat` has more than one column, each column is treated as separate time series.
The function returns two arguments, the test statistics and the probability for each of the `ntau` lags.
The following example illustrates the use of this function:

```
xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ]';
 xc = xd2 - univar(xd2,"ari");


   print "LM test for serial correlation: Nonrobust";
   tau = 5; rob = 0;
   optn = [ 2 tau rob ];
   < stat,prob > = tsmeas(xc,"lmtst",optn);
   print "LMTST: stat=",stat;
   print "LMTST: prob=",prob;
```

```
LM test for serial correlation: Nonrobust

LMTST: stat=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.03455   0.14361   0.49659    2.3261    5.1064


LMTST: prob=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.85255   0.93071   0.91964   0.67603   0.40303
```

```
print "LM test for serial correlation: Robust";
tau = 5; rob = 1;
optn = [ 2 tau rob ];
< stat,prob > = tsmeas(xc,"lmtst",optn);
print "LMTST: stat=",stat;
print "LMTST: prob=",prob;
```

```
LM test for serial correlation: Robust

LMTST: stat=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.03603   0.14081   0.54035    2.2770    4.5774


LMTST: prob=
   |         1         2         3         4         5
-------------------------------------------------------
 1 |   0.84945   0.93201   0.90994   0.68496   0.46960
```

4.1.4 Newey-West (Bartlett) Covariance Matrix

nwcov = tsmeas(xdat,"nwcov"<,optn>)

This function is normally applied to the scores of an ARMA application. The following specific options could be specified.

2 an integer `ntau >= 1` specifying how many lags $\tau = 1, \ldots, ntau$ are to be considered. The default is 1.

3 indicates whether the data are centered around the mean (=1) or the data are used as input (=0). The default is 0.

The function returns only one argument, the Newey-West covariance matrix. The following example shows not a typical use of the function:

```
 xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
          0.0975 0.2785 0.5469 0.9575 0.9649
          0.1576 0.9706 0.9572 0.4854 0.8003
          0.1419 0.4218 0.9157 0.7922 0.9595
          0.6557 0.0357 0.8491 0.9340 0.6787
          0.7577 0.7431 0.3922 0.6555 0.1712
          0.7060 0.0318 0.2769 0.0462 0.0971
          0.8235 0.6948 0.3171 0.9502 0.0344 ]‘;
  xc = xd2 - univar(xd2,"ari");


    print "Newey-West Covariance Matrix: Uncentered";
    tau = 5; cent = 0;
    optn = [ 2 tau cent ];
    nwc = tsmeas(xd2,"nwcov",optn);
    print "Newey-West COV: nwc=",nwc;


Newey-West COV: nwc= 1.9405


    print "Newey-West Covariance Matrix: Centered";
    tau = 5; cent = 1;
    optn = [ 2 tau cent ];
    nwc = tsmeas(xd2,"nwcov",optn);
    print "Newey-West COV: nwc=",nwc;


Newey-West COV: nwc= 0.1060
```

4.1.5 OLS Regression with Newey-West HAC Standard Errors

$< \text{gof,est,cov,yhat} > = \text{tsmeas(ydat,"nwreg",xdat<,optn>)}$

This function performs ordinary least squares (OLS) regression $\min \parallel \mathbf{X}\beta - y \parallel$ and computes t values based on the Newey-West covariance matrix for $\tau = 1, \ldots, nlag$ lags. The first input argument `ydat` specifies usually a N vector of values of a dependent variable (response), whereas the third input argument `xdat` specifies a $N \times n$ matrix of values of n independent variables (predictors). The number p of parameter estimates is then either $p = n + 1$ if an intercept variable is added or not, depending on the setting of `optn[3]`.
If `ydat` has more than one column, each column is treated as separate time series.
The following specific options could be specified:

2 an integer `ntau >= 1` specifying how many lags $\tau = 1, \ldots, ntau$ are to be considered. The default is 1.

3 indicates whether a constant is added (=1: default) or not (=0) to the set of parameter estimates.

Some examples follow:

1. OLS solution of 40×3 linear system, without intercept:

```
print "Matlab 200 by 3 RandomData: ran203[200,3]";
options NOECHO;
%inc "..\\tdata\\matlb.dat";
options ECHO;

ran40 = ran203[1:40,];


xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ]';
 xd2 = flip(xd2,"ud");


 print "Newey-West Regression: w/o Intercept";
 pri = 2; tau = 5; intc = 0;
 optn = [ pri tau intc ];
 < gof,est,nwcov,yhat > = tsmeas(xd2,"nwreg",ran40,optn);
 print "Newey-West REG: gof=",gof;
 print "Newey-West REG: est=",est;
 print "Newey-West REG: nwcov=",nwcov;
 print "Newey-West REG: yhat=",yhat;
```

```
Newey-West REG: gof=
   |         1
---------------
 1 |   0.09458
 2 |   0.75065
 3 |   0.73718

Newey-West REG: est=
   |         1         2
------------------------
 1 |   0.50690    4.8994
 2 |  -0.04048  -0.19482
 3 |   0.55293    4.6315

Newey-West REG: nwcov=
 S |         1         2         3
-----------------------------------
 1 |   0.01070
 2 |  -0.01403   0.04318
 3 |  -0.00069  -0.01440   0.01425
```

```
Newey-West REG: yhat=
   |         1
---------------
 1 |   0.62073
 2 |   0.49591
 3 |   0.36245
 4 |   0.70181
 5 |   0.69115
 6 |   0.39841
 7 |   0.45876
 8 |   0.27353
 9 |   0.49820
10 |   0.64206
11 |   0.36503
12 |   0.84164
13 |   0.69152
14 |   0.69012
15 |   0.76872
16 |   0.59961
17 |   0.49844
18 |   0.63702
19 |   0.45074
20 |   0.80658
21 |   0.75040
22 |   0.21488
23 |   0.46320
24 |   0.61332
25 |   0.39239
26 |   0.49979
27 |   0.60226
28 |   0.48576
29 |   0.57474
30 |   0.55427
31 |   0.62026
32 |   0.52725
33 |   0.46856
34 |   0.52417
35 |   0.17334
36 |   0.78652
37 |   0.50004
38 |   0.51929
39 |   0.84883
40 |   0.03448
```

2. OLS solution of 40×3 linear system, including intercept:

```
print "Newey-West Regression: With Intercept";
 pri = 2; tau = 5; intc = 1;
 optn = [ pri tau intc ];
 < gof,est,nwcov,yhat > = tsmeas(xd2,"nwreg",ran40,optn);
 print "Newey-West REG: gof=",gof;
 print "Newey-West REG: est=",est;
 print "Newey-West REG: nwcov=",nwcov;
 print "Newey-West REG: yhat=",yhat;
```

Newey-West REG: gof=

	1
1	0.07753
2	0.13355
3	0.06134

Newey-West REG: est=

	1	2
1	0.46282	2.5038
2	0.24662	1.9244
3	-0.30860	-1.5547
4	0.21950	1.1649

Newey-West REG: nwcov=

S	1	2	3	4
1	0.03417			
2	-0.01549	0.01642		
3	-0.01478	-0.00474	0.03940	
4	-0.02892	0.01066	0.00371	0.03551

Newey-West REG: yhat=

	1
1	0.55774
2	0.59005
3	0.37505
4	0.62702
5	0.66331
6	0.35072
7	0.40135
8	0.43528
9	0.52196
10	0.58979
11	0.55413
12	0.75288
13	0.64303
14	0.69139
15	0.55733
16	0.65031
17	0.61375
18	0.70733
19	0.61110
20	0.69911
21	0.69947
22	0.27962
23	0.55940
24	0.69463
25	0.38469
26	0.40903
27	0.60724
28	0.64096
29	0.64523
30	0.57107
31	0.56707
32	0.59687
33	0.48506
34	0.46495
35	0.47119
36	0.77809
37	0.60607
38	0.59010
39	0.71883
40	0.32949

4.1.6 (Augmented) Dickey-Fuller Testing

< stat,pval,cval,lags,resi > = tsmeas(xdat,"difu1"<,optn>)

This function performs (augmented) Dickey-Fuller testing with or without automatic lag selection. This function is based on Kevin Sheppard's implementation in the Oxford MFE Toolbox and is using the tables of critical values obtained by simulation with a Matlab script posted on that website. If automatic lag selection based on AIC or BIC is specified, the estimates are returned for the optimal number of lags.
If `xdat` has more than one column, each column is treated as separate time series.
It returns:

stat the test statistics (if `xdat` has more than one column this is a vector)

pval the probability (if `xdat` has more than one column this is a vector)

cval a vector of six critical values corresponding to the following probabilities: [.01.05.1.9.95.99] (if `xdat` has more than one column this is a matrix)

lags returns the number of lags to which the the other estimates correspond, i.e. for automatic lag selection the optimal number of lags (must be integer in `[0,opt[2]]`

resi a vector or matrix of residuals.

Note, that the optimal solution may correspond to 0 lags. The following specific options could be specified:

2 an integer `ntau >= 1` specifying how many lags $\tau = 1, \ldots, ntau$ are to be considered. For automatic lage selection this is the maximum number of lags tried. The default is 1.

3 specifies the degree of the polynomial included in the ADF regression, i.e. the order of augmentation:

- **0** no deterministic terms included
- **1** constant is included
- **2** time trend is added to constant
- **3** Constant, DGP assumed to have a time trend

4 specifies which criterion is used for the automatic lag determination:

- **0** no automatic lag determination (default);
- **1** automatic lag determination using AIC;
- **2** automatic lag determination using BIC.

All of the following examples use this data set:

```
xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ]‘;
```

1. $ndeg = 0$ with specified number of 5 lags:

```
lags = 5; ndeg = 0;
optn = [ 2 lags ndeg ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"diful",optn);
print "Dickey-Fuller: stat=",stat;
print "Dickey-Fuller: prob=",prob;
print "Dickey-Fuller: cval=",cval;
```

```
Dickey-Fuller Statistics -0.689827, Probability  0.4199 for 5 Lags


Dickey-Fuller: stat=-0.6898

Dickey-Fuller: prob= 0.4199

Dickey-Fuller: cval=
   |         1         2         3         4         5         6
---------------------------------------------------------------------
 1 |   -2.7328   -2.0186   -1.6648   0.94878    1.3721    2.1844
```

2. $ndeg = 1$ with specified number of 5 lags:

```
lags = 5; ndeg = 1;
optn = [ 2 lags ndeg ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"diful",optn);
print "Dickey-Fuller: stat=",stat;
print "Dickey-Fuller: prob=",prob;
print "Dickey-Fuller: cval=",cval;
```

```
Dickey-Fuller Statistics -1.51482, Probability  0.5435 for 5 Lags
```

```
Dickey-Fuller: stat=-1.5148

Dickey-Fuller: prob= 0.5435

Dickey-Fuller: cval=
   |         1         2         3         4         5         6
---------------------------------------------------------------------
 1 |   -4.1091   -3.2310   -2.8221  -0.39517  -0.01339   0.71737
```

3. $ndeg = 2$ with specified number of 5 lags:

```
lags = 5; ndeg = 2;
optn = [ 2 lags ndeg ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller: stat=",stat;
print "Dickey-Fuller: prob=",prob;
print "Dickey-Fuller: cval=",cval;
```

```
 Dickey-Fuller Statistics -2.24125, Probability  0.5143 for 5 Lags


Dickey-Fuller: stat=-2.2413

Dickey-Fuller: prob= 0.5143

Dickey-Fuller: cval=
   |         1         2         3         4         5         6
---------------------------------------------------------------------
 1 |   -4.7563   -3.8449   -3.4375   -1.2658  -0.93043  -0.22962
```

4. $ndeg = 3$ with specified number of 5 lags:

```
lags = 5; ndeg = 3;
optn = [ 2 lags ndeg ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller: stat=",stat;
print "Dickey-Fuller: prob=",prob;
print "Dickey-Fuller: cval=",cval;
```

```
 Dickey-Fuller Statistics -1.78214, Probability  0.0374 for 5 Lags
```

```
Dickey-Fuller: stat=-1.7821

Dickey-Fuller: prob= 0.03736

Dickey-Fuller: cval=
   |         1         2         3         4         5         6
---------------------------------------------------------------------
 1 |   -2.3263   -1.6449   -1.2816    1.2816    1.6449    2.3263
```

5. $ndeg = 0$ with BIC selection of a maximum of 8 lags:

```
lagmax = 8; ndeg = 0; bic = 2;
optn = [ 2 lagmax ndeg bic ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller BIC: stat=",stat;
print "Dickey-Fuller BIC: prob=",prob;
print "Dickey-Fuller BIC: cval=",cval;
print "Dickey-Fuller BIC: lags=",lags;
```

```
***********************************************
Automatic Lag Search (BIC) for Dickey-Fuller Test
***********************************************

      Lags         SSE         AIC         BIC

         0  0.15601843 -1.80778113 -1.76555914
         1  0.14110212 -1.85827139 -1.77382741
         2  0.12912306 -1.89698941 -1.77032345
         3  0.12888016 -1.84887229 -1.67998434
         4  0.10631328 -1.99136509 -1.78025516
         5  0.10241622 -1.97871020 -1.72537828
         6  0.09598876 -1.99352419 -1.69797029
         7  0.08584639 -2.05519570 -1.71741980
         8  0.08292584 -2.03980850 -1.65981063

    The Smallest BIC Value of -1.78026 Determines Lag Size 4
Dickey-Fuller Statistics -0.73469, Probability  0.4007 for 4 Lags


Dickey-Fuller BIC: stat=-0.7347

Dickey-Fuller BIC: prob= 0.4007
```

```
Dickey-Fuller BIC: cval=
   |         1         2         3         4         5         6
---------------------------------------------------------------------
 1 |   -2.7289   -2.0169   -1.6638   0.94707    1.3697    2.1797

Dickey-Fuller BIC: lags= 4.0000
```

6. $ndeg = 1$ with BIC selection of a maximum of 8 lags:

```
lagmax = 8; ndeg = 1; bic = 2;
optn = [ 2 lagmax ndeg bic ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller BIC: stat=",stat;
print "Dickey-Fuller BIC: prob=",prob;
print "Dickey-Fuller BIC: cval=",cval;
print "Dickey-Fuller BIC: lags=",lags;
```

```
***********************************************
Automatic Lag Search (BIC) for Dickey-Fuller Test
***********************************************

      Lags         SSE         AIC         BIC

         0  0.11179578 -2.09108143 -2.00663745
         1  0.11143637 -2.04430150 -1.91763554
         2  0.11009370 -2.00642348 -1.83753553
         3  0.10451435 -2.00843088 -1.79732094
         4  0.09616274 -2.04171333 -1.78838142
         5  0.09499131 -2.00396983 -1.70841592
         6  0.08321125 -2.08637271 -1.74859682
         7  0.07928279 -2.08473426 -1.70473638
         8  0.07215951 -2.12887620 -1.70665634

    The Smallest BIC Value of -2.00664 Determines Lag Size 0
 Dickey-Fuller Statistics -5.94067, Probability  0.0000 for 0 Lags


Dickey-Fuller BIC: stat=-5.9407

Dickey-Fuller BIC: prob= 0.0000

Dickey-Fuller BIC: cval=
   |         1         2         3         4         5         6
```

```
-------------------------------------------------------------
 1 |   -4.0210   -3.1847   -2.7907  -0.40155  -0.02262   0.70209

Dickey-Fuller BIC: lags= 0.0000
```

7. $ndeg = 2$ with BIC selection of a maximum of 8 lags:

```
lagmax = 8; ndeg = 2; bic = 2;
optn = [ 2 lagmax ndeg bic ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller BIC: stat=",stat;
print "Dickey-Fuller BIC: prob=",prob;
print "Dickey-Fuller BIC: cval=",cval;
print "Dickey-Fuller BIC: lags=",lags;
```

```
*************************************************
Automatic Lag Search (BIC) for Dickey-Fuller Test
*************************************************

      Lags         SSE         AIC         BIC

         0  0.09891179 -2.16352685 -2.03686090
         1  0.09831659 -2.11956245 -1.95067451
         2  0.09828716 -2.06986186 -1.85875193
         3  0.08904510 -2.11861232 -1.86528040
         4  0.08625527 -2.10044407 -1.80489016
         5  0.08613447 -2.05184556 -1.71406967
         6  0.07481467 -2.14274130 -1.76274342
         7  0.07231972 -2.12665843 -1.70443857
         8  0.06426427 -2.19475143 -1.73030958

    The Smallest BIC Value of -2.03686 Determines Lag Size 0
 Dickey-Fuller Statistics -6.26798, Probability  0.0000 for 0 Lags


Dickey-Fuller BIC: stat=-6.2680

Dickey-Fuller BIC: prob= 0.0000

Dickey-Fuller BIC: cval=
   |         1         2         3         4         5         6
-------------------------------------------------------------------
 1 |   -4.6275   -3.7780   -3.3917   -1.2638  -0.93208  -0.24390
```

```
Dickey-Fuller BIC: lags= 0.0000
```

8. $ndeg = 3$ with BIC selection of a maximum of 8 lags:

```
lagmax = 8; ndeg = 3; bic = 2;
optn = [ 2 lagmax ndeg bic ];
< stat,prob,cval,lags,resi > = tsmeas(xd2,"difu1",optn);
print "Dickey-Fuller BIC: stat=",stat;
print "Dickey-Fuller BIC: prob=",prob;
print "Dickey-Fuller BIC: cval=",cval;
print "Dickey-Fuller BIC: lags=",lags;
```

```
**********************************************
Automatic Lag Search (BIC) for Dickey-Fuller Test
**********************************************

      Lags         SSE         AIC         BIC

         0  0.11179578 -2.09108143 -2.00663745
         1  0.11143637 -2.04430150 -1.91763554
         2  0.11009370 -2.00642348 -1.83753553
         3  0.10451435 -2.00843088 -1.79732094
         4  0.09616274 -2.04171333 -1.78838142
         5  0.09499131 -2.00396983 -1.70841592
         6  0.08321125 -2.08637271 -1.74859682
         7  0.07928279 -2.08473426 -1.70473638
         8  0.07215951 -2.12887620 -1.70665634

    The Smallest BIC Value of -2.00664 Determines Lag Size 0
Dickey-Fuller Statistics -6.09299, Probability  0.0000 for 0 Lags


Dickey-Fuller BIC: stat=-6.0930

Dickey-Fuller BIC: prob= 5.541e-010

Dickey-Fuller BIC: cval=
   |         1         2         3         4         5         6
-------------------------------------------------------------------
 1 |   -2.3263   -1.6449   -1.2816    1.2816    1.6449    2.3263

Dickey-Fuller BIC: lags= 0.0000
```

4.1.7 Granger Causality Testing

< stat,pval > = tsmeas(xdat,"grang",tau<,optn>)

The function implements the Granger causality test for an N *timesn* data matrix and for a specified integer vector of K positive lags $\tau = \tau_1, \ldots, \tau_K$ where $tmax = \max_k = 1^K(\tau_1, \ldots, \tau_K)$.
It returns:

stat the test statistics (if `xdat` has more than one column this is a vector)

pval the probability (if `xdat` has more than one column this is a vector)

The following specific options could be specified:

2 an integer specifying the inference method:

- **1** likelihood ratio test (default)
- **2** LM test
- **3** Wald test

3 binary integer *intc* specifying whether (=1) or not (=0, is default) a constant intercept variable is used in the model;

4 binary integer specifying Homoskedastic (=0) or Heteroskedastic (=1, is default);

5 binary integer specifying uncorrelated (=0, is default) or correlated (=1) error;

All examples work with the same 200×3 random generated data set:

```
print "Matlab 200 by 3 RandomData: ran203[200,3]";
options NOECHO;
%inc "..\\tdata\\matlb.dat";
options ECHO;


 print "Ex 1: infer=1 ihet=1 uncor=0";
 tau = [ 1 3 4 ];
 pri = 2; infer = 1; intc = 1; ihet = 1; uncor = 0;
 optn = [ pri infer intc ihet uncor ];
 < stat,prob > = tsmeas(ran203,"grang",tau,optn);
 print "STAT=",stat;
 print "PROB=",prob;
```

```
STAT=
        |     Var_1     Var_2     Var_3
-------------------------------------
  Var_1 |    3.2738    2.5597   0.85104
  Var_2 |   0.28437   0.18616    3.6460
  Var_3 |    2.6118    2.1632    1.9429


PROB=
        |     Var_1     Var_2     Var_3
-------------------------------------
  Var_1 |   0.35130   0.46460   0.83722
  Var_2 |   0.96294   0.97979   0.30231
  Var_3 |   0.45542   0.53924   0.58435


    print "Ex 2: infer=1 ihet=0 uncor=0";
    pri = 2; infer = 1; intc = 1; ihet = 0; uncor = 0;
    optn = [ pri infer intc ihet uncor ];
    < stat,prob > = tsmeas(ran203,"grang",tau,optn);
    print "STAT=",stat;
    print "PROB=",prob;


STAT=
        |     Var_1     Var_2     Var_3
-------------------------------------
  Var_1 |    2.9727    1.9965   0.67241
  Var_2 |   0.26558   0.15035    3.0786
  Var_3 |    2.3809    1.9809    1.4207


PROB=
        |     Var_1     Var_2     Var_3
-------------------------------------
  Var_1 |   0.39586   0.57313   0.87967
  Var_2 |   0.96637   0.98518   0.37966
  Var_3 |   0.49720   0.57639   0.70069


    print "Ex 3: infer=2 ihet=0 uncor=1";
    pri = 2; infer = 2; intc = 1; ihet = 0; uncor = 1;
    optn = [ pri infer intc ihet uncor ];
    < stat,prob > = tsmeas(ran203,"grang",tau,optn);
    print "STAT=",stat;
    print "PROB=",prob;
```

```
STAT=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |   0.07282   0.01647   0.00751
  Var_2 |   0.00216   0.00826   0.01171
  Var_3 |   0.08615   0.03545   0.02027


PROB=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |   0.99489   0.99944   0.99983
  Var_2 |   0.99997   0.99980   0.99966
  Var_3 |   0.99345   0.99824   0.99924


    print "Ex 4: infer=3 ihet=1 uncor=0";
    pri = 2; infer = 3; intc = 1; ihet = 1; uncor = 0;
    optn = [ pri infer intc ihet uncor ];
    < stat,prob > = tsmeas(ran203,"grang",tau,optn);
    print "STAT=",stat;
    print "PROB=",prob;


STAT=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |    30.036    55.127   0.18754
  Var_2 |    1.7373    4.0050    2.0567
  Var_3 |    1.8878    1.7107    2.0502


PROB=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |   0.00000    6e-012   0.97958
  Var_2 |   0.62866   0.26093   0.56071
  Var_3 |   0.59603   0.63456   0.56205


    print "Ex 5: infer=3 ihet=1 uncor=1";
    pri = 2; infer = 3; intc = 1; ihet = 1; uncor = 1;
    optn = [ pri infer intc ihet uncor ];
    < stat,prob > = tsmeas(ran203,"grang",tau,optn);
    print "STAT=",stat;
    print "PROB=",prob;
```

```
STAT=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |    30.036    55.127   0.18754
  Var_2 |    1.7373    4.0050    2.0567
  Var_3 |    1.8878    1.7107    2.0502


PROB=
        |     Var_1     Var_2     Var_3
---------------------------------------
  Var_1 |   0.00000    6e-012   0.97958
  Var_2 |   0.62866   0.26093   0.56071
  Var_3 |   0.59603   0.63456   0.56205
```

4.1.8 Vector AR Modeling

< est,ase,r2s,cov,resi > = tsmeas(xdat,"arvec",tau<,optn>)

The function implements (multivariate) vector AR modeling for an N *timesn* data matrix and for a specified integer vector of K positive lags $\tau = \tau_1, \ldots, \tau_K$ where $tmax = \max_k = 1^K(\tau_1, \ldots, \tau_K)$.
There are $p = (intc + K * n) * n$ parameters to fit.
It returns:

est $(intc + K * n) \times n$ matrix of parameter estimates;

ase $(intc + K * n) \times 3 * n$ matrix of asymptotic standard errors, t values, and probabilities;

r2s $n \times n + 1$ matrix containing the n R-squared values in its first column and the $n \times n$ matrix of sums of squared errors in its last n columns;

cov the symmetric $p \times p$ covariance matrix of estimates;

resi a N vector or $N \times n$ matrix of residuals.

The following specific options could be specified:

2 unused;

3 binary integer *intc* specifying whether (=1) or not (=0, is default) a constant intercept variable is used in the model;

4 binary integer specifying Homoskedastic (=0) or Heteroskedastic (=1, is default);

5 binary integer specifying uncorrelated (=0, is default) or correlated (=1) error;

All examples work with the same 200×3 random generated data set:

```
print "Matlab 200 by 3 RandomData: ran203[200,3]";
options NOECHO;
%inc "..\\tdata\\matlb.dat";
options ECHO;
```

```
print "Ex 1.1: intc=1: ihet=1 uncor=0: tau=[ 1 4 ]";
tau = [ 1 4 ];
pri = 2; intc = 1; ihet = 1; uncor = 0;
optn = [ pri . intc ihet uncor ];
< est,ase,r2s,cov,resi > = tsmeas(ran203,"arvec",tau,optn);
print "EST=",est;
print "ASE=",ase;
print "R2S=",r2s;
print "COV=",cov;
print "Resi=",resi;
```

```
EST=
              |     Var_1     Var_2     Var_3
----------------------------------------------
     Constant |   0.57015   0.45458   0.57867
  Var_1_Lag_1 |  -0.02170  -0.02963  -0.05872
  Var_1_Lag_2 |  -0.02145  -0.01881  -0.04751
  Var_2_Lag_1 |  -0.03458   0.02628   0.05220
  Var_2_Lag_2 |   0.01759   0.01559  -0.11212
  Var_3_Lag_1 |  -0.11315  -0.02130   0.05411
  Var_3_Lag_2 |   0.02703   0.10050  -0.05293
```

```
ASE=
              |     ASE_1     ASE_2     ASE_3       T_1       T_2
-------------------------------------------------------------------
     Constant |   0.10354   0.09385   0.09510    5.5064  -0.20038
  Var_1_Lag_1 |   0.07624   0.07069   0.06236    5.9625  -0.67202
  Var_1_Lag_2 |   0.07035   0.07366   0.06548    8.2255  -0.46939
  Var_2_Lag_1 |   0.07735   0.07770   0.07132  -0.28052   0.33827
  Var_2_Lag_2 |   0.07406   0.06606   0.06734  -0.40005   0.79026
  Var_3_Lag_1 |   0.07531   0.06917   0.06825  -0.77972   0.25424
  Var_3_Lag_2 |   0.08528   0.07872   0.07435  -0.25154   0.19803

              |       T_3       P_1       P_2       P_3
-------------------------------------------------------
     Constant |   -1.1790    4e-008   0.84118   0.23839
```

```
  Var_1_Lag_1 |   -1.8146    2e-009   0.50157   0.06958
  Var_1_Lag_2 |  -0.32534    2e-016   0.63879   0.74492
  Var_2_Lag_1 |   0.75864   0.77908   0.73516   0.44807
  Var_2_Lag_2 |   0.40134   0.68912   0.42938   0.68817
  Var_3_Lag_1 |    1.4725   0.43556   0.79931   0.14088
  Var_3_Lag_2 |  -0.71191   0.80140   0.84302   0.47652


R2S=
              |        R2     SE2_1     SE2_2     SE2_3
-------------------------------------------------------
     Constant |   0.01380   0.08493  -0.00081  -0.01204
  Var_1_Lag_1 |   0.01073  -0.00081   0.07861   0.00077
  Var_1_Lag_2 |   0.02752  -0.01204   0.00077   0.06652


    print "Ex 1.3: intc=1: ihet=1 uncor=1: tau=[ 1 4 ]";
    tau = [ 1 4 ];
    pri = 2; intc = 1; ihet = 1; uncor = 1;
    optn = [ pri . intc ihet uncor ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"arvec",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;


EST=
              |     Var_1     Var_2     Var_3
-----------------------------------------------
     Constant |   0.57015   0.45458   0.57867
  Var_1_Lag_1 |  -0.02170  -0.02963  -0.05872
  Var_1_Lag_2 |  -0.02145  -0.01881  -0.04751
  Var_2_Lag_1 |  -0.03458   0.02628   0.05220
  Var_2_Lag_2 |   0.01759   0.01559  -0.11212
  Var_3_Lag_1 |  -0.11315  -0.02130   0.05411
  Var_3_Lag_2 |   0.02703   0.10050  -0.05293


ASE=
              |     ASE_1     ASE_2     ASE_3       T_1       T_2
-------------------------------------------------------------------
     Constant |   0.10354   0.09385   0.09510    5.5064  -0.20038
  Var_1_Lag_1 |   0.07624   0.07069   0.06236    5.9625  -0.67202
  Var_1_Lag_2 |   0.07035   0.07366   0.06548    8.2255  -0.46939
  Var_2_Lag_1 |   0.07735   0.07770   0.07132  -0.28052   0.33827
```

```
  Var_2_Lag_2 |   0.07406   0.06606   0.06734  -0.40005   0.79026
  Var_3_Lag_1 |   0.07531   0.06917   0.06825  -0.77972   0.25424
  Var_3_Lag_2 |   0.08528   0.07872   0.07435  -0.25154   0.19803


              |       T_3       P_1       P_2       P_3
-------------------------------------------------------------
     Constant |   -1.1790    4e-008   0.84118   0.23839
  Var_1_Lag_1 |   -1.8146    2e-009   0.50157   0.06958
  Var_1_Lag_2 |  -0.32534    2e-016   0.63879   0.74492
  Var_2_Lag_1 |   0.75864   0.77908   0.73516   0.44807
  Var_2_Lag_2 |   0.40134   0.68912   0.42938   0.68817
  Var_3_Lag_1 |    1.4725   0.43556   0.79931   0.14088
  Var_3_Lag_2 |  -0.71191   0.80140   0.84302   0.47652


    print "Ex 2.1: intc=0: ihet=1 uncor=0: tau=[ 1 4 ]";
    tau = [ 1 4 ];
    pri = 2; intc = 0; ihet = 1; uncor = 0;
    optn = [ pri . intc ihet uncor ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"arvec",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;


EST=
              |     Var_1     Var_2     Var_3
---------------------------------------------
  Var_1_Lag_1 |   0.13654   0.09653   0.10188
  Var_1_Lag_2 |   0.13901   0.10913   0.11536
  Var_2_Lag_1 |   0.19367   0.20827   0.28386
  Var_2_Lag_2 |   0.19434   0.15651   0.06727
  Var_3_Lag_1 |   0.04117   0.10174   0.21073
  Var_3_Lag_2 |   0.25120   0.27924   0.17460


ASE=
              |     ASE_1     ASE_2     ASE_3       T_1       T_2
-----------------------------------------------------------------
  Var_1_Lag_1 |   0.07368   0.06868   0.06076    1.8532    2.8201
  Var_1_Lag_2 |   0.06605   0.06990   0.06476    1.4616    2.9796
  Var_2_Lag_1 |   0.07002   0.06924   0.06170    1.4551    4.1000
  Var_2_Lag_2 |   0.06978   0.06523   0.06153    1.9922    2.9791
  Var_3_Lag_1 |   0.07550   0.06921   0.06660    1.4455    2.2614
```

```
  Var_3_Lag_2 |   0.07819   0.07063   0.07340    1.4753   0.95240

              |       T_3       P_1       P_2       P_3
--------------------------------------------------------
  Var_1_Lag_1 |   0.67757   0.06386   0.00480   0.49805
  Var_1_Lag_2 |    1.5710   0.14386   0.00289   0.11618
  Var_2_Lag_1 |    3.4152   0.14564   0.00004   0.00064
  Var_2_Lag_2 |    4.0824   0.04635   0.00289   0.00004
  Var_3_Lag_1 |    4.1930   0.14831   0.02373   0.00003
  Var_3_Lag_2 |    2.3786   0.14014   0.34089   0.01738
```

```
    print "Ex 2.3: intc=0: ihet=1 uncor=1: tau=[ 1 4 ]";
    tau = [ 1 4 ];
    pri = 2; intc = 0; ihet = 1; uncor = 1;
    optn = [ pri . intc ihet uncor ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"arvec",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;
```

```
EST=
              |     Var_1     Var_2     Var_3
----------------------------------------------
  Var_1_Lag_1 |   0.13654   0.09653   0.10188
  Var_1_Lag_2 |   0.13901   0.10913   0.11536
  Var_2_Lag_1 |   0.19367   0.20827   0.28386
  Var_2_Lag_2 |   0.19434   0.15651   0.06727
  Var_3_Lag_1 |   0.04117   0.10174   0.21073
  Var_3_Lag_2 |   0.25120   0.27924   0.17460

ASE=
              |     ASE_1     ASE_2     ASE_3       T_1       T_2
-------------------------------------------------------------------
  Var_1_Lag_1 |   0.07368   0.06868   0.06076    1.8532    2.8201
  Var_1_Lag_2 |   0.06605   0.06990   0.06476    1.4616    2.9796
  Var_2_Lag_1 |   0.07002   0.06924   0.06170    1.4551    4.1000
  Var_2_Lag_2 |   0.06978   0.06523   0.06153    1.9922    2.9791
  Var_3_Lag_1 |   0.07550   0.06921   0.06660    1.4455    2.2614
  Var_3_Lag_2 |   0.07819   0.07063   0.07340    1.4753   0.95240
```

```
              |       T_3       P_1       P_2       P_3
--------------------------------------------------------
  Var_1_Lag_1 |   0.67757   0.06386   0.00480   0.49805
  Var_1_Lag_2 |    1.5710   0.14386   0.00289   0.11618
  Var_2_Lag_1 |    3.4152   0.14564   0.00004   0.00064
  Var_2_Lag_2 |    4.0824   0.04635   0.00289   0.00004
  Var_3_Lag_1 |    4.1930   0.14831   0.02373   0.00003
  Var_3_Lag_2 |    2.3786   0.14014   0.34089   0.01738
```

4.1.9 Impulse Response Modeling

< est,ase,r2s,cov,resi > = tsmeas(xdat,"iresp",tau<,optn>)

The function implements (multivariate) impulse response modeling for an N *timesn* data matrix, fora specified value of lead, and for a specified integer vector of K positive lags $\tau = \tau_1, \ldots, \tau_K$ where $tmax = \max_k = 1^K(\tau_1, \ldots, \tau_K)$.
There are $p = (intc + tmax * n) * n$ parameters to fit.
It returns:

est $(intc + tmax * n) \times n$ matrix of parameter estimates;

ase $(intc + tmax * n) \times n$ matrix of asymptotic standard errors;

r2s $n \times n + 1$ matrix containing the n R-squared values in its first column and the $n \times n$ matrix used for `optn[6]` type scaling in its last n columns;

cov the symmetric $p \times p$ covariance matrix of estimates;

resi a N vector or $N \times n$ matrix of residuals.

The following specific options could be specified:

2 positive integer specifying the number of leads (default is 1);

3 binary integer *intc* specifying whether (=1) or not (=0, is default) a constant intercept variable is used in the model;

4 binary integer specifying Homoskedastic (=0) or Heteroskedastic (=1, is default);

5 binary integer specifying uncorrelated (=0, is default) or correlated (=1) error;

6 positive integer specifying the type of shocks:

- **0** unit (unscaled) shocks: use $n \times n$ identity matrix;
- **1** scaled but uncorrelated shocks: use square root of diagonal values of $n \times n$ SSE matrix of vector AR model, this is the default;

2 scaled and correlated shocks: use Cholesky factor of $n \times n$ SSE matrix of vector AR model,

3 scaled and correlated shocks (spectral decomposition): use square roots of all entries of $n \times n$ SSE matrix of vector AR model.

All examples work with the same 200×3 random generated data set:

```
print "Matlab 200 by 3 RandomData: ran203[200,3]";
options NOECHO;
%inc "..\\tdata\\matlb.dat";
options ECHO;
```

```
print "Ex 0.2: intc=1: ihet=1 uncor=0 sqtyp=0: tau=[ 1 3 ]";
tau = [ 1 3 ];
pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 0; sqtyp = 0;
optn = [ pri lead intc ihet uncor sqtyp ];
< est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
print "EST=",est;
print "ASE=",ase;
print "R2S=",r2s;
print "COV=",cov;
print "Resi=",resi;
```

```
EST=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
---------------------------------------------------------------
  Lead_0 |   1.00000   0.00000   0.00000   0.00000   1.00000
  Lead_1 |  -0.02467  -0.03246  -0.04451  -0.01073  -0.02738
  Lead_2 |   0.00242   0.00030   0.00010   0.00195  -0.00023
  Lead_3 |  -0.15225   0.02240  -0.00580   0.07529  -0.01507
  Lead_4 |   0.00160   0.01018   0.01235  -0.00221   0.00093
  Lead_5 |  -0.00089   0.00009   0.00010  -0.00026   0.00012


         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   1.00000
  Lead_1 |  -0.04249  -0.03279   0.03131   0.05340
  Lead_2 |  -0.00063  -0.00128   0.00188   0.00298
  Lead_3 |  -0.09679   0.07659  -0.13703  -0.07969
  Lead_4 |  -0.00178   0.01363  -0.00976  -0.00894
  Lead_5 |   0.00003   0.00063  -0.00092  -0.00108


ASE=
```

```
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
---------------------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.07385   0.06916   0.07486   0.06948   0.07358
  Lead_2 |   0.00529   0.00509   0.00540   0.00593   0.00619
  Lead_3 |   0.07101   0.07005   0.07817   0.06932   0.06783
  Lead_4 |   0.02534   0.01869   0.02245   0.01672   0.01554
  Lead_5 |   0.00206   0.00175   0.00237   0.00188   0.00155

         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.07843   0.06274   0.06483   0.07103
  Lead_2 |   0.00597   0.00534   0.00546   0.00905
  Lead_3 |   0.07831   0.06380   0.06626   0.06801
  Lead_4 |   0.01606   0.01870   0.01507   0.01901
  Lead_5 |   0.00219   0.00229   0.00205   0.00274


R2S=
        |        R2     SE2_1     SE2_2     SE2_3
-----------------------------------------------
  Var_1 |   0.03679   1.00000   0.00000   0.00000
  Var_2 |   0.01942   0.00000   1.00000   0.00000
  Var_3 |   0.02485   0.00000   0.00000   1.00000


    print "Ex 0.3: intc=1: ihet=1 uncor=1 sqtyp=0: tau=[ 1 3 ]";
    tau = [ 1 3 ];
    pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 1; sqtyp = 0;
    optn = [ pri lead intc ihet uncor sqtyp ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;


EST=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
---------------------------------------------------------------
  Lead_0 |   1.00000   0.00000   0.00000   0.00000   1.00000
  Lead_1 |  -0.02467  -0.03246  -0.04451  -0.01073  -0.02738
  Lead_2 |   0.00242   0.00030   0.00010   0.00195  -0.00023
  Lead_3 |  -0.15225   0.02240  -0.00580   0.07529  -0.01507
```

```
  Lead_4 |   0.00160   0.01018   0.01235  -0.00221   0.00093
  Lead_5 |  -0.00089   0.00009   0.00010  -0.00026   0.00012


         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   1.00000
  Lead_1 |  -0.04249  -0.03279   0.03131   0.05340
  Lead_2 |  -0.00063  -0.00128   0.00188   0.00298
  Lead_3 |  -0.09679   0.07659  -0.13703  -0.07969
  Lead_4 |  -0.00178   0.01363  -0.00976  -0.00894
  Lead_5 |   0.00003   0.00063  -0.00092  -0.00108


ASE=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
-----------------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.07385   0.06916   0.07486   0.06948   0.07358
  Lead_2 |   0.00527   0.00518   0.00524   0.00591   0.00607
  Lead_3 |   0.07101   0.07005   0.07816   0.06930   0.06781
  Lead_4 |   0.02512   0.01799   0.02129   0.01645   0.01527
  Lead_5 |   0.00203   0.00172   0.00228   0.00183   0.00141


         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.07843   0.06274   0.06483   0.07103
  Lead_2 |   0.00589   0.00520   0.00548   0.00895
  Lead_3 |   0.07833   0.06384   0.06627   0.06800
  Lead_4 |   0.01549   0.01780   0.01481   0.01812
  Lead_5 |   0.00202   0.00221   0.00193   0.00256
```

```
print "Ex 1.2: intc=1: ihet=1 uncor=0 sqtyp=1: tau=[ 1 3 ]";
tau = [ 1 3 ];
pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 0; sqtyp = 1;
optn = [ pri lead intc ihet uncor sqtyp ];
< est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
print "EST=",est;
print "ASE=",ase;
print "R2S=",r2s;
print "COV=",cov;
print "Resi=",resi;
```

```
EST=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
---------------------------------------------------------------
  Lead_0 |   0.28801   0.00000   0.00000   0.00000   0.27914
  Lead_1 |  -0.00711  -0.00935  -0.01282  -0.00299  -0.00764
  Lead_2 |   0.00070   0.00009   0.00003   0.00054  -0.00006
  Lead_3 |  -0.04385   0.00645  -0.00167   0.02102  -0.00421
  Lead_4 |   0.00046   0.00293   0.00356  -0.00062   0.00026
  Lead_5 |  -0.00026   0.00003   0.00003  -0.00007   0.00003


         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.25827
  Lead_1 |  -0.01186  -0.00847   0.00809   0.01379
  Lead_2 |  -0.00018  -0.00033   0.00049   0.00077
  Lead_3 |  -0.02702   0.01978  -0.03539  -0.02058
  Lead_4 |  -0.00050   0.00352  -0.00252  -0.00231
  Lead_5 |   0.00001   0.00016  -0.00024  -0.00028


ASE=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
---------------------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.02127   0.01992   0.02156   0.01939   0.02054
  Lead_2 |   0.00152   0.00147   0.00155   0.00166   0.00173
  Lead_3 |   0.02045   0.02018   0.02251   0.01935   0.01894
  Lead_4 |   0.00730   0.00538   0.00647   0.00467   0.00434
  Lead_5 |   0.00059   0.00050   0.00068   0.00052   0.00043


         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.02189   0.01620   0.01674   0.01834
  Lead_2 |   0.00167   0.00138   0.00141   0.00234
  Lead_3 |   0.02186   0.01648   0.01711   0.01756
  Lead_4 |   0.00448   0.00483   0.00389   0.00491
  Lead_5 |   0.00061   0.00059   0.00053   0.00071
```

```
print "Ex 1.3: intc=1: ihet=1 uncor=1 sqtyp=1: tau=[ 1 3 ]";
tau = [ 1 3 ];
pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 1; sqtyp = 1;
optn = [ pri lead intc ihet uncor sqtyp ];
< est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
```

```
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;
```

EST=

	Var_1_1	Var_1_2	Var_1_3	Var_2_1	Var_2_2
Lead_0	0.28801	0.00000	0.00000	0.00000	0.27914
Lead_1	-0.00711	-0.00935	-0.01282	-0.00299	-0.00764
Lead_2	0.00070	0.00009	0.00003	0.00054	-0.00006
Lead_3	-0.04385	0.00645	-0.00167	0.02102	-0.00421
Lead_4	0.00046	0.00293	0.00356	-0.00062	0.00026
Lead_5	-0.00026	0.00003	0.00003	-0.00007	0.00003

	Var_2_3	Var_3_1	Var_3_2	Var_3_3
Lead_0	0.00000	0.00000	0.00000	0.25827
Lead_1	-0.01186	-0.00847	0.00809	0.01379
Lead_2	-0.00018	-0.00033	0.00049	0.00077
Lead_3	-0.02702	0.01978	-0.03539	-0.02058
Lead_4	-0.00050	0.00352	-0.00252	-0.00231
Lead_5	0.00001	0.00016	-0.00024	-0.00028

ASE=

	Var_1_1	Var_1_2	Var_1_3	Var_2_1	Var_2_2
Lead_0	0.00000	0.00000	0.00000	0.00000	0.00000
Lead_1	0.02127	0.01992	0.02156	0.01939	0.02054
Lead_2	0.00152	0.00149	0.00151	0.00165	0.00169
Lead_3	0.02045	0.02018	0.02251	0.01934	0.01893
Lead_4	0.00724	0.00518	0.00613	0.00459	0.00426
Lead_5	0.00059	0.00050	0.00066	0.00051	0.00039

	Var_2_3	Var_3_1	Var_3_2	Var_3_3
Lead_0	0.00000	0.00000	0.00000	0.00000
Lead_1	0.02189	0.01620	0.01674	0.01834
Lead_2	0.00165	0.00134	0.00142	0.00231
Lead_3	0.02186	0.01649	0.01711	0.01756
Lead_4	0.00432	0.00460	0.00382	0.00468
Lead_5	0.00056	0.00057	0.00050	0.00066

```
    print "Ex 2.2: intc=1: ihet=1 uncor=0 sqtyp=2: tau=[ 1 3 ]";
    tau = [ 1 3 ];
    pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 0; sqtyp = 2;
    optn = [ pri lead intc ihet uncor sqtyp ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;
```

EST=

	Var_1_1	Var_1_2	Var_1_3	Var_2_1	Var_2_2
Lead_0	0.28801	0.00311	-0.04061	0.00311	0.27913
Lead_1	-0.00581	-0.01071	-0.01512	-0.00302	-0.00779
Lead_2	0.00075	0.00001	-0.00009	0.00055	-0.00007
Lead_3	-0.04672	0.01197	0.00126	0.02043	-0.00394
Lead_4	-0.00010	0.00333	0.00391	-0.00063	0.00031
Lead_5	-0.00028	0.00006	0.00007	-0.00008	0.00004

	Var_2_3	Var_3_1	Var_3_2	Var_3_3
Lead_0	-0.00143	-0.04061	-0.00143	0.25505
Lead_1	-0.01207	-0.00735	0.00934	0.01549
Lead_2	-0.00018	-0.00043	0.00047	0.00076
Lead_3	-0.02692	0.02561	-0.03584	-0.01995
Lead_4	-0.00045	0.00342	-0.00290	-0.00278
Lead_5	0.00001	0.00020	-0.00024	-0.00028

ASE=

	Var_1_1	Var_1_2	Var_1_3	Var_2_1	Var_2_2
Lead_0	0.00000	0.00000	0.00000	0.00000	0.00000
Lead_1	0.02153	0.02042	0.02216	0.01940	0.02053
Lead_2	0.00144	0.00142	0.00174	0.00166	0.00173
Lead_3	0.02087	0.02063	0.02300	0.01935	0.01894
Lead_4	0.00749	0.00563	0.00683	0.00462	0.00431
Lead_5	0.00061	0.00054	0.00076	0.00052	0.00043

	Var_2_3	Var_3_1	Var_3_2	Var_3_3

```
-------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.02188   0.01640   0.01712   0.01878
  Lead_2 |   0.00166   0.00127   0.00134   0.00241
  Lead_3 |   0.02187   0.01682   0.01742   0.01799
  Lead_4 |   0.00442   0.00509   0.00418   0.00531
  Lead_5 |   0.00060   0.00059   0.00056   0.00076
```

```
    print "Ex 2.3: intc=1: ihet=1 uncor=1 sqtyp=2: tau=[ 1 3 ]";
    tau = [ 1 3 ];
    pri = 2; lead = 5; intc = 1; ihet = 1; uncor = 1; sqtyp = 2;
    optn = [ pri lead intc ihet uncor sqtyp ];
    < est,ase,r2s,cov,resi > = tsmeas(ran203,"iresp",tau,optn);
    print "EST=",est;
    print "ASE=",ase;
    print "R2S=",r2s;
    print "COV=",cov;
    print "Resi=",resi;
```

```
EST=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
-------------------------------------------------------------
  Lead_0 |   0.28801   0.00311  -0.04061   0.00311   0.27913
  Lead_1 |  -0.00581  -0.01071  -0.01512  -0.00302  -0.00779
  Lead_2 |   0.00075   0.00001  -0.00009   0.00055  -0.00007
  Lead_3 |  -0.04672   0.01197   0.00126   0.02043  -0.00394
  Lead_4 |  -0.00010   0.00333   0.00391  -0.00063   0.00031
  Lead_5 |  -0.00028   0.00006   0.00007  -0.00008   0.00004

         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------
  Lead_0 |  -0.00143  -0.04061  -0.00143   0.25505
  Lead_1 |  -0.01207  -0.00735   0.00934   0.01549
  Lead_2 |  -0.00018  -0.00043   0.00047   0.00076
  Lead_3 |  -0.02692   0.02561  -0.03584  -0.01995
  Lead_4 |  -0.00045   0.00342  -0.00290  -0.00278
  Lead_5 |   0.00001   0.00020  -0.00024  -0.00028

ASE=
         |   Var_1_1   Var_1_2   Var_1_3   Var_2_1   Var_2_2
-------------------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.02142   0.02009   0.02176   0.01940   0.02054
```

```
  Lead_2 |   0.00143   0.00144   0.00167   0.00165   0.00169
  Lead_3 |   0.02062   0.02036   0.02268   0.01934   0.01893
  Lead_4 |   0.00737   0.00533   0.00638   0.00455   0.00424
  Lead_5 |   0.00059   0.00053   0.00072   0.00051   0.00039

         |   Var_2_3   Var_3_1   Var_3_2   Var_3_3
-------------------------------------------------------
  Lead_0 |   0.00000   0.00000   0.00000   0.00000
  Lead_1 |   0.02189   0.01628   0.01677   0.01837
  Lead_2 |   0.00164   0.00123   0.00134   0.00237
  Lead_3 |   0.02186   0.01654   0.01714   0.01763
  Lead_4 |   0.00428   0.00480   0.00399   0.00495
  Lead_5 |   0.00056   0.00057   0.00052   0.00071
```

4.2 Extension to the `tstrans` Function

We have added a number of more functions:

"boxc" Box-Cox Transform

"lagd" Lag Difference Transform

"logd" Log Difference Transform

"baki" Baxter-King Filtering

"hopr" Hodrick-Prescott Filtering

4.2.1 Box-Cox, Lag- and Log-Difference Transforms

The old form of the `tstrans()` function

ydat = tstrans(xdat,"sopt"<,optn>)

was extended with an optional input argument `add` to

ydat = tstrans(xdat,"sopt"<,optn<,add>>)

An additional argument was necessary to compute Box-Cox, Log and Lag Difference transformations. Only these three new transforms use the additional input argument. The following are the new `"sopt"` transformations:

"boxc" Box-Cox transform: the input argument specifies a real scalar or a vector with λ values:

$$y = \begin{cases} x^{\lambda-1}/\lambda & \text{for } \lambda \neq 0 \\ \log(x) & \text{for } \lambda = 0 \text{ and } xmin > 0 \\ \log(x + xmin + 1) & \text{for } \lambda = 0 \text{ and } xmin \leq 0 \end{cases}$$

"lagd" Lag difference transform: the last input argument specifies an integer scalar or a vector with lag values:

$$y_i = x_{i+lag} - x_i \quad , \quad i = 1, \ldots, n - lag$$

"logd" Log difference transform: the last input argument specifies an integer scalar or a vector with lag values:

$$y_i = \log(x_{i+lag}) - \log(x_i) \quad , \quad i = 1, \ldots, n - lag$$

"boxc" Example for Box-Cox transform:

```
xd1 = [ 0.9501  0.2311  0.6068  0.4860  0.8913
        0.7621  0.4565  0.0185  0.8214  0.4447
        0.6154  0.7919  0.9218  0.7382  0.1763
        0.4057  0.9355  0.9169  0.4103  0.8936 ];


optn = [ 2 ]; lamb = [ 1. 0. 2. ];
trans51 = tstrans(xd1,"boxc",optn,lamb);
print "Trans51=",trans51;
```

```
Trans51=
   |         1         2         3
-----------------------------------
 1 |  -0.04990  -0.05119  -0.04865
 2 |  -0.76890   -1.4649  -0.47330
 3 |  -0.39320  -0.49956  -0.31590
 4 |  -0.51400  -0.72155  -0.38190
 5 |  -0.10870  -0.11507  -0.10279
 6 |  -0.23790  -0.27168  -0.20960
 7 |  -0.54350  -0.78417  -0.39580
 8 |  -0.98150   -3.9900  -0.49983
 9 |  -0.17860  -0.19675  -0.16265
10 |  -0.55530  -0.81036  -0.40112
11 |  -0.38460  -0.48548  -0.31064
12 |  -0.20810  -0.23332  -0.18645
13 |  -0.07820  -0.08143  -0.07514
14 |  -0.26180  -0.30354  -0.22753
15 |  -0.82370   -1.7356  -0.48446
16 |  -0.59430  -0.90214  -0.41770
17 |  -0.06450  -0.06667  -0.06242
18 |  -0.08310  -0.08676  -0.07965
19 |  -0.58970  -0.89087  -0.41583
20 |  -0.10640  -0.11250  -0.10074
```

"lagd" Example for Lag difference transform:

```
optn = [ 2 ]; lags = [ 0 1 2 ];
trans61 = tstrans(xd1,"lagd",optn,lags);
print "Trans61=",trans61;
```

```
Trans61=
   |         1         2         3
-----------------------------------
 1 |   0.00000  -0.71900  -0.34330
 2 |   0.00000   0.37570   0.25490
 3 |   0.00000  -0.12080   0.28450
 4 |   0.00000   0.40530   0.27610
 5 |   0.00000  -0.12920  -0.43480
 6 |   0.00000  -0.30560  -0.74360
 7 |   0.00000  -0.43800   0.36490
 8 |   0.00000   0.80290   0.42620
 9 |   0.00000  -0.37670  -0.20600
10 |   0.00000   0.17070   0.34720
11 |   0.00000   0.17650   0.30640
12 |   0.00000   0.12990  -0.05370
13 |   0.00000  -0.18360  -0.74550
14 |   0.00000  -0.56190  -0.33250
15 |   0.00000   0.22940   0.75920
16 |   0.00000   0.52980   0.51120
17 |   0.00000  -0.01860  -0.52520
18 |   0.00000  -0.50660  -0.02330
19 |   0.00000   0.48330     .
20 |   0.00000     .         .
```

"logd" Example for Log difference transform:

```
optn = [ 2 ];  lags = [ 0 1 2 ];
trans71 = tstrans(xd1,"logd",optn,lags);
print "Trans71=",trans71;
```

```
Trans71=

   |         1         2         3
-----------------------------------
 1 |   -1e-018   -1.4137  -0.44837
 2 |   -5e-017   0.96535   0.74336
```

```
 3 |    1e-017  -0.22199   0.38448
 4 |    3e-017   0.60647   0.44987
 5 |    6e-018  -0.15660  -0.66909
 6 |    1e-017  -0.51249   -3.7183
 7 |    2e-017   -3.2058   0.58742
 8 |    1e-016    3.7932    3.1796
 9 |    1e-017  -0.61361  -0.28874
10 |   -3e-017   0.32487   0.57704
11 |   -5e-018   0.25216   0.40406
12 |   -6e-018   0.15189  -0.07022
13 |    6e-018  -0.22211   -1.6541
14 |   -3e-017   -1.4320  -0.59860
15 |    6e-017   0.83343    1.6689
16 |   -4e-017   0.83547   0.81538
17 |   -2e-018  -0.02008  -0.82419
18 |    4e-019  -0.80411  -0.02574
19 |   -2e-017   0.77837     .
20 |   -5e-018     .         .
```

4.2.2 Baxter-King and Hodrick-Prescott Filtering

< ydat,cycl,noise > = tstrans(xdat,"baki"<,optn>)

The following specific options could be specified.

2 Number of periods to use in the higher frequency filter (e.g. 6 for quarterly data). Must be at least 2. The default is 6.

3 Number of periods to use in the lower frequency filter (e.g. 32 for quarterly data). There must be [3] >= [2]. The default is 32.

4 Number of points to use in the finite approximation bandpass filter. The filter throws away the first and last K points. The default is 12.

```
xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ]';
 xc = xd2 - univar(xd2,"ari");


  nphi = 6; nplo = 18; nkpt = 12;
```

```
optn = [ 2 nphi nplo nkpt ];
< trnd,cycl,nois > = tstrans(xc,"baki",optn);
print "Trend,Cycle,Noise=",trnd -> cycl -> nois;
```

```
Trend,Cycle,Noise=
```

L	1	2	3
1	0.24741	0	0
2	0.33851	0.00000	0
3	-0.44029	0.00000	0.00000
4	0.34611	0.00000	0.00000
5	0.06511	0.00000	0.00000
6	-0.46979	0.00000	0.00000
7	-0.28879	0.00000	0.00000
8	-0.02039	0.00000	0.00000
9	0.39021	0.00000	0.00000
10	0.39761	0.00000	0.00000
11	-0.40969	0.00000	0.00000
12	0.40331	0.00000	0.00000
13	0.06946	0.08202	0.23842
14	0.08122	-0.07580	-0.08732
15	0.09513	-0.19182	0.32970
16	0.09570	-0.17032	-0.35077
17	0.10864	-0.07158	-0.18256
18	0.12467	0.03503	0.18871
19	0.10820	0.12761	-0.01091
20	0.11169	0.08479	0.19573
21	0.11225	-0.01655	-0.00730
22	0.12558	-0.11698	-0.54019
23	0.13092	-0.11240	0.26328
24	0.08668	0.05017	0.22986
25	0.06173	0.15960	-0.10992
26	0.04178	0.16433	-0.01571
27	-0.01183	0.13531	0.05233
28	-0.17509	0.00000	0.00000
29	0.08821	0.00000	0.00000
30	-0.39609	0.00000	0.00000
31	0.13871	0.00000	0.00000
32	-0.53549	0.00000	0.00000
33	-0.29039	0.00000	0.00000
34	-0.52109	0.00000	0.00000
35	-0.47019	0.00000	0.00000
36	0.25621	0.00000	0.00000
37	0.12751	0.00000	0.00000

```
38 |  -0.25019   0.00000   0.00000
39 |   0.38291   0.00000   0.00000
40 |  -0.53289   0.00000   0.00000
```

< ydat,cycl > = tstrans(xdat,"hopr"<,optn>)

The following specific options could be specified.

2 Positive, scalar integer containing the smoothing parameter of the HP filter. The default is 1600.

```
xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ]';
 xc = xd2 - univar(xd2,"ari");


  tlam = 100;
  optn = [ 2 tlam ];
  < trnd,cycl > = tstrans(xc,"hopr",optn);
  print "Trend,Cycle=",trnd -> cycl;
```

```
Trend,Cycle=

   |         1         2
------------------------
 1 |   0.09799   0.14942
 2 |   0.07174   0.26677
 3 |   0.04698  -0.48727
 4 |   0.02787   0.31824
 5 |   0.01371   0.05140
 6 |   0.00697  -0.47676
 7 |   0.01063  -0.29942
 8 |   0.02291  -0.04330
 9 |   0.03903   0.35118
10 |   0.05379   0.34382
11 |   0.06548  -0.47518
12 |   0.07586   0.32745
13 |   0.08189   0.30801
```

```
14 |   0.08386  -0.16575
15 |   0.08510   0.14791
16 |   0.08730  -0.51269
17 |   0.09362  -0.23912
18 |   0.10211   0.24630
19 |   0.10841   0.11650
20 |   0.11062   0.28159
21 |   0.10803  -0.01962
22 |   0.10272  -0.63431
23 |   0.09659   0.18522
24 |   0.08519   0.28151
25 |   0.06593   0.04548
26 |   0.03902   0.15139
27 |   0.00514   0.17067
28 |  -0.03353  -0.14156
29 |  -0.07309   0.16130
30 |  -0.11107  -0.28503
31 |  -0.14336   0.28207
32 |  -0.16874  -0.36675
33 |  -0.18313  -0.10726
34 |  -0.18615  -0.33494
35 |  -0.17847  -0.29172
36 |  -0.16412   0.42033
37 |  -0.15004   0.27755
38 |  -0.13898  -0.11121
39 |  -0.13091   0.51381
40 |  -0.12689  -0.40600
```

4.3 Extension to the `nlp` Function

4.3.1 New Techniques

nondif subgradient *bundle trust region* methods,

simann *Simulated Annealing* for global optmization (Goffe et.al, 1994).

4.3.2 New Option

The `nlec` option was added as an alternative to the `nlcb` option permitting a second form of nonlinear constraint specification: Assuming `nlcon` is the name of a user specified function evaluating n_{nlc} nonlinear equality or inequality constraints $c = (c_1, \ldots, n_{nlc})^T$ for given variables (points) $x = (x_1, \ldots, x_n)^T$. There are two different forms for specifying nonlinear constraints:

- The $n_{nlc} = n_{nlec} + n_{nlic}$ nonlinear equality and inequality constraints can

always be specified with the function `nlcon` in the form:

$$\begin{aligned} c_j(x) &= 0 \quad \text{for } 1 \le j \le n_{nlec} \\ c_j(x) &\ge 0 \quad \text{for } n_{nlec} < j \le n_{nlc} \end{aligned}$$

If the function `nlcon` specifies the constraints in this form, the `nlec` option can be used for specifying the number n_{nlec} of constraints equal to zero. If there are no equality constraints, $n_{nlec} = 0$, the `nlec` option must not be specified since it defaults to zero.

- If for some reason you want to specify the constraints with the `nlcon` module in the form:

$$nllb_j \le c_j(x) \le nlub_j \quad \text{for } j = 1, \ldots, n_{nlc}$$

then you can specify the lower $nnlb_j$ and upper $nnub_j$ bounds using a two column matrix `nlcb`. This object must be specified using the `nlcb` option. The order of the n_{nlc} constraint functions specified in `nlcon` must correspond with the order of rows in the lower-upper-bound matrix specified with the `nlcb` option.

- If you specify both, the `nlec` and the `nlcb` options, the `nlcb` option will be used. However, using the `nlec` option and not `nlcb` is recommended.

The module `nlcon` must have the form `c = nlcon(x)`. During the execution of `nlp`, the argument `x` is set with trial values and the returned values of y are used to reduce the constraint violations when approximating the optimal location x^*. Note, that the definition of `nlcon` can have additional arguments `x2,...,xm` defined with the `global` clause. No nonlinear constraints are imposed, if the `nlcon` function is not specified.

5 New Developments

5.1 Function `arhetero`

gof = arhetero(y,lagbnd<,optn>)

< gof,est,root,ase,nwcov,cov,resi > = arhet(y,lagbnd<,optn>)

Purpose: The `arhetero` function implements the modeling step for the Heterogeneous AutoRegressive method. There are two versions implemented (see MFE Toolbox):

original version using overlapping lag intervals,

modified version with non-overlapping lag intervals.

Input: **y** must be an N vector of time series data;

lagbnd can be a positive integer scalar or a p vector of positive positive upper bounds (assuming lower bounds of 1), or a $p \times 2$ matrix with lower (in first column) and upper bounds (in second column);

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
`"alpha"`	real	probability for confidence intervals, default=.05
`"cent"`		centers the y data and as a consequnce
		does not estimate the intercept
`"meth"`	string	method of estimation
	"ori"	original (overlapping) lag intervals (default)
	"mod"	modified (non-overlapping) lag intervals
`"noint"`		does not estimate the intercept
`"print"`	int	amount of printed output (def=0: no output)
`"nopr"`		do not print any output
`"nwlag"`	int	number of lags for Newey-West cov matrix (def=0: no lags)

Output: **gof** is a vector of scalar results currently containing:

1. indicator for problems in the algorithm (=0: no problems)
2. mean value μ of input data y
3. standard deviation σ of input data y
4. variance σ^2
5. computation time in seconds
6. error sum of squares
7. Akaikes information criterion AIC

8. Bayesian (Schwartz) information criterion BIC

est is a vector of ML parameter estimates; the order of the estimates is:

1. intercept (optional)
2. nar parameters for the p lag intervals

root the magnitude (absolute values) of roots for the $\max_j ar_j$ lags;

ase the asymptotic standard errors of the p estimates from the diagonal of the $p \times p$ covariance matrix;

nwcov the $p \times p$ (robust) Newey-West covariance matrix of the parameter estimates for a specified number of lags;

cov the $p \times p$ asymptotic covariance matrix of the parameter estimates;

resi is an N vector of residuals.

Restrictions: 1. The time series data y cannot have any missing values or string data. If x is specified as nonmissing, then the same is true for the predictors x.

2. The lag numbers in `lagbnd` must be positive integers.

Relationships: arma(), armafore(), arima(), burg(), tsmeas()

Examples: 1. Lag Intervals with Vector Input, Original Version:

```
print "Robust NW Covariance Matrix without Lags";
plag = [ 1 5 22 ];
optn = [ "print"     2 ];
< gof,est,root,ase,nwcov,cov,resi > = arhetero(yy,plag,optn);
print "GOF=",gof;
print "PARM=",est;
print "Root=",root;
print "ASE=",ase;
print "NWCOV=",nwcov;
print "COV=",cov;
print "RESI[20:30]=",resi[20:30];
```

```
*******************************************************
Heterogeneous AR Modeling: Original Version (Uncentered)
*******************************************************

Number of Observations . . . . . .      1000
Number of Parameter Estimates. . . . . .         4
Mean Y . . . . . .     20.7847
Standard Deviation Y . . . . . .      1.1768
Mean Squared Error (MSE) . . . . . .      1.3849
Number of Newey-West Lags. . . . . .         0
```

Original (Overlapping) Lag Intervals

	Begin	End
AR_1	1	1
AR_2	1	5
AR_3	1	22

Standard Error Regression= 1.0037
BIC=0.00970143 AIC=-0.00992959

****
AR Roots and Coefficients
****

N	Coefficients	RealRoot	ImagRoot	Abs_Root
0	1.000000000			
1	-0.074546642	0.986396377	0.000000000	0.986396377
2	-0.086335537	-0.702542050	0.440720754	0.829337154
3	-0.086335537	0.852923558	-0.278322611	0.897185640
4	-0.086335537	-0.360801238	0.744403873	0.827233135
5	-0.086335537	0.319003199	-0.784422842	0.846807083
6	-0.027232902	0.527981456	0.664972164	0.849089158
7	-0.027232902	0.527981456	-0.664972164	0.849089158
8	-0.027232902	0.319003199	0.784422842	0.846807083
9	-0.027232902	0.709173134	-0.491738826	0.862979494
10	-0.027232902	0.094137249	-0.837687446	0.842960307
11	-0.027232902	-0.360801238	-0.744403873	0.827233135
12	-0.027232902	-0.137630194	-0.821719040	0.833165200
13	-0.027232902	-0.137630194	0.821719040	0.833165200
14	-0.027232902	0.709173134	0.491738826	0.862979494
15	-0.027232902	-0.791989818	-0.230843482	0.824946413
16	-0.027232902	-0.555251096	-0.614614010	0.828283865
17	-0.027232902	-0.791989818	0.230843482	0.824946413
18	-0.027232902	-0.702542050	-0.440720754	0.829337154
19	-0.027232902	0.852923558	0.278322611	0.897185640
20	-0.027232902	0.094137249	0.837687446	0.842960307
21	-0.027232902	-0.555251096	0.614614010	0.828283865
22	-0.027232902	-0.821858136	0.000000000	0.821858136

Parameter Estimates
****

Dense Row Vector (ncol=4)

```
R |   Intercept       AR_1       AR_2       AR_3
      2.4289563 -0.0117889  0.2955132  0.5991238


   Robust (Newey-West) Asymptotic Standard Errors
   **********************************************

               Dense Matrix (4 by 3)

              |   A_St_Err.    T_Value    P_Value
            ------------------------------------
  Intercept |   0.9738120  2.4942763  0.0063913
      AR_1 |   0.0360480 -0.3270335  0.6281444
      AR_2 |   0.0820911  3.5998212  1.67e-004
      AR_3 |   0.0859935  6.9670817  2.94e-012


 Newey-West Covariance Matrix of Estimates (0 Lags)
 **************************************************

          Dense Symmetric Matrix (4 by 4)

       S |   Intercept       AR_1       AR_2       AR_3
       ------------------------------------------------
Intercept |   0.9483098
     AR_1 |  -0.0019482  0.0012995
     AR_2 |   9.91e-005 -0.0013292  0.0067389
     AR_3 |  -0.0438759  1.24e-004 -0.0054079  0.0073949


       Covariance Matrix of Parameter Estimates
       ****************************************

          Dense Symmetric Matrix (4 by 4)

       S |   Intercept       AR_1       AR_2       AR_3
       ------------------------------------------------
Intercept |   0.9181031
     AR_1 |  -2.56e-004  0.0013961
     AR_2 |  -0.0019961 -0.0014799  0.0074492
     AR_3 |  -0.0418698  9.62e-005 -0.0058720  0.0077889
```

2. Lag Intervals with Matrix Specification, Original Version:

```
print "Matrix Specification of LAG Intervals: > 1 Start";
plag = [ 1 1, 2 2, 10 22 ];
```

```
optn = [ "print"       2 ];
< gof,est,root,ase,nwcov,cov,resi > = arhetero(yy,plag,optn);
print "GOF=",gof;
print "PARM=",est;
print "Root=",root;
print "ASE=",ase;
print "NWCOV=",nwcov;
print "COV=",cov;
print "RESI[20:30]=",resi[20:30];
```

```
   ******************************************************
   Heterogeneous AR Modeling: Original Version (Uncentered)
   ******************************************************

  Number of Observations . . . . . .      1000
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .                    20.7847
  Standard Deviation Y . . . . . .       1.1768
  Mean Squared Error (MSE) . . . . . .   1.3849
  Number of Newey-West Lags. . . . . .       0

            Original (Overlapping) Lag Intervals

                        Begin          End
              AR_1          1            1
              AR_2          2            2
              AR_3         10           22

            Standard Error Regression= 1.02833
                BIC=0.0581787 AIC=0.0385477

                   Parameter Estimates
                   *******************

                Dense Row Vector (ncol=4)

        R |   Intercept       AR_1       AR_2       AR_3
              3.8495117  0.1228098  0.0785620  0.6130674

        Robust (Newey-West) Asymptotic Standard Errors
        **********************************************

                   Dense Matrix (4 by 3)
```

```
                |   A_St_Err.    T_Value    P_Value
              ------------------------------------
      Intercept |   0.9693597  3.9711902  3.83e-005
           AR_1 |   0.0309529  3.9676305  3.89e-005
           AR_2 |   0.0319426  2.4594742  0.0070416
           AR_3 |   0.0562102  10.906701  0.0000000


  Newey-West Covariance Matrix of Estimates (0 Lags)
  **************************************************

           Dense Symmetric Matrix (4 by 4)

        S |   Intercept       AR_1       AR_2       AR_3
        ---------------------------------------------------
Intercept |   0.9396582
     AR_1 |  -0.0046478  9.58e-004
     AR_2 |  -0.0036824 -9.88e-005  0.0010203
     AR_3 |  -0.0369353 -6.33e-004 -7.46e-004  0.0031596
```

3. Newey West Covariance Matrix with Lags:

```
print "With Newey-West Covariance Matrix: Default Lags";
plag = [ 1 5 22 ];
tt = (real)ncol(yy); tt = ceil(pow(tt,1. / 3.));
optn = [ "print"     2 ,
         "nwlag"    tt ];
< gof,est,root,ase,nwcov,cov,resi > = arhetero(yy,plag,optn);
print "GOF=",gof;
print "PARM=",est;
print "Root=",root;
print "ASE=",ase;
print "NWCOV=",nwcov;
print "COV=",cov;
print "RESI[20:30]=",resi[20:30];


   ********************************************************
   Heterogeneous AR Modeling: Original Version (Uncentered)
   ********************************************************

  Number of Observations . . . . . .      1000
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .                    20.7847
  Standard Deviation Y . . . . . .       1.1768
```

```
Mean Squared Error (MSE) . . . . . .      1.3849
Number of Newey-West Lags . . . . . .       10
```

Original (Overlapping) Lag Intervals

	Begin	End
AR_1	1	1
AR_2	1	5
AR_3	1	22

Standard Error Regression= 1.0037
BIC=0.00970143 AIC=-0.00992959

Parameter Estimates
****

Dense Row Vector (ncol=4)

R	Intercept	AR_1	AR_2	AR_3
	2.4289563	-0.0117889	0.2955132	0.5991238

Robust (Newey-West) Asymptotic Standard Errors
****

Dense Matrix (4 by 3)

	A_St_Err.	T_Value	P_Value
Intercept	0.9452535	2.5696347	0.0051625
AR_1	0.0319191	-0.3693364	0.6440222
AR_2	0.0807807	3.6582162	1.34e-004
AR_3	0.0898222	6.6701070	2.12e-011

Newey-West Covariance Matrix of Estimates (10 Lags)
****

Dense Symmetric Matrix (4 by 4)

S	Intercept	AR_1	AR_2	AR_3
Intercept	0.8935043			
AR_1	-0.0032674	0.0010188		
AR_2	0.0064259	-9.20e-004	0.0065255	

```
      AR_3 |  -0.0461011  5.98e-005 -0.0059129  0.0080680

         Covariance Matrix of Parameter Estimates
         ****************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept       AR_1       AR_2       AR_3
         -----------------------------------------------
 Intercept |   0.9181031
      AR_1 |  -2.56e-004  0.0013961
      AR_2 |  -0.0019961 -0.0014799  0.0074492
      AR_3 |  -0.0418698  9.62e-005 -0.0058720  0.0077889
```

4. Modified version: Non-overlapping Lag Intervals

```
print "Use Modified Version";
plag = [ 1 1, 1 5, 1 22 ];
optn = [ "print"     2 ,
         "meth"  "mod" ];
< gof,est,root,ase,nwcov,cov,resi > = arhetero(yy,plag,optn);
print "GOF=",gof;
print "PARM=",est;
print "Root=",root;
print "ASE=",ase;
print "NWCOV=",nwcov;
print "COV=",cov;
print "RESI[20:30]=",resi[20:30];
```

```
   ********************************************************
   Heterogeneous AR Modeling: Modified Version (Uncentered)
   ********************************************************

  Number of Observations . . . . . .      1000
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .                    20.7847
  Standard Deviation Y . . . . . .       1.1768
  Mean Squared Error (MSE) . . . . . .   1.3849
  Number of Newey-West Lags. . . . . .       0

           Original (Overlapping) Lag Intervals

                      Begin          End
    AR_1                  1            1
```

AR_2	1	5
AR_3	1	22

Modified (Non-Overlapping) Lag Intervals

	Begin	End
AR_1	1	1
AR_2	2	5
AR_3	6	22

Standard Error Regression= 1.0037
BIC=0.00970143 AIC=-0.00992959

Parameter Estimates
****

Dense Row Vector (ncol=4)

R	Intercept	AR_1	AR_2	AR_3
	2.4289563	0.0745466	0.3453421	0.4629593

Robust (Newey-West) Asymptotic Standard Errors
****

Dense Matrix (4 by 3)

	A_St_Err.	T_Value	P_Value
Intercept	0.9738120	2.4942763	0.0063913
AR_1	0.0310735	2.3990420	0.0083105
AR_2	0.0546277	6.3217440	1.95e-010
AR_3	0.0664495	6.9670817	2.94e-012

Newey-West Covariance Matrix of Estimates (0 Lags)
****

Dense Symmetric Matrix (4 by 4)

S	Intercept	AR_1	AR_2	AR_3
Intercept	0.9483098			
AR_1	-0.0039228	9.66e-004		
AR_2	-0.0078981	-2.95e-004	0.0029842	

```
      AR_3 |  -0.0339041 -4.80e-004 -0.0023041  0.0044155

         Covariance Matrix of Parameter Estimates
         ****************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept       AR_1       AR_2       AR_3
         ---------------------------------------------------
 Intercept |   0.9181031
      AR_1 |  -0.0025589  0.0010202
      AR_2 |  -0.0092096 -3.37e-004  0.0033167
      AR_3 |  -0.0323540 -5.60e-004 -0.0025356  0.0046508
```

5.2 Function `arma`

gof = arma(y<,x<,ar<,ma<,optn<,p0> . >)

< gof,est,root,ase,cis,cov,sco > = arma(y<,x<,ar<,ma<,optn <,p0> . >)

Purpose: The `arma` function implements the modeling step for the AutoRegressive Moving-Average method. It computes ML estimates and the (robust) asymptotic covariance matrix of estimates. In a second step, the optimal estimates can be used for computing predicted values using the `armafore` function.

Input: **y** must be an N vector of time series data;

x [optional] could be a $N \times n$ matrix of predictors; can be specified as missing;

ar can be a positive integer scalar or vector of positive integers with the *nar* AR lag numbers, default is 1;

ma can be an positive integer scalar or vector of positive integers with the *nma* MA lag numbers, default is 1;

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

p0 specifies an optional p vector of starting values for the ML estimation.

In addition to many of the optimization options, like termination criteria etc., the following options may be specified:

Option	Second Column	Meaning
"alpha"	real	probability for confidence intervals, default=.05
"cent"		centers the y data and as a consequnce does not estimate the intercept
"lstech"	string	optimization technique for least squares minimization, default is Levenberg-Marquardt;
"mltech"	string	optimization technique for maximum likelihood optimization, default is Newton Raphson;
"nacor"	int	specifies number of (partial) autocorrelations
"noint"		does not estimate the intercept
"print"	int	amount of printed output (def=0: no output)
"nopr"		do not print any output

Output: **gof** is a vector of scalar results currently containing:

1. indicator for problems in the algorithm (=0: no problems)
2. mean value μ of input data y
3. standard deviation σ of input data y
4. variance σ^2
5. computation time in seconds
6. Loglikelihood
7. number of iterations for optimization
8. $L1$ norm of gradient at the optimal solution
9. error sum of squares
10. Akaikes information criterion AIC
11. Bayesian (Schwartz) information criterion BIC

est is a vector of ML parameter estimates; the order of the estimates is:

1. intercept (optional)
2. nar parameters for the AR lags,
3. nma parameters for the MA lags.

root the magnitude (absolute values) of roots for the $\max_j ar_j$ and $\max_k ma_k$ lags;

ase the asymptotic standard errors of the p estimates from the diagonal of the $p \times p$ covariance matrix;

cis confidence intervals based on the asymptotic standard errors of the p estimates

cov the $p \times p$ asymptotic covariance matrix of the ML parameter estimates

sco is an $N \times p$ matrix of scores.

Restrictions:

1. The time series data y cannot have any missing values or string data. If x is specified as nonmissing, then the same is true for the predictors x.
2. The lag numbers in `ar` and `ma` must be positive integers.

Relationships: arhetero(), armafore(), arima(), burg(), tsmeas()

Examples: The following three examples refer to the $N = 40$ data of randomly generated values:

```
xd2  = [ 0.8147 0.9058 0.1270 0.9134 0.6324
         0.0975 0.2785 0.5469 0.9575 0.9649
         0.1576 0.9706 0.9572 0.4854 0.8003
         0.1419 0.4218 0.9157 0.7922 0.9595
         0.6557 0.0357 0.8491 0.9340 0.6787
         0.7577 0.7431 0.3922 0.6555 0.1712
         0.7060 0.0318 0.2769 0.0462 0.0971
         0.8235 0.6948 0.3171 0.9502 0.0344 ];
```

1. Use simple lags ar=1 and ma=1:
   The algorithm is running two optimizations, the first is a least squares optimization computing excellent starting values for the ML optimization. That's why we able to specify two optimization methods:

```
print "Simple ARMA: LSTech=LEVMAR, MLTech=TRUREG";
ar = 1; ma = 1;
optn = [ "lstech"   "lm" ,
         "mltech"   "tr" ,
         "print"       3 ];
< gof,parm,root,ase,cis,cov,scor > = arma(xd2,.,ar,ma,optn);
print "GOF=",gof;
print "PARM=",parm;
print "Root=",root;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
print "SCOR=",scor;
```

```
          ****************************************
          ARMA Modeling: N(AR)=1 N(MA)=1 (Uncentered)
          ****************************************
                   AR[1] =    1 MA[1] =    1

   Number of Observations . . . . . .          40
   Number of Predictor Variables. . . .         0
   Number of Parameter Estimates. . . .         3
   Mean . . . . . .                        0.5673
   Standard Deviation . . . . . .           0.3323
   Mean Squared Error (MSE) . . . . . .     0.1104
   LS Optimization Technique. . . . . .     LEVMAR
```

```
ML Optimization Technique. . . . . .    TRUREG
```

When specifying the `"print" 3` option the optimization histories are printed:

```
                    ******************
                    Optimization Start
                    ******************

                   Parameter Estimates
                   -------------------

          Parameter      Estimate   Gradient

           1 Intercept 0.34996599  0.1010283
           2 AR_1      0.37292898 -0.0057028
           3 MA_1     -0.41847023 -0.0743579

        Value of Objective Function =      2.15852


            Levenberg-Marquardt Optimization
              Scaling Update of More (1978)
          Gradient Computed by Finite Differences
        CRP Jacobian Computed by Finite Differences


 Iteration Start:
 N. Variables              3        N. Equations             40
 Criterion       2.158519520        Max Grad Entry  0.101028317
 TR Radius       1.081565870


Iter rest nfun act   optcrit  difcrit maxgrad  lambda      rho
   1    0    0   0  2.158312 2.1e-004 0.02017 0.00000  0.21832
   2    0    0   0  2.157933 3.8e-004 0.01434  2e-003  0.61247
   3    0    0   0  2.157910 2.2e-005  4e-004 0.00000  0.99724
   4    0    0   0  2.157910 5.4e-008  7e-004  5e-004  0.10030
   5    0    0   0  2.157910 1.7e-007  3e-004  2e-003  0.56175
   6    0    0   0  2.157910 5.5e-009  8e-005 0.00000  0.40304
   7    0    0   0  2.157910 4.7e-009  5e-005  2e-003  0.57227
   8    0    0   0  2.157910 2.1e-010  1e-005 0.00000  0.63508

 Successful Termination After     8 Iterations
 GCONV convergence criterion satisfied.
 Criterion       2.157910208        Max Grad Entry  1.0103e-005
```

```
Ridge (lambda)  0.000000000        TR Radius       0.000373462
Act.dF/Pred.dF  0.635080549
N. Function Calls         0        N. Gradient Calls         3
N. Hessian Calls         10        Preproces. Time           0
Time for Method           0        Effective Time            0


                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

              Parameter      Estimate   Gradient

                1 Intercept 0.36010117  7.25e-007
                2 AR_1      0.35313547 -3.04e-006
                3 MA_1     -0.38641826 -1.01e-005

          Value of Objective Function =      2.15791
```

The negative LogLikelihood is minimized:

```
                  Trust Region Optimization
                Scaling Update of More (1978)
             Gradient Computed by Finite Differences
          Hessian Computed by Finite Differences (dense)
                (Using Only Function Calls)


 Iteration Start:
 N. Variables              3
 Criterion       12.22570100        Max Grad Entry  9.3640e-005
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1    0    2   0  12.22570 7.0e-010  8e-009 0.00000  1.00000

 Successful Termination After     1 Iterations
 GCONV convergence criterion satisfied.
 Criterion       12.22570100        Max Grad Entry  7.6173e-009
 Ridge (lambda)  0.000000000        TR Radius       1.000000000
 Act.dF/Pred.dF  1.000119405
 N. Function Calls         3        N. Gradient Calls         7
```

N. Hessian Calls	3	Preproces. Time	0
Time for Method	0	Effective Time	0

****
Optimization Results
****

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	0.36011365	-4.74e-009
2	AR_1	0.35311336	2.17e-009
3	MA_1	-0.38639573	-7.62e-009

Value of Objective Function = 12.2257

Hessian Matrix
****

Symmetric Matrix: Dense Storage

S	1	2	3
1	933.34733	537.45465	10.248984
2	537.45465	354.55737	51.546290
3	10.248984	51.546290	49.056449

Log Likelihood= -12.2257
SBC=-1.89313 AIC=-2.0198

Covariance Matrix 2: H = (NOBS/d) inv(G)
****

Dense Symmetric Matrix (3 by 3)

S	Intercept	AR_1	MA_1
Intercept	0.1288180		
AR_1	-0.2258580	0.3993282	
MA_1	0.2104084	-0.3724091	0.3677363

```
Factor sigma=1 (nobs=40 df=3 vard=37)
     Determinant = 8.74154e-006
Matrix has Only Positive Eigenvalues


Approximate Correlation Matrix of Parameter Estimates
*****************************************************

          Dense Symmetric Matrix (3 by 3)

             S |   Intercept       AR_1       MA_1
             -------------------------------------
     Intercept |   1.0000000
          AR_1 |  -0.9958231  1.0000000
          MA_1 |   0.9667328 -0.9718229  1.0000000

               Determinant = 0.00046211
          Matrix has Only Positive Eigenvalues


        ***********************************************
        Parameter Estimates and Asympt. Standard Errors
        ***********************************************

 Parameter   Estimate    AStdErr    T Value   P Value
 Intercept  0.3601136  0.3589123  1.0033473  0.161105
      AR_1  0.3531134  0.6319242  0.5587907  0.289836
      MA_1 -0.3863957  0.6064126 -0.6371828  0.736035


           ***********************************
           Wald Confidence Limits (Alpha=0.05)
           ***********************************

          Parameter   Estimate      Lower      Upper
          Intercept  0.3601136 -0.3433415  1.0635688
               AR_1  0.3531134 -0.8854352  1.5916620
               MA_1 -0.3863957 -1.5749427  0.8021512
          Factor sigma=1 (nobs=40 df=3 vard=37)


GOF=
          |  g-of-fit
----------------------
    Error |    0.00000
   Y_Mean |    0.56729
   Y_Stdv |    0.33231
```

```
      MSE |    0.11043
     Time |    0.00000
   LogLik |     12.226
   N_Iter |     9.0000
  MaxGrad |     8e-009
    SEReg |    0.35137
      AIC |    -2.0198
     SBIC |    -1.8931
   unused |       .
```

PARM=

	Estimates	Gradient
Intercept	0.36011	-5e-009
AR_1	0.35311	2e-009
MA_1	-0.38640	-8e-009

Root=

	1
AR_Root	0.35311
MA_Root	0.38642

ASE=

	A_St_Err.	T_Value	P_Value
Intercept	0.35891	1.0033	0.16111
AR_1	0.63192	0.55879	0.28984
MA_1	0.60641	-0.63718	0.73604

CIs=

	Wald_LCI	Wald_UCI
Intercept	-0.34334	1.0636
AR_1	-0.88544	1.5917
MA_1	-1.5749	0.80215

COV=

SYM	Intercept	AR_1	MA_1
Intercept	0.12882		
AR_1	-0.22586	0.39933	
MA_1	0.21041	-0.37241	0.36774

```
SCOR=

   |   Intercept        AR_1        MA_1
-----------------------------------------
 1 |      1e-010     -4e-011      2e-010
 2 |      2.3912      1.9481      9e-011
 3 |     -5.8243     -5.1278     -1.0839
 4 |      4.7439      1.8492     -1.0922
.........................................
37 |      2.0787      1.1384    -0.06555
38 |     -3.5523     -2.2662    -0.25670
39 |      5.8491      2.5799    -0.68075
40 |     -7.7277     -5.8226     -1.4887
```

2. Use lags ar=1,2,3 and ma=1:

```
print "Simple ARMA: Tech=LEVMAR";
ar = [ 1:3 ]; ma = 1;
optn = [ "lstech"   "lm" ,
         "mltech"   "tr" ,
         "print"       3 ];
< gof,parm,root,ase,cis,cov > = arma(xd2,.,ar,ma,optn);
print "GOF=",gof;
print "PARM=",parm;
print "Root=",root;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

Here we only report the final result:

```
                    Trust Region Optimization
                 Scaling Update of More (1978)
            Gradient Computed by Finite Differences
         Hessian Computed by Finite Differences (dense)
                   (Using Only Function Calls)


   Iteration Start:
   N. Variables             5
   Criterion       9.861684378       Max Grad Entry  0.001189907
   TR Radius       1.000000000
```

```
Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1    0    2   0  9.861684 4.2e-008  3e-007 0.00000  1.00000

 Successful Termination After     1 Iterations
 GCONV convergence criterion satisfied.
 Criterion       9.861684336        Max Grad Entry  2.8946e-007
 Ridge (lambda)  0.000000000        TR Radius       1.000000000
 Act.dF/Pred.dF  0.999854944
 N. Function Calls         3        N. Gradient Calls         7
 N. Hessian Calls          3        Preproces. Time           0
 Time for Method           0        Effective Time            0


                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

             Parameter      Estimate   Gradient

              1 Intercept 0.72978139  5.02e-008
              2 AR_1     -0.43982250  5.09e-008
              3 AR_2     -0.03502382 -3.68e-008
              4 AR_3      0.16555089  1.91e-007
              5 MA_1      0.48770070  2.89e-007

         Value of Objective Function =      9.86168


                         Hessian Matrix
                         **************

              Symmetric Matrix: Dense Storage

   S |           1          2          3          4          5
   ---------------------------------------------------------------
   1 |   178.94289  99.914095  103.26008  103.07296  4.0279008
   2 |   99.914095  103.41070  37.594099  59.708318  44.177141
   3 |   103.26008  37.594099  108.73105  41.247163 -13.278503
   4 |   103.07296  59.708318  41.247163  106.80399  6.6848378
   5 |   4.0279008  44.177141 -13.278503  6.6848378  55.149279


                  Log Likelihood= -9.86168
                   SBC=-1.74689 AIC=-1.958
```

```
          Covariance Matrix 2: H = (NOBS/d) inv(G)
          ****************************************

              Dense Symmetric Matrix (5 by 5)

        S |   Intercept       AR_1       AR_2       AR_3       MA_1
        ------------------------------------------------------------
Intercept |   0.0677439
     AR_1 |  -0.0532225  0.0727391
     AR_2 |  -0.0323532  0.0162133  0.0295282
     AR_3 |  -0.0251912  0.0076546  0.0110593  0.0254186
     MA_1 |   0.0329496 -0.0514044 -0.0048556 -0.0047101  0.0563052

           Factor sigma=1 (nobs=40 df=5 vard=35)
                 Determinant = 4.0632e-009
           Matrix has Only Positive Eigenvalues


     Approximate Correlation Matrix of Parameter Estimates
     ****************************************************

              Dense Symmetric Matrix (5 by 5)

        S |   Intercept       AR_1       AR_2       AR_3       MA_1
        ------------------------------------------------------------
Intercept |   1.0000000
     AR_1 |  -0.7581866  1.0000000
     AR_2 |  -0.7233759  0.3498402  1.0000000
     AR_3 |  -0.6070690  0.1780181  0.4036760  1.0000000
     MA_1 |   0.5335085 -0.8032331 -0.1190829 -0.1245039  1.0000000

                  Determinant = 0.0195116
           Matrix has Only Positive Eigenvalues


        ***********************************************
        Parameter Estimates and Asympt. Standard Errors
        ***********************************************

      Parameter   Estimate    AStdErr    T Value   P Value
      Intercept  0.7297814  0.2602766  2.8038686  0.004091
          AR_1 -0.4398225  0.2697019 -1.6307726  0.944047
          AR_2 -0.0350238  0.1718377 -0.2038192  0.580162
          AR_3  0.1655509  0.1594322  1.0383780  0.153107
          MA_1  0.4877007  0.2372872  2.0553185  0.023685
```

```
                  **********************************
                  Wald Confidence Limits (Alpha=0.05)
                  **********************************

               Parameter   Estimate      Lower      Upper
               Intercept  0.7297814  0.2196486  1.2399142
                    AR_1 -0.4398225 -0.9684286  0.0887836
                    AR_2 -0.0350238 -0.3718195  0.3017719
                    AR_3  0.1655509 -0.1469305  0.4780323
                    MA_1  0.4877007  0.0226264  0.9527750
                 Factor sigma=1 (nobs=40 df=5 vard=35)
```

GOF=

	g-of-fit
Error	0.00000
Y_Mean	0.56729
Y_Stdv	0.33231
MSE	0.11043
Time	1.00000
LogLik	9.8617
N_Iter	12.000
MaxGrad	3e-007
SEReg	0.35443
AIC	-1.9580
SBIC	-1.7469
unused	.

PARM=

	Estimates	Gradient
Intercept	0.72978	5e-008
AR_1	-0.43982	5e-008
AR_2	-0.03502	-4e-008
AR_3	0.16555	2e-007
MA_1	0.48770	3e-007

Root=

	1	2	3
AR_Root	0.79557	0.79557	0.69489
MA_Root	0.48763	.	.

```
SCOR=
   |   Intercept        AR_1        AR_2        AR_3        MA_1
-------------------------------------------------------------------
 1 |     -1e-009     -1e-009      9e-010     -5e-009     -7e-009
 2 |     -1e-009     -1e-009      9e-010     -5e-009     -7e-009
 3 |     -1e-009     -1e-009      9e-010     -5e-009     -7e-009
 4 |      1.4220     0.18060      1.2881      1.1585     -6e-009
 5 |     0.49355     0.82030    -0.30324     0.48986     0.13134
 6 |     -3.0377    -0.87932     -4.3205     0.48992    -0.10477
...................................................................
36 |      1.7194     0.45974    -0.43340      1.1310     0.10133
37 |      1.4259      1.5610     0.38128    -0.35944     0.47922
38 |     -1.3568    -0.67808     -1.4853    -0.36280    -0.18813
39 |      2.3998     0.54720      1.1993      2.6271    -0.85315
40 |     -3.8445     -5.0071    -0.87660     -1.9213     -2.6241
```

3. Use lags ar=1 and ma=1,2,3:

```
ar = 1; ma = [ 1:3 ];
optn = [ "lstech"     "lm" ,
         "mltech"     "tr" ,
         "print"         3 ];
< gof,parm,root,ase,cis,cov > = arma(xd2,.,ar,ma,optn);
print "GOF=",gof;
print "PARM=",parm;
print "Root=",root;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
GOF=
          |  g-of-fit
----------------------
    Error |    0.00000
   Y_Mean |    0.56729
   Y_Stdv |    0.33231
      MSE |    0.11043
     Time |    1.00000
   LogLik |     11.734
   N_Iter |     16.000
  MaxGrad |     7e-008
    SEReg |    0.34594
      AIC |    -2.0065
     SBIC |    -1.7954
   unused |       .
```

PARM=

	Estimates	Gradient
Intercept	0.32928	-2e-008
AR_1	0.40673	-6e-009
MA_1	-0.43325	1e-008
MA_2	-0.04240	-4e-008
MA_3	0.08842	7e-008

Root=

	1	2	3
AR_Root	0.40673	.	.
MA_Root	0.74631	0.76196	0.76196

ASE=

	A_St_Err.	T_Value	P_Value
Intercept	0.32817	1.0034	0.16128
AR_1	0.57655	0.70545	0.24260
MA_1	0.59611	-0.72679	0.76391
MA_2	0.24669	-0.17187	0.56774
MA_3	0.23102	0.38274	0.35211

CIs=

	Wald_LCI	Wald_UCI
Intercept	-0.31392	0.97247
AR_1	-0.72329	1.5367
MA_1	-1.6016	0.73510
MA_2	-0.52590	0.44110
MA_3	-0.36438	0.54122

COV=

SYM	Intercept	AR_1	MA_1
Intercept	0.10769		
AR_1	-0.18832	0.33241	
MA_1	0.18658	-0.32970	0.35535
MA_2	-0.00489	0.00882	-0.02982
MA_3	-0.01137	0.01971	-0.00963

```
        SYM |        MA_2        MA_3
------------------------------------
       MA_2 |     0.06086
       MA_3 |    -0.04140     0.05337
```

```
SCOR=
   |   Intercept        AR_1        MA_1        MA_2        MA_3
-------------------------------------------------------------------
 1 |      4e-010      1e-010     -3e-010      1e-009     -2e-009
 2 |      2.3948      1.9511     -1e-010      4e-010     -7e-010
 3 |     -6.5027     -5.7111     -1.1123     -1e-009      2e-009
 4 |      5.5508      2.3590     -1.1955     0.81815      2e-010
 5 |     0.63274     0.44886     0.07356    -0.13389     0.09163
 6 |     -6.8298     -4.3627    -0.35289    -0.80142      1.4588
...................................................................
37 |      2.7454      1.4474    -0.09572     -1.2659     -1.2981
38 |     -2.7003     -1.7532    -0.30430     0.09414      1.2451
39 |      6.4223      3.1440    -0.10074     0.72373    -0.22391
40 |     -8.4361     -6.5401     -2.0917     0.13233    -0.95065
```

5.3 Function `armafore`

gof = armafore(np,nh,est,y<,x<,ar<,ma<,optn> . >)

< gof,yhat > = armafore(np,nh,est,y<,x<,ar<,ma<,optn> . >)

Purpose: The `armafore` function implements the forecasting step of the AutoRegressive Moving-Average method.

Input: **np** must be a positive integer with $1 \leq np < N$ specifying the number of last observations of y should be forecast; that means, the first $N - np$ observations are used for regression

nh must be a positive integer with $1 \leq nh < N$ specifying the forecast "horizon";

est must be an p vector with the optimal ML parameters estimated from calling the `arma` function.

y must be an N vector of time series data.

x [optional] could be a $N \times n$ matrix of predictors

ar can be an integer scalar or vector of integers with the *nar* AR lag numbers, default is 1;

ma can be an integer scalar or vector of integers with the *nma* MA lag numbers, default is 1;

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

Option	Second Column	Meaning
`"center"`		centers the y data and as a consequnce does not use an intercept parameter
`"noint"`		does not use an intercept parameter
`"print"`	int	amount of printed outpout (def=0: no output)
`"nopr"`		do not print any output
`"sereg"`	real	should be the standard error output from the `gof` result of `arma`.

Output: gof is a vector of scalar results currently containing:

1. indicator for problems in the algorithm (=0: no problems)
2. mean value μ of input data y
3. standard deviation σ of input data y
4. variance σ^2
5. computation time in seconds
6. theoretical standard deviation of the h-step ahead forecast (assuming homoskedasticity)

yhat is a $N \times 3$ matrix containing in its columns:

1. the predicted $\hat{y}$ values;
2. the shifted input data $\bar{y}$;
3. the residuals $\hat{y} - \bar{y}$.

Restrictions:

1. The time series data y cannot have any missing values or string data. If x is specified as nonmissing, then the same is true for the predictors x.
2. There are methodological problems with ARMA forecasting of data including a regressor matrix $\mathbf{X}$.
3. The `ar` and `ma` specifications must agree with those used in the `arma` modeling step.
4. Some of the options should be compatible with those used in the `arma` modeling step, like "center" or "noint"!

Relationships: arhetero(), arma(), arima(), burg(), garch(), tsmeas()

Examples: Here we use the results from the three examples documented for the `arma` function:

1. Example 1 of `arma`: Use simple lags ar=1 and ma=1:

```
est = parm[,1]; sereg = gof[9];
np = 10; nh = 2;
optn = [ "print"       3 ,
         "sereg"   sereg ];
 < gof,yhat > = armafore(np,nh,est,xd2,.,ar,ma,optn);
print "GOF=",gof;
print "YHAT=",yhat;
```

```
          ********************************************
          ARMA Forecasting: N(AR)=1 N(MA)=1 (Uncentered)
          ********************************************
                    AR[1] =    1 MA[1] =    1

     Number of Observations . . . . . .        40
     Number of Predictor Variables. . . .       0
     Number of Parameter Estimates. . . .       3
     Mean . . . . . .                      0.5673
     Standard Deviation . . . . . .        0.3323
     Mean Squared Error (MSE) . . . . .    0.1104
     Standard Error Regression. . . . .    0.3514
```

GOF= Page 8

	g-of-fit
Error	0.00000
Y_Mean	0.56729
Y_Stdv	0.33231
MSE	0.11043
Time	0.00000
YSTD	0.35156
unused	.

YHAT=

	YHAT	YTPH	FERR
1	.	.	.
2	.	.	.
3	.	.	.
...			
27	.	.	.
28	.	.	.
29	.	.	.
30	0.56085	0.03180	-0.52905

```
31 |   0.55654   0.27690  -0.27964
32 |   0.56280   0.04620  -0.51660
33 |   0.56234   0.09710  -0.46524
34 |   0.56487   0.82350   0.25863
35 |   0.56525   0.69480   0.12955
36 |   0.55686   0.31710  -0.23976
37 |   0.55513   0.95020   0.39507
38 |   0.55890   0.03440  -0.52450
39 |   0.55292    .         .
40 |   0.56137    .         .
```

2. Example 2 of `arma`: Use lags ar=1,2,3 and ma=1:

```
est = parm[,1]; sereg = gof[9];
np = 10; nh = 2;
optn = [ "print"       3 ,
         "sereg"   sereg ];
 < gof,yhat > = armafore(np,nh,est,xd2,.,ar,ma,optn);
print "GOF=",gof;
print "YHAT=",yhat;
```

```
GOF=
         |  g-of-fit
---------------------
   Error |    0.00000
  Y_Mean |    0.56729
  Y_Stdv |    0.33231
     MSE |    0.11043
    Time |    0.00000
    YSTD |    0.35484
  unused |     .
  unused |     .
  unused |     .
  unused |     .
```

```
YHAT=
   |      YHAT      YTPH      FERR
---------------------------------
 1 |    .         .         .
 2 |    .         .         .
 3 |    .         .         .
.................................
27 |    .         .         .
28 |    .         .         .
```

```
29 |    .          .          .
30 |   0.61619   0.03180  -0.58439
31 |   0.45786   0.27690  -0.18096
32 |   0.65669   0.04620  -0.61049
33 |   0.43973   0.09710  -0.34263
34 |   0.59330   0.82350   0.23020
35 |   0.47969   0.69480   0.21511
36 |   0.50087   0.31710  -0.18377
37 |   0.61720   0.95020   0.33300
38 |   0.56632   0.03440  -0.53192
39 |   0.49271    .          .
40 |   0.68072    .          .
```

3. Example 3 of `arma`: Use lags ar=1 and ma=1,2,3:

```
est = parm[,1]; sereg = gof[9];
np = 10; nh = 2;
optn = [ "print"       3 ,
         "sereg"   sereg ];
 < gof,yhat > = armafore(np,nh,est,xd2,.,ar,ma,optn);
print "GOF=",gof;
print "YHAT=",yhat;
```

```
GOF=
         |  g-of-fit
---------------------
   Error |    0.00000
  Y_Mean |    0.56729
  Y_Stdv |    0.33231
     MSE |    0.11043
    Time |    0.00000
    YSTD |    0.34607
  unused |     .
  unused |     .
  unused |     .
  unused |     .
```

```
YHAT=
   |      YHAT      YTPH      FERR
-----------------------------------
 1 |    .          .          .
 2 |    .          .          .
 3 |    .          .          .
...................................
```

```
27 |     .          .          .
28 |     .          .          .
29 |     .          .          .
30 |   0.58534    0.03180   -0.55354
31 |   0.54813    0.27690   -0.27123
32 |   0.58139    0.04620   -0.53519
33 |   0.52150    0.09710   -0.42440
34 |   0.58340    0.82350    0.24010
35 |   0.53209    0.69480    0.16271
36 |   0.55172    0.31710   -0.23462
37 |   0.52947    0.95020    0.42073
38 |   0.49106    0.03440   -0.45666
39 |   0.54282     .          .
40 |   0.57429     .          .
```

5.4 Function `affrma`

gof = affrma(data,optn<,ref>)

< gof,dnew > = affrma(data,optn<,ref>)

Purpose: The `affrma` function implements the Bioconductor RMA algorithm by Bolstad et.al (2003) for the normalization of microarray data. This algorithm is an alternative to the `affvsn` and `affarms` methods for obtaining column normalized microarray data. However, other than fitting the parameters of a Maximum Likelihood model, the RMA method applies robust Medianpolish (Tukey, 1977a, p. 179), see also the `mpolish` function in CMAT to the data.

Input: **data** this should be a $N \times n$ matrix of microarray data where the rows correspond to features (genes) and the columns to samples which need to be normalized. The data may obtain a column for an ID variable specifying a strata. The column number of the ID variable must be specified by the `optn` argument.

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

ref not yet implemented.

Option	Second Column	Meaning
"bgcor"		perform background correction default is background correction
"idvar"	int	column number containing ID variable; default is no ID variable
"medp"	int	algorithm for mean or median polish default is =0: median polish, =1: mean polish
"mpit"	int	maximum number of median polish iterations; default is 10 for median and 1 for mean polish
"mpeps"	real	for terminating median polish; default is 1.e-2
"nobg"		do not perform background correction default is background correction
"nsamp"		size $n_r \leq N$ of a reduced data set, needed only when N is too large for the computer resources. default is $n_r = 0$
"pdat"	int	print input and fitted data sets
"print"	int	amount of printed output =0 specifies no printed output, default is 1

Output: **gof** a vector of scalar results, see below for content;

dnew this is the normalized $N \times n$ data set.

Restrictions: 1. The input data must be numeric and cannot contain any string or complex data.

2. Currently the input data must not have any missing values.

Relationships: affarms(), affvsn()

Examples: 1. :

5.5 Function `berkow`

prob = berkow(y<,optn<,"dist"<,...>>)

< prob,stat > = berkow(y<,optn<,"dist"<,...>>)

Purpose: Implements the Berkowitz test for a large variety of data distributions, including uniform and normal.

Input: **y** The first input object y must be either a n vector or a $n \times nc$ matrix. If y is a vector the scalar value of the probability of the CDF is returned, if y is a matrix then a vector of nc probabilities is returned by using each column of y separately.

optn this is a vector of options which should be initialized with missing values for defaults:

1. (int) amount of printed output (default is no output);
2. (real) probability for test (default is 0.05), must be in $[0, 1]$;
3. (int) version number of algorithm:
   0 for time series data (is default),
   1 for cross sectional data.

"dist" The following string options are valid:

("bern",x,prob) Bernoulli distribution

- `x` is the upper limit for cumulation $0 \leq x \leq 1$.
- `p` is the probability for success of 1, $0 \leq prob \leq 1$.

("beta",x,a,b<,<l>,<u>>) Beta distribution

- `x` is the upper limit for integration $0 \leq x \leq 1$.
- `a` is the first shape parameter $a \geq 0$.
- `b` is the second shape parameter $b \geq 0$.

("bin",s,prob,n) Binomial distribution

- `s` is the number of successes, $0 \leq s \leq n$.
- `prob` is the probability for success, $0 \leq prob \leq 1$.
- `n` is the number of independend trials, $n > 0$.

("cau",x<,< μ >,< σ >>) Cauchy distribution

- `x` is the upper limit for integration $x > 0$.
- μ location parameter $mu \geq 0$.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("chis",x,df<,nc>) (noncentral) ChiSquare distribution

- `x` is the upper limit for integration $x > 0$.
- `df` degrees of freedom $df > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("expo",x<,σ >) Exponential distribution

- `x` is the upper limit for integration $0 \leq x \leq 1$.
- σ scale parameter ($\sigma > 0$).

Default is $\sigma = 1$.

("f",x,ndf,ddf<,nc>) (noncentral) F distribution

- `x` is the upper limit for integration $x > 0$.
- `ndf` numerator degrees of freedom $ndf > 0$.
- `ddf` denominator degrees of freedom $ddf > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("gam",x,shape<,scale>)

- `x` is the upper limit for integration $x > 0$.
- `shape` shape parameter $shape > 0$.
- `scale` is the scale parameter $scale > 0$.

Default is $scale = 1$.

("gaus",x<,< μ >,< σ >>) Gauss (Normal) distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("geom",m,p)

- `m` is the upper limit for cumulation $m > 0$.
- `p` is the probability for success of 1, $0 \leq prob \leq 1$.

("hypg",x,m,k,n<,r>) Hypergeometric distribution

- `x` is the upper limit for cumulation $x > 0$.
- `m` is the population size ($m \geq 1$.
- `k` is the number of items ($0 \leq k \leq m$.
- `n` is the sample size ($0 \leq n \leq m$.
- `r` is an optional odds ratio ($r > 0$.

Default is $r = 1$.

("igau",x,d) Inverse Gauss (Wald) distribution

- `x` is the upper limit for integration $x > 0$.
- `d` shape parameter $d > 0$.

("lapl",x<,< μ >,< σ >>) Laplace distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("logi",x<,< μ >,< σ >>)

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("logn",x<, < μ >, < σ >>) LogNormal distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("negb",x,prob,n)

- `x` is the number of failures, $0 \leq f \leq n$.
- `prob` is the probability for success, $0 \leq prob \leq 1$.
- `n` is the number of successes, $n > 0$.

("norm",x<, < μ >, < σ >>) Normal (Gauss) distribution

- `x` is the upper limit for integration $x > 0$.
- μ shape parameter.
- σ scale parameter ($\sigma > 0$).

Default is $\mu = 0$ and $\sigma = 1$.

("pare",x,a<,k>) Pareto distribution

- `x` is the upper limit for integration $x > 0$.
- `a` shape parameter ($a > 0$.
- `k` scale parameter ($k > 0$).

Default is $k = 1$.

("pois",n,λ) Poisson distribution

- `x` is the upper limit for cumulation $x > 0$.
- λ shape parameter.

("t",t,df<,nc>) (noncentral) t distribution

- `t` is chosen from t distribution.
- `df` degrees of freedom $df > 0$.
- `nc` optional noncentrality $nc \geq 0$.

("unif",x<, <l>, <u>>)

- `x` is the upper limit for integration $x > 0$.
- `l` left location.
- `u` right location ($u > l$).

Default is $l = 0$ and $u = 1$.

("wald",x,d)

- `x` is the upper limit for integration $x > 0$.
- `d` shape parameter $d > 0$.

("weib",x,shape<,scale>) Weibull distribution

- `t` is the upper limit for integration $x > 0$.
- `shape` shape parameter $shape > 0$.
- `scale` scale parameter $scale > 0$.

Default is $scale = 1$.

... There may be a variable number of additional input arguments specifying non-default distributional parameters.

Output: **prob** The first result is either a scalar or an object of the test probabilities.

stat The `stat` result has the same size as `prob` and returns the test statistics (only for `optn[3]=0`).

Restrictions: 1. Missing value are returned if the input y contains missing data.

2.

Relationships: jarbera(), ksprob(), kstest(), shapwilk()

Examples: 1. Test $N = 1000$ random uniform generated data:

```
   options NOECHO;
#include "..\\tdata\\matlb.dat"
   options ECHO;


optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           1 ]; /* version=cs */
< p21,s21 > = berkow(uf1000,optn,"unif");
print "BERKOW P: UNIF",p21;
print "BERKOW S: UNIF",s21;


     Hypothesis H0: Data are uniformally distributed versus
       Hypothesis H1: Data are not uniformally distributed
      H0 accepted: Probability=0.0701486 Statistics=5.31428


optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           1 ]; /* version=cs */
< p22,s22 > = berkow(uf1000,optn,"norm");
print "BERKOW P: NORM",p22;
print "BERKOW S: NORM",s22;


       Hypothesis H0: Data are normally distributed versus
        Hypothesis H1: Data are not normally distributed
          H0 rejected: Probability=0 Statistics=1843.02
```

5.6 Function `garch`

gof = garch(meth,y,x,o,q,p<,optn<x0>>)

< gof,est,ase,cis,cov,sco > = garch(meth,y,x,o,q,p<,optn<x0>>)

Purpose: The `garch` function implements ARCH, GARCH, TARCH, AVARCH, ZARCH, APARCH, EGARCH, AGARCH, NAGARCH, IGARCH, and FIGARCH time series modeling.

The methods $2, \ldots, 8$ permit the specification of the error distribution using the `"error"` option:

"norm" Gaussian innovations (default); no additional parameter is estimated;

"stdt" Students T distributed error; one additional parameter $\nu > 2$ is estimated;

"gend" generalized error distribution; one additional parameter $\nu > 1$ is estimated;

"skew" skewed T error distribution; two additional parameters $\nu > 2$ and $-.99 < \lambda < .99$ are estimated.

meth=1: (G)ARCH implements the ARCH and GARCH: the `o` input argument specifies the AR lags; the predictor data `x` may be specified; the input `y` must not be centered; for $p = 0$ the ARCH model is estimated; for $o > 1$ the optimization is subject to nonlinear constraints; for $o = 1$ optimization algorithm LMEMQN is default, also NRRIDG or TRUREG are among preferred; for $o > 1$ optimization algorithm DQNSQP is default or COBYLA could be tried. There are

$$p = ni + nb + nar + nq + np$$

parameters to estimate, where the intercept $ni = 0 or 1$ is optional, $nb \geq 0$ is the number of columns of $\mathbf{X}$, and nar the number of entries of o, nq the number of entries of q, and np the number of entries of p.

meth=2: ARCH, GARCH, TARCH, AVGARCH, ZARCH : The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- the `o` input argument specifies the no asymmetric innovations;
- the `q` input argument specifies the nq symmetric innovations;
- the `p` input argument specifies the np lagged variances.

For $\delta = 2$ (default) and $\delta = 1$:

- for $\delta = 2$ squared value estimation ARCH, GARCH, GJR-GARCH;
- for $\delta = 1$ absolute value estimation TARCH, AVGARCH, ZARCH.

The conditional variance is given by:

$$\sigma_\delta^2 = \omega + \sum_{q=1}^{Q} \alpha_q |\epsilon_{t-q}|^\delta + \sum_{o=1}^{O} \gamma_o |\epsilon_{t-o}|^\delta I_{[\epsilon_{t-o}<0]} + \sum_{p=1}^{P} \beta_p \sigma_{t-p}^\delta$$

There are

$$p = 1 + no + nq + np + nm$$

parameters to estimate, where no the number of entries of o, nq the number of entries of q, np the number of entries of p, and nm is either zero, or one or two depending on estimating additional μ and λ parameters for nonnormal error distributions.

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$\alpha_q \geq 0, \quad q = 1, \ldots, Q$$

$$\beta_p \geq 0, \quad p = 1, \ldots, P$$

$$\alpha_j + \gamma_j \geq 0, \quad j = 1, \ldots, \max(Q, O)$$

$$\sum_q \alpha_q + .5 \sum_o \gamma_o + \sum_p \beta_p < 1$$

meth=3: APARCH implements the APARCH method: which is similar to $meth = 2$, but there are two versions:

- a fixed power of $\delta > .3$ can be specified; i.e. there are $p = 1 + no + nq + np + nm$ parameters to estimate;
- the best $\delta > .3$ is estimated as an additional parameter, i.e. there are $p = 2 + no + nq + np + nm$ parameters to estimate;

The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- the `o` input argument specifies the no asymmetric innovations;
- the `q` input argument specifies the nq symmetric innovations;
- the `p` input argument specifies the np lagged variances.

The conditional variance is given by:

$$\sigma_\delta^2 = \omega + \sum_{j=1}^{\max(Q,O)} \alpha_j (|\epsilon_{t-j}| + \gamma_j \epsilon_{t-j})^\delta + \sum_{p=1}^{P} \beta_p \sigma_{t-p}^\delta$$

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$\alpha_q \geq 0, \quad q = 1, \ldots, Q$$
$$\beta_p \geq 0, \quad p = 1, \ldots, P$$
$$-1 \leq \gamma_o \leq 1, \quad o = 1, \ldots, O$$
$$\delta \geq .3$$
$$\sum_q \alpha_q + \sum_p \beta_p < 1$$

meth=4: EGARCH implements the EGARCH: The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- the `o` input argument specifies the *no* asymmetric innovations;
- the `q` input argument specifies the *nq* symmetric innovations;
- the `p` input argument specifies the *np* lagged variances.

The conditional variance is given by:

$$ln(h_t) = \omega + \sum_{q=1}^{Q} \alpha_q(|\epsilon_{t-q}| - C) + \sum_{o=1}^{O} \gamma_o \epsilon_{t-o} + \sum_{p=1}^{P} \beta_p ln(h_{t-p})$$

where $e_t = r_t/\sqrt{h_t}$ and $C = \sqrt{2}/\sqrt{\pi}$.
There are

$$p = 1 + no + nq + np + nm$$

parameters to estimate, where *no* the number of entries of *o* etc.

For $p > 0$ the optimization is subject to p nonlinear constraints on the absolute size of the roots of the characteristic polynomial of the β parameters. The optimization algorithm DQNSQP is default or COBYLA could be tried; there may be convergence problems if the model is not fitting the data well enough.

meth=5: AGARCH implements the AGARCH (Engle, 1990): The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing. The input argument `o` must be specified as missing.

- the `q` input argument specifies the *nq* symmetric innovations;
- the `p` input argument specifies the *np* lagged variances.

An additional model parameter γ is estimated.

The volatility dynamics is given by:

$$h_t = \omega + \sum_{q=1}^{Q} (r_{t-q} - \gamma)^2 + \sum_{p=1}^{P} h_{t-p}$$

There are

$$p = 2 + nq + np + nm$$

parameters to estimate, where nq the number of entries of q etc. The input `y` must be centered around the mean.

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$\alpha_q \geq 0, \quad q = 1, \ldots, Q$$

$$\beta_p \geq 0, \quad p = 1, \ldots, P$$

$$q(.01, y) \leq \gamma \leq q(.99, y)$$

$$\sum_q \alpha_q + \sum_p \beta_p < 1$$

meth=6: NAGARCH implements the NAGARCH (Engle & Ng, 1993): The NAGARCH model is very similar to AGARCH. The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- The input argument `o` must be specified as missing;
- the `q` input argument specifies the nq symmetric innovations;
- the `p` input argument specifies the np lagged variances.

An additional model parameter γ is estimated.

The volatility dynamics is given by:

$$h_t = \omega + \sum_{q=1}^{Q} (r_{t-q} - \gamma\sqrt{h_{t-q}})^2 + \sum_{q=1}^{P} h_{t-p}$$

There are

$$p = 2 + nq + np + nm$$

parameters to estimate, where nq the number of entries of q etc.

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$\alpha_q \geq 0, \quad q = 1, \ldots, Q$$

$$\beta_p \geq 0, \quad p = 1, \ldots, P$$

$$\sum_q \alpha_q + \sum_p \beta_p + (1 + \gamma^2) < 1$$

meth=7: IGARCH implements the IGARCH and IAVARCH methods: The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- The input argument `o` must be specified as missing;
- the `q` input argument specifies the nq symmetric innovations;
- the `p` input argument specifies the np lagged variances.

An intercept ω is estimated optional. It is default and is suppressed by NOINT option. A $\delta = 1$ or $\delta = 2$ (default) can be specified using the options argument.

The conditional variance is given by:

$$g(h_t) = \omega + \sum_{q=1}^{Q} \alpha_q |r_{t-q}|^\delta + \sum_{q=1}^{P} \beta_p g(h_{t-p})$$

where

$$g(x) = \begin{cases} \sqrt{x} & \text{for } \delta = 1 \\ x & \text{for } \delta = 2 \end{cases}$$

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$\alpha_q \geq 0, \quad q = 1, \ldots, Q$$

$$\beta_p \geq 0, \quad p = 1, \ldots, P$$

$$\sum_q \alpha_q + \sum_p \beta_p = 1$$

meth=8: FIGARCH implements the FIGARCH method: The input `y` must be centered around the mean. Currently the argument `x` must be specified as missing.

- The input argument `o` must be specified as missing;
- the `q` input argument specifies the order of the short memory autoregressive process, must be 0 or 1;
- the `p` input argument specifies the order of the moving average process, must be 0 or 1.

Additionally the integer truncation lag must be specified with the options argument.

The conditional variance is given by:

$$h_t = \omega + \sum_{i=1}^{\infty} \lambda_i \epsilon_{t-i}^2$$

where

$$\begin{aligned} \delta_1 &= d \\ \lambda_1 &= \Phi - \beta + d \\ \delta_i &= \frac{i - 1 - d}{i} \delta_{i-1}, \quad i = 2, \ldots \\ \lambda_i &= \beta \lambda_{i-1} + \delta_i - \Phi \delta_{i-1}, \quad i = 2, \ldots \end{aligned}$$

The following constraints are resolved by model reparametrization so that the optimization is unconstrained:

$$\omega \geq 0$$

$$0 \leq d \leq 1$$

$$0 \leq \Phi \leq (1-d)/2$$

$$0 \leq \beta \leq d + \Phi$$

Input: **meth** should be an integer scalar $meth \in \{1, \ldots, 8\}$ (see above for method assignment).

y should be an N vector $\mathbf{Y}$ of time series data.

x could be either a missing value or an $N \times nb$ matrix $\mathbf{X}$ of predictor data.

o should be either an integer scalar or vector of lag numbers depending on the estimation method.

q should be either an integer scalar or vector determining the lag numbers of the ARCH parameter.

p should be either an integer scalar or vector determining the lag numbers of the GARCH parameter.

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

x0 a p vector specifying initial estimates for the optimization.

In addition to many options for the optimization algorithms the following can be specified:

Option	Second Column	Meaning
"alpha"	real	probability for confidence intervals, default=.05
"center"		centers the y data
"delta"	real	power for methods=2,3,7, default is delta=2
"error"	string	error distribution
	"norm"	Gaussian innovations (default);
	"stdt"	T distributed error;
	"gend"	generalized error distribution;
	"skew"	skewed T error distribution;
"noint"		does not use an intercept parameter
		valid only for methods 1 and
"norob"		suppress robust (sandwich) covariance matrix
"nosco"		suppress computation of scores
"nopr"		do not print any output
"pcov"		print asymptotic covariance matrices
"phis"		print optimization history
"pscor"		print (large $N \times p$) score matrix
"print"	int	amount of printed outpout (def=0: no output)
"truncl"	int	size of truncation lag (default=1000)
		valid only for meth=8
"tech"	string	optimization technique
		see function `nlp()`

Output: **gof** a vector of scalar results, see below for content;

est a $p \times 2$ matrix of estimates (in first column) and the gradient vector in its second column;

ase a $p \times 3$ matrix with asymptotic standard error (based on the diagonal of the robust covariance matrix) in its first column, the t value in its second, and the p value in its third column;

cis a $p \times 2$ marix of (robust) lower and upper Wald confidence intervals of the estimates;

cov a $p \times p$ (robust, sandwich, White) asymptotic covariance matrix;

sco is an $N \times p$ matrix of scores.

The current content of the `gof` vector:

1. indicator for problems in the algorithm (=0: no problems)
2. mean value μ of input data y
3. standard deviation σ of input data y
4. variance σ^2
5. computation time in seconds
6. Loglikelihood
7. Akaikes information criterion AIC

8. Bayesian (Schwartz) information criterion BIC
9. number of iterations for optimization
10. $L1$ norm of gradient at the optimal solution

Restrictions: 1. The input data y must be numeric and cannot contain any string or complex data.

2. Currently the input data y must not have any missing values.

Relationships: arma(), arima(), tsmeas()

Examples: 1. Data:

The following time series data has $N = 300$ observations:

```
print "Read Data TWO by Minbo: from nlp213";
options NOECHO;
%inc "c:\\cmat\\tdata\\minbo0.dat";
options ECHO;
```

2. Method 1:

```
m = q = p = 1;
print "Method 1: Unconstrained (GARAR)";
optn = [ "tech"     "lq" ,
         "cent"          ,
         "print"       3 ];
< gof,est,ase,cis,cov,scor > = garch(1,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
          ****************************************
          Model 1: (G)ARCH Modeling: AR=1 NQ=1 NP=1
          ****************************************
            AR[1] =    1 Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .         4
  Mean Y . . . . . .                      0.0016
  Standard Deviation Y . . . . . .        1.1442
  Mean Squared Error (MSE) . . . . . .    1.3093
  Number of Back Casts . . . . . .            1
  Optimization Technique . . . . . .      LMEMQN
```

```
                         ******************
                         Optimization Start
                         ******************

                        Parameter Estimates
                        -------------------

Parameter     Estimate   Gradient   Lower BC   Upper BC

 1 AR_1     0.00000000 -24.019592 -0.9999000  0.9999000
 2 ARCH_0   1.30926796  0.0016684  1.00e-008     .
 3 ARCH_1   1.000e-006 -29.834384  1.00e-008     .
 4 GARCH_1  1.000e-006  0.0015156  1.00e-008     .

           Value of Objective Function =      466.102


              Limited Memory Quasi Newton Method
            Gradient Computed by Finite Differences

  Iteration Start:
  N. Variables              4
  N. Bound. Constr.         5        N. Mask  Constr.          0
  Criterion       466.1017562        Max Grad Entry  29.83438353
  N. Active Constraints     0        N. Grad Storage           5


 Iter nfun act  optcrit  difcrit maxgrad   alpha    slope   gcrit
    1    3   2 465.0617 1.040046 18.4056  7e-003 -914.108 1.30926
    2    4   1 462.8073 2.254445 14.3460 1.00000 -2.78328 1.25336
    3    5   1 460.9802 1.827046 9.98118 1.00000 -3.48729 20.0384
    4    7   0 454.6460 6.334191 15.0617 5.00000 -1.30142 1.43248
    5   10   0 454.1369 0.509175 11.4576 0.15347 -5.91310 1.48010
    6   11   0 453.2002 0.936687 15.2952 1.00000 -1.84432 15.2952
    7   12   0 452.7153 0.484894 7.43397 1.00000 -0.68535 7.43397
    8   13   0 452.4463 0.268983 6.31992 1.00000 -0.51750 0.85465
    9   14   0 452.1939 0.252427 2.23234 1.00000 -0.33597 1.82205
   10   15   0 452.1174 0.076441 1.79658 1.00000 -0.21904 1.79658
   11   16   0 452.1039 0.013526 0.29609 1.00000 -0.03298 0.29609
   12   17   0 452.1032 6.6e-004 0.28550 1.00000  -2e-003 0.28550
   13   19   0 452.1031 9.1e-005 0.06968 0.19375  -9e-004 0.06968
   14   20   0 452.1031 2.1e-005  5e-003 1.00000  -4e-005  5e-003
   15   21   0 452.1031 7.9e-008  3e-004 1.00000  -2e-007  3e-004
   16   22   0 452.1031 5.5e-010  3e-004 1.00000  -7e-010  3e-004
   17   23   0 452.1031 3.6e-010  9e-005 1.00000  -1e-009  9e-005
   18   24   0 452.1031 6.3e-011  8e-006 1.00000  -1e-010  8e-006
```

```
Successful Termination After    18 Iterations
ABSGCONV convergence criterion satisfied.
Criterion       452.1031258        Max Grad Entry  8.0944e-006
N. Active Constraints     0        N. Grad Storage           5
Skipped Updates           0        Segments Explored        21
N. Function Calls        25        N. Gradient Calls        25
N. Line Searches         18        Preproces. Time           0
Time for Method           0        Effective Time            0
```

****
Optimization Results
****

Parameter Estimates

Parameter	Estimate	Gradient	Active BC
1 AR_1	0.26060427	8.09e-006	
2 ARCH_0	0.42786973	-6.27e-006	
3 ARCH_1	0.30722878	-7.83e-007	
4 GARCH_1	0.37519470	-4.16e-007	

Value of Objective Function = 452.103

Log Likelihood= -452.103
SBC=-881.391 AIC=-896.206

Covariance Matrix 2: H = (NOBS/d) inv(G)
****

Dense Symmetric Matrix (4 by 4)

S	AR_1	ARCH_0	ARCH_1	GARCH_1
AR_1	0.0119455			
ARCH_0	0.0010376	0.0125162		
ARCH_1	-6.45e-004	0.0010024	0.0097181	
GARCH_1	-3.92e-004	-0.0095789	-0.0066432	0.0128294

Factor sigma=1 (nobs=300 df=4 vard=300)
Determinant = 2.72817e-009
Matrix has Only Positive Eigenvalues

Robust Asymptotic Covariance Matrix of Estimates

Dense Symmetric Matrix (4 by 4)

S	AR_1	ARCH_0	ARCH_1	GARCH_1
AR_1	0.0363805			
ARCH_0	-0.0023527	0.0077765		
ARCH_1	-0.0060095	-6.61e-004	0.0094990	
GARCH_1	0.0043184	-0.0043883	-0.0045065	0.0064840

Robust Asymptotic Correlation Matrix of Estimates

Dense Symmetric Matrix (4 by 4)

S	AR_1	ARCH_0	ARCH_1	GARCH_1
AR_1	1.0000000			
ARCH_0	-0.1398734	1.0000000		
ARCH_1	-0.3232716	-0.0769177	1.0000000	
GARCH_1	0.2811663	-0.6179963	-0.5742187	1.0000000

Parameter Estimates and Asympt. Standard Errors

Parameter	Estimate	AStdErr	T Value	P Value
AR_1	0.2606043	0.1907367	1.3663040	0.086440
ARCH_0	0.4278697	0.0881842	4.8520000	0.000001
ARCH_1	0.3072288	0.0974628	3.1522683	0.000893
GARCH_1	0.3751947	0.0805233	4.6594522	0.000002

Wald Confidence Limits (Alpha=0.05)

Parameter	Estimate	Lower	Upper
AR_1	0.2606043	-0.1132327	0.6344413
ARCH_0	0.4278697	0.2550319	0.6007076
ARCH_1	0.3072288	0.1162053	0.4982523
GARCH_1	0.3751947	0.2173718	0.5330176

Factor sigma=1 (nobs=300 df=4 vard=300)

3. Method 2: GARCH: delta=2:

```
m = q = p = 1;
print "Method 2: Reparametrized Constraints: delta=2";
optn = [ "cent"            ,
         "delta"        2. ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(2,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
          ****************************************
          Model 2: (G)ARCH Modeling: NO=1 NQ=1 NP=1
          ****************************************
              Modeling Squared Values (Delta=2)
             O[1] =    1 Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .          300
  Number of Parameter Estimates. . . .          4
  Mean Y . . . . . .                       0.0016
  Standard Deviation Y . . . . . .         1.1442
  Mean Squared Error (MSE) . . . . . .     1.3093
  Number of Back Casts . . . . . .             1
  Error Distribution . . . . . .          Normal
  Optimization Technique . . . . . .      TRUREG

                   *****************
                   Optimization Start
                   *****************

                  Parameter Estimates
                  -------------------

              Parameter      Estimate   Gradient

               1 Intercept-1.33996974 -7.4183275
               2 Alpha_1  -1.38629190  0.3059796
               3 Gamma_1  -1.25276015  0.1473073
```

```
      4 Beta_1    0.91630058 -1.9482010

 Value of Objective Function =      455.645


                Hessian Matrix
                **************

      Symmetric Matrix: Dense Storage

 S |           1          2          3          4
 ---------------------------------------------------
 1 |   47.491963  1.7283319 -4.3302148  26.269433
 2 |   1.7283319  3.5187833 -2.9335752  1.6697438
 3 |  -4.3302148 -2.9335752  7.3798025 -1.0955059
 4 |   26.269433  1.6697438 -1.0955059  16.182565


          Trust Region Optimization
          Without Parameter Scaling
     Gradient Computed by Finite Differences
 Hessian Computed by Finite Differences (dense)
          (Using Only Function Calls)


 Iteration Start:
 N. Variables             4
 Criterion       455.6448096        Max Grad Entry  7.418327504
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1    0    2   0  454.4628 1.181992 4.81094 1.10038  1.00000
   2    0    3   0  454.2628 0.200034 0.20377 0.00000  1.04053
   3    0    4   0  454.2590 3.8e-003  7e-003 0.00000  0.18087
   4    0    5   0  454.2590 1.3e-006  4e-006 0.00000  0.04976

 Successful Termination After     4 Iterations
 GCONV convergence criterion satisfied.
 Criterion       454.2590151        Max Grad Entry  4.0547e-006
 Ridge (lambda)  0.000000000        TR Radius       0.049761569
 Act.dF/Pred.dF  0.999835589
 N. Function Calls         6        N. Gradient Calls         3
 N. Hessian Calls          6        Preproces. Time           0
 Time for Method           0        Effective Time            0


                ********************
```

```
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

          Parameter      Estimate   Gradient

           1 Intercept-0.82432306 -4.05e-006
           2 Alpha_1  -1.35239391  1.92e-007
           3 Gamma_1  -1.08135794  4.42e-007
           4 Beta_1    0.14335696 -3.94e-006

       Value of Objective Function =      454.259


                     Hessian Matrix
                     **************

           Symmetric Matrix: Dense Storage

       S |           1          2          3          4
       ---------------------------------------------------
       1 |   62.341578  3.1051598 -1.4674951  25.986019
       2 |   3.1051598  2.7639938 -1.8008995  1.4816808
       3 |  -1.4674951 -1.8008995  5.2601208 -0.0024428
       4 |   25.986019  1.4816808 -0.0024428  13.691414


              Log Likelihood= -454.259
              SBC=-885.703 AIC=-900.518


         Covariance Matrix 2: H = (NOBS/d) inv(G)
         ****************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept    Alpha_1    Gamma_1     Beta_1
         ---------------------------------------------------
Intercept |   0.0151475
  Alpha_1 |   5.65e-004  0.0133848
  Gamma_1 |   0.0029399 -0.0071469  0.0263580
   Beta_1 |  -0.0120534 -0.0071370 -0.0045574  0.0165277

         Factor sigma=1 (nobs=300 df=4 vard=300)
               Determinant = 8.94864e-009
```

Matrix has Only Positive Eigenvalues

Robust Asymptotic Covariance Matrix of Estimates

Dense Symmetric Matrix (4 by 4)

S	Intercept	Alpha_1	Gamma_1	Beta_1
Intercept	0.0088788			
Alpha_1	2.27e-005	0.0124115		
Gamma_1	-7.70e-005	-0.0048557	0.0213510	
Beta_1	-0.0058128	-0.0068907	-0.0017048	0.0101131

Robust Asymptotic Correlation Matrix of Estimates

Dense Symmetric Matrix (4 by 4)

S	Intercept	Alpha_1	Gamma_1	Beta_1
Intercept	1.0000000			
Alpha_1	0.0021639	1.0000000		
Gamma_1	-0.0055927	-0.2982873	1.0000000	
Beta_1	-0.6134338	-0.6150440	-0.1160142	1.0000000

Parameter Estimates and Asympt. Standard Errors

Parameter	Estimate	AStdErr	T Value	P Value
Intercept	0.4385318	0.0942275	4.6539678	0.000002
Alpha_1	0.2054382	0.1114069	1.8440345	0.033089
Gamma_1	0.2489317	0.1461199	1.7036131	0.044751
Beta_1	0.3589155	0.1005640	3.5690243	0.000209

Wald Confidence Limits (Alpha=0.05)

Parameter	Estimate	Lower	Upper
Intercept	0.4385318	0.2538492	0.6232143
Alpha_1	0.2054382	-0.0129153	0.4237917

```
 Gamma_1  0.2489317 -0.0374580  0.5353214
  Beta_1  0.3589155  0.1618136  0.5560174
Factor sigma=1 (nobs=300 df=4 vard=300)
```

4. Method 2: GARCH: delta=1:

```
m = q = p = 1;
print "Method 2: Reparametrized Constraints: delta=1";
optn = [ "cent"            ,
         "delta"        1. ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(2,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
          ****************************************
          Model 2: (G)ARCH Modeling: NO=1 NQ=1 NP=1
          ****************************************
             Modeling Absolute Values (Delta=1)
             O[1] =    1 Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .     0.0016
  Standard Deviation Y . . . . . .     1.1442
  Mean Squared Error (MSE) . . . . . .     1.3093
  Number of Back Casts . . . . . .         1
  Error Distribution . . . . . .    Normal
  Optimization Technique . . . . . .    TRUREG


                   ******************
                   Optimization Start
                   ******************

                  Parameter Estimates
                  -------------------

             Parameter      Estimate   Gradient
```

```
          1 Intercept-1.53586564 -58.124474
          2 Alpha_1  -1.38629190  3.6784135
          3 Gamma_1  -2.02437929 -6.7925300
          4 Beta_1    1.09025389 -34.811202

       Value of Objective Function =      464.281


                    Hessian Matrix
                    **************

          Symmetric Matrix: Dense Storage

       S |           1          2          3          4
       ---------------------------------------------------
       1 |   244.19334  4.7451953  20.045738  170.53481
       2 |   4.7451953  12.384253 -6.9473193  7.9429617
       3 |   20.045738 -6.9473193  15.343707  17.500619
       4 |   170.53481  7.9429617  17.500619  115.50154


             Trust Region Optimization
             Without Parameter Scaling
        Gradient Computed by Finite Differences
     Hessian Computed by Finite Differences (dense)
             (Using Only Function Calls)


 Iteration Start:
 N. Variables             4
 Criterion       464.2813571        Max Grad Entry  58.12447438
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
  1*   0    2   0  456.3216 7.959783 20.3783 7.64481  1.00000
  2    0    3   0  455.2247 1.096896 1.54743 0.00000  1.04789
  3    0    4   0  455.1950 0.029695 0.14818 0.00000  0.30112
  4    0    5   0  455.1947 2.8e-004  3e-003 0.00000  0.12380
  5    0    6   0  455.1947 6.3e-008  1e-006 0.00000  0.01537

 Successful Termination After     5 Iterations
 GCONV convergence criterion satisfied.
 Criterion       455.1947014        Max Grad Entry  1.2954e-006
 Ridge (lambda)  0.000000000        TR Radius       0.015373816
 Act.dF/Pred.dF  1.000341620
 N. Function Calls         7        N. Gradient Calls         3
```

```
N. Hessian Calls          7        Preproces. Time          0
Time for Method           0        Effective Time           0
```

```
                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

              Parameter      Estimate   Gradient

               1 Intercept-0.84061128  1.30e-006
               2 Alpha_1  -1.48322181  1.51e-008
               3 Gamma_1  -1.57885226  2.15e-007
               4 Beta_1    0.22287797  3.49e-007

         Value of Objective Function =      455.195

                      Hessian Matrix
                      **************

            Symmetric Matrix: Dense Storage

         S |          1          2          3          4
         ---------------------------------------------------
         1 |   287.78217  9.7874108  6.9416767  133.94190
         2 |   9.7874108  5.3759445 -3.8963319  5.0234865
         3 |   6.9416767 -3.8963319  11.843638  4.2857946
         4 |   133.94190  5.0234865  4.2857946  65.293889

                  Log Likelihood= -455.195
                  SBC=-887.574 AIC=-902.389

           Covariance Matrix 2: H = (NOBS/d) inv(G)
           ****************************************

               Dense Symmetric Matrix (4 by 4)

           S |   Intercept    Alpha_1    Gamma_1     Beta_1
           ---------------------------------------------------
   Intercept |   0.0151876
     Alpha_1 |   0.0018433  0.0068219
```

```
  Gamma_1 |   4.43e-004 -0.0035559  0.0077095
   Beta_1 |  -0.0148603 -0.0052871 -3.88e-004  0.0175127

        Factor sigma=1 (nobs=300 df=4 vard=300)
              Determinant = 3.03027e-010
         Matrix has Only Positive Eigenvalues


    Robust Asymptotic Covariance Matrix of Estimates
    ************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1
        ---------------------------------------------
Intercept |   0.0097406
  Alpha_1 |   4.07e-004  0.0062409
  Gamma_1 |  -0.0010235 -0.0027778  0.0062047
   Beta_1 |  -0.0084654 -0.0039304  0.0010388  0.0102681


   Robust Asymptotic Correlation Matrix of Estimates
   *************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1
        ---------------------------------------------
Intercept |   1.0000000
  Alpha_1 |   0.0522012  1.0000000
  Gamma_1 |  -0.1316572 -0.4463881  1.0000000
   Beta_1 |  -0.8464653 -0.4909820  0.1301413  1.0000000


    ********************************************
    Parameter Estimates and Asympt. Standard Errors
    ********************************************

  Parameter   Estimate    AStdErr    T Value   P Value
  Intercept  0.4314467  0.0986943  4.3715476  0.000009
    Alpha_1  0.1849043  0.0789995  2.3405741  0.009959
    Gamma_1  0.1253326  0.0787700  1.5911220  0.056325
     Beta_1  0.4178559  0.1013315  4.1236515  0.000024


          ***********************************
          Wald Confidence Limits (Alpha=0.05)
```

```
**********************************

Parameter   Estimate      Lower      Upper
Intercept  0.4314467  0.2380095  0.6248839
  Alpha_1  0.1849043  0.0300680  0.3397405
  Gamma_1  0.1253326 -0.0290537  0.2797189
   Beta_1  0.4178559  0.2192498  0.6164620
 Factor sigma=1 (nobs=300 df=4 vard=300)
```

5. Method 3: APARCH: specify $\delta = 1.5$:

```
m = q = p = 1;
print "Method 3: APARCH: Reparametrized Constraints: Specify Delta";
optn = [ "cent"            ,
         "delta"       1.5 ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(3,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
**************************************
Model 3: APARCH Modeling: NO=1 NQ=1 NP=1
**************************************
      Modeling Constant Delta=1.5
  O[1] =    1 Q[1] =    1 P[1] =    1

Number of Observations . . . . . .      300
Number of Parameter Estimates. . . .      4
Mean Y . . . . . .     0.0016
Standard Deviation Y . . . . . .     1.1442
Mean Squared Error (MSE) . . . . . .     1.3093
Number of Back Casts . . . . . .        1
Error Distribution . . . . . .    Normal
Optimization Technique . . . . . .    TRUREG

                  ******************
                  Optimization Start
                  ******************
```

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	-0.80807956	-67.266644
2	Alpha_1	-1.04456937	-24.750763
3	Gamma_1	0.00000000	1.9300417
4	Beta_1	0.14454518	-134.16255

Value of Objective Function = 484.494

Hessian Matrix

Symmetric Matrix: Dense Storage

S	1	2	3	4
1	104.67489	52.890431	-3.3000239	241.90207
2	52.890431	9.0992573	-0.2050936	61.237870
3	-3.3000239	-0.2050936	0.3314837	-4.9151948
4	241.90207	61.237870	-4.9151948	376.88691

Trust Region Optimization
Without Parameter Scaling
Gradient Computed by Finite Differences
Hessian Computed by Finite Differences (dense)
(Using Only Function Calls)

Iteration Start:

N. Variables	4		
Criterion	484.4939750	Max Grad Entry	134.1625549
TR Radius	1.000000000		

Iter	rest	nfun	act	optcrit	difcrit	maxgrad	lambda	radius
1*	0	3	0	461.7268	22.76718	25.9680	66.6286	0.45258
2	0	5	0	458.1502	3.576632	9.88555	1.44608	1.42239
3	0	6	0	457.5656	0.584535	0.71054	0.08086	1.46799
4	0	7	0	457.4567	0.108951	0.16465	0.00000	1.51947
5	0	8	0	457.4223	0.034412	0.04680	0.00000	1.09544
6	0	9	0	457.4101	0.012209	0.01514	0.00000	1.01036
7	0	10	0	457.4055	4.5e-003	5e-003	0.00000	0.95878

```
 8    0   11   0  457.4036 1.9e-003  3e-003 0.00000  0.90175
 9    0   12   0  457.4029 7.6e-004  1e-003 0.00000  0.95438
10    0   13   0  457.4026 2.4e-004  2e-004 0.00000  1.02312
11    0   14   0  457.4026 5.2e-005  1e-004 0.00000  0.79537
12*   0   15   0  457.4025 9.5e-005  3e-004  2e-004  1.10559
13    0   16   0  457.4025 1.6e-005  2e-004 0.00000  1.11262
14    0   17   0  457.4025 3.8e-006  6e-005 0.00000  1.14367

Successful Termination After    14 Iterations
GCONV convergence criterion satisfied.
Criterion       457.4024572       Max Grad Entry  5.9941e-005
Ridge (lambda)  0.000000000       TR Radius       1.143670882
Act.dF/Pred.dF  1.871066805
N. Function Calls        18       N. Gradient Calls         7
N. Hessian Calls         16       Preproces. Time           0
Time for Method           0       Effective Time            0

                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

          Parameter      Estimate   Gradient

           1 Intercept-1.13572475 -5.99e-005
           2 Alpha_1   0.45005298 -3.25e-005
           3 Gamma_1  -10.9289805  2.68e-005
           4 Beta_1    0.34625681 -1.76e-005

        Value of Objective Function =      457.402

                     Hessian Matrix
                     **************

           Symmetric Matrix: Dense Storage

        S |           1          2          3          4
        ---------------------------------------------------
        1 |   31.066380  12.768076 -1.83e-004  110.73000
        2 |   12.768076  8.3003477  2.24e-005  53.621172
        3 |  -1.83e-004  2.24e-005 -7.53e-005 -2.41e-004
        4 |   110.73000  53.621172 -2.41e-004  481.50746
```

```
                Log Likelihood= -457.402
                SBC=-891.99 AIC=-906.805


         Covariance Matrix 2: H = (NOBS/d) inv(G)
         ****************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept    Alpha_1    Gamma_1     Beta_1
         ---------------------------------------------------
Intercept |   0.0195339
  Alpha_1 |  -0.0019982  0.0249742
  Gamma_1 |   0.0060737 -0.0028987  0.0357288
   Beta_1 |  -0.0127947 -0.0100935 -0.0013812  0.0157775


         Factor sigma=1 (nobs=300 df=4 vard=300)
               Determinant = 3.37945e-008
          Matrix has Only Positive Eigenvalues


     Robust Asymptotic Covariance Matrix of Estimates
     ************************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept    Alpha_1    Gamma_1     Beta_1
         ---------------------------------------------------
Intercept |   0.0175506
  Alpha_1 |  -0.0120204  0.0257178
  Gamma_1 |   0.0145278 -0.0151766  0.0377656
   Beta_1 |  -0.0055463 -0.0031582 -0.0029162  0.0061195


    Robust Asymptotic Correlation Matrix of Estimates
    *************************************************

             Dense Symmetric Matrix (4 by 4)

         S |   Intercept    Alpha_1    Gamma_1     Beta_1
         ---------------------------------------------------
Intercept |   1.0000000
  Alpha_1 |  -0.5657894  1.0000000
  Gamma_1 |   0.5642936 -0.4869768  1.0000000
   Beta_1 |  -0.5351819 -0.2517489 -0.1918264  1.0000000
```

```
*********************************************
Parameter Estimates and Asympt. Standard Errors
*********************************************

Parameter   Estimate    AStdErr    T Value   P Value
Intercept  0.3211893  0.1324788  2.4244575  0.007965
  Alpha_1  0.6105297  0.1603676  3.8070647  0.000085
  Gamma_1  0.1047319  0.1943338  0.5389280  0.295171
   Beta_1  0.3462568  0.0782271  4.4263051  0.000007


      ***********************************
      Wald Confidence Limits (Alpha=0.05)
      ***********************************

    Parameter   Estimate      Lower      Upper
    Intercept  0.3211893  0.0615356  0.5808430
      Alpha_1  0.6105297  0.2962150  0.9248444
      Gamma_1  0.1047319 -0.2761553  0.4856192
       Beta_1  0.3462568  0.1929346  0.4995790
     Factor sigma=1 (nobs=300 df=4 vard=300)
```

6. Method 3: APARCH: estimate δ:

```
m = q = p = 1;
print "Method 3: APARCH: Reparametrized Constraints: Estimate Delta";
optn = [ "cent"            ,
         "delta"         . ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(3,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
*************************************
Model 3: APARCH Modeling: NO=1 NQ=1 NP=1
*************************************
       Modeling Variable Delta=2
  O[1] =    1 Q[1] =    1 P[1] =    1
```

Number of Observations	300
Number of Parameter Estimates	5
Mean Y	0.0016
Standard Deviation Y	1.1442
Mean Squared Error (MSE)	1.3093
Number of Back Casts	1
Error Distribution	Normal
Optimization Technique	TRUREG

****
Optimization Start
****

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	-0.80807956	-67.266644
2	Alpha_1	-1.04456937	-24.750763
3	Gamma_1	0.00000000	1.9300417
4	Beta_1	0.14454518	-134.16255
5	Delta	-1.45528723	-16.203738

Value of Objective Function = 484.494

Hessian Matrix
****

Symmetric Matrix: Dense Storage

S	1	2	3	4	5
1	104.67489	52.890431	-3.3000239	241.90207	68.519298
2	52.890431	9.0992573	-0.2050936	61.237870	21.760473
3	-3.3000239	-0.2050936	0.3314837	-4.9151948	-0.7538566
4	241.90207	61.237870	-4.9151948	376.88691	118.88276
5	68.519298	21.760473	-0.7538566	118.88276	12.505634

Trust Region Optimization
Without Parameter Scaling
Gradient Computed by Finite Differences
Hessian Computed by Finite Differences (dense)
(Using Only Function Calls)

```
 Iteration Start:
 N. Variables              5
 Criterion       484.4939750        Max Grad Entry  134.1625549
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1*   0    3   0  462.6536 21.84039 35.7901 196.014  0.22429
   2*   0    4   0  460.9077 1.745880 11.7648 17.8182  0.24322
   3*   0    7   0  456.9692 3.938522 3.02933 0.84716  2.24672
   4*   0    8   0  454.7810 2.188210 0.42557 0.44642  2.25214
   5    0    9   0  454.4958 0.285173 0.53019 0.00000  2.32350
   6    0   10   0  454.4208 0.074962 0.15293 0.00000  1.42516
   7    0   11   0  454.3954 0.025435 0.06860 0.00000  1.29484
   8    0   12   0  454.3861 9.3e-003 0.02121 0.00000  1.01182
   9*   0   15   0  454.3843 1.9e-003  4e-003 -4e-003  2.14396
  10*   0   17   0  454.3830 1.3e-003  2e-003  6e-003  1.69460
  11*   0   18   0  454.3824 5.5e-004  2e-003 -2e-003  1.70251
  12*   0   20   0  454.3819 5.4e-004  3e-003  3e-003  1.79430
  13    0   21   0  454.3812 7.3e-004  2e-003 0.00000  1.79980
  14*   0   24   0  454.3812 3.7e-006  3e-004 0.02389  0.03805


 Successful Termination After    14 Iterations
 GCONV convergence criterion satisfied.
 Criterion       454.3811585        Max Grad Entry  0.000290528
 Ridge (lambda)  0.023888796        TR Radius       0.038049758
 Act.dF/Pred.dF  1.007922555
 N. Function Calls        25        N. Gradient Calls         7
 N. Hessian Calls         16        Preproces. Time           0
 Time for Method           0        Effective Time            0


                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

              Parameter      Estimate   Gradient

               1 Intercept-1.56352973 -8.09e-006
               2 Alpha_1   9.76986195 -2.91e-004
               3 Gamma_1  -2.95937948  1.38e-005
```

```
4 Beta_1    0.23833660  1.44e-006
5 Delta     1.18702907  4.02e-005

Value of Objective Function =      454.381

                Hessian Matrix
                **************

          Symmetric Matrix: Dense Storage

S |          1          2          3          4          5
-------------------------------------------------------------
1 |  2.5715182 -1.40e-004 -1.91e-004  7.7087105  0.4908551
2 | -1.40e-004  4.70e-004  0.0000000  8.43e-004 -2.63e-004
3 | -1.91e-004  0.0000000 -2.39e-004  3.16e-004  5.82e-004
4 |  7.7087105  8.43e-004  3.16e-004  134.15224  4.4204392
5 |  0.4908551 -2.63e-004  5.82e-004  4.4204392  0.2892853

            Log Likelihood= -454.381
            SBC=-880.243 AIC=-898.762

      Covariance Matrix 2: H = (NOBS/d) inv(G)
      ****************************************

          Dense Symmetric Matrix (5 by 5)

      S |   Intercept    Alpha_1    Gamma_1     Beta_1      Delta
      ---------------------------------------------------------
Intercept |   0.0336666
  Alpha_1 |  -0.0295690  0.2319315
  Gamma_1 |   0.0104679 -0.0055272  0.0208672
   Beta_1 |   0.0094309 -0.0646167  0.0047090  0.0335773
    Delta |  -0.2785531  0.5637437 -0.1686795 -0.3327043  6.3247923

      Factor sigma=1 (nobs=300 df=5 vard=300)
            Determinant = 2.79972e-006
       Matrix has Only Positive Eigenvalues

   Robust Asymptotic Covariance Matrix of Estimates
   ************************************************

          Dense Symmetric Matrix (5 by 5)
```

```
        S |   Intercept    Alpha_1    Gamma_1     Beta_1      Delta
        ----------------------------------------------------------
Intercept |   0.0351807
  Alpha_1 |  -0.0283652  0.1171894
  Gamma_1 |   0.0222053 -0.0201169  0.0291104
   Beta_1 |   0.0133267 -0.0234699  0.0095243  0.0164274
    Delta |  -0.3641835  0.2552282 -0.3449748 -0.2426870  6.6053951


        Robust Asymptotic Correlation Matrix of Estimates
        *************************************************

                 Dense Symmetric Matrix (5 by 5)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1      Delta
        ----------------------------------------------------------
Intercept |   1.0000000
  Alpha_1 |  -0.4417626  1.0000000
  Gamma_1 |   0.6938757 -0.3444232  1.0000000
   Beta_1 |   0.5543539 -0.5349114  0.4355375  1.0000000
    Delta |  -0.7554725  0.2900916 -0.7867091 -0.7367370  1.0000000


         ***********************************************
         Parameter Estimates and Asympt. Standard Errors
         ***********************************************

       Parameter   Estimate    AStdErr    T Value   P Value
       Intercept  0.2093957  0.1875652  1.1163889  0.132582
         Alpha_1  0.9997429  0.3423293  2.9204126  0.001883
         Gamma_1  1.65e-005  0.1706176  9.67e-005  0.499961
          Beta_1  0.2383366  0.1281695  1.8595424  0.031973
           Delta  3.1349744  2.5700963  1.2197887  0.111760


             ***********************************
             Wald Confidence Limits (Alpha=0.05)
             ***********************************

           Parameter   Estimate      Lower      Upper
           Intercept  0.2093957 -0.1582253  0.5770166
             Alpha_1  0.9997429  0.3287897  1.6706960
             Gamma_1  1.65e-005 -0.3343878  0.3344208
              Beta_1  0.2383366 -0.0128710  0.4895442
               Delta  3.1349744 -1.9023218  8.1722706
          Factor sigma=1 (nobs=300 df=5 vard=300)
```

7. Method 4: EGARCH

```
m = q = p = 1;
print "Method 4: EGARCH: Nonlinear Constraints: DQNSQP";
optn = [ "cent"            ,
         "tech"      "sqp" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(4,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
print "SCOR=",scor;
```

```
****************************************
Model 4: EGARCH Modeling: NO=1 NQ=1 NP=1
****************************************
  O[1] =    1 Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .      0.0016
  Standard Deviation Y . . . . . .      1.1442
  Mean Squared Error (MSE) . . . . . .      1.3093
  Number of Back Casts . . . . . .         1
  Error Distribution . . . . . .    Normal
  Optimization Technique . . . . . .    DQNSQP

                  ******************
                  Optimization Start
                  ******************

                 Parameter Estimates
                 -------------------

Parameter      Estimate   Gradient   Grad. LF   Lower BC   Upper BC

 1 Intercept 0.11103067  5.20e-005  5.20e-005     .          .
 2 Alpha_1   0.50000000  0.6583880  0.6583880 -3.0000000  3.0000000
 3 Gamma_1  -0.20000000 -15.839860 -15.839860 -3.0000000  3.0000000
 4 Beta_1    0.50000000 -7.3873198 -7.3873198     .          .

           Value of Objective Function =      457.143
```

```
Value of Lagrange Function =      457.143

          Values of Nonlinear Constraints
          -------------------------------

             Constraint   Residual
                 1 1     0.4999800


                NLC Jacobian Matrix
                ******************

             Row Vector: Dense Storage

      R |           1          2          3          4
        0.0000000  0.0000000  0.0000000 -1.0000000


 DQNSQP (Modified VMCWD) Algorithm by Powell (1978, 1982)
       Lagrange Multiplier Update of Powell(1982)
         Gradient Computed by Finite Differences
         Jacobian Nonlinear Constraints (dense)
            (Computed by Finite Differences)


 Iteration Start:
 N. Variables              4
 N. Bound. Constr.         4        N. Mask  Constr.          0
 N. NonLin Constr.         1        NonLin EquConstr.         0
 Criterion       457.1429772        Max Grad Entry  15.83985967
 N. Active Constraints     0        Max Const Viol. 0.000000000


Iter rest   nfun  objfunc   conmax     pred     alfa    lfgmax
   1    0      4  456.134  0.00000  0.43037  0.06967  5.770163
   2    0      5  455.898  0.00000  0.45711  1.00000  5.238383
   3    0      6  455.683  0.00000  0.02384  1.00000  1.366673
   4    0      7  455.670  0.00000   6e-003  1.00000  0.670068
   5    0      8  455.666  0.00000   1e-003  1.00000  0.229266
   6    0      9  455.665  0.00000   4e-005  1.00000  0.110952
   7    0     10  455.665  0.00000   2e-006  1.00000  0.028218

 Successful Termination After     7 Iterations
 GCONV convergence criterion satisfied.
 Criterion       455.6652474        Max Grad Entry  0.028217578
 Lagrange Funct. 455.6652474        Max Grad LagF.  0.028217578
```

Max Const Viol.	0.000000000	N. Active Constraints	0
N. Function Calls	11	N. Gradient Calls	9
Preproces. Time	0	Time for Method	0
Effective Time	0		

****
Optimization Results
****

Parameter Estimates

Parameter		Estimate	Gradient	Grad. LF	Active BC
1	Intercept	0.06951210	-0.0282176	-0.0282176	
2	Alpha_1	0.45112468	-0.0015194	-0.0015194	
3	Gamma_1	-0.10095187	-0.0099995	-0.0099995	
4	Beta_1	0.64448549	-0.0046092	-0.0046092	

Value of Objective Function = 455.665
Value of Lagrange Function = 455.665

Values of Nonlinear Constraints

Constraint	Residual
1 1	0.3554945

NLC Jacobian Matrix
****

Row Vector: Dense Storage

R	1	2	3	4
	0.0000000	0.0000000	0.0000000	-1.0000000

Log Likelihood= -455.665
SBC=-888.515 AIC=-903.33

Covariance Matrix 2: H = (NOBS/d) inv(G)
****

```
            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1
        ------------------------------------------------
Intercept |   0.0024646
  Alpha_1 |   0.0016399  0.0175085
  Gamma_1 |  -2.99e-005  2.80e-004  0.0050599
   Beta_1 |  -0.0026972 -0.0043345 -3.90e-004  0.0134336

        Factor sigma=1 (nobs=300 df=4 vard=300)
             Determinant = 2.05744e-009
         Matrix has Only Positive Eigenvalues


    Robust Asymptotic Covariance Matrix of Estimates
    ************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1
        ------------------------------------------------
Intercept |   0.0020083
  Alpha_1 |   7.84e-004  0.0173408
  Gamma_1 |   5.04e-007 -1.67e-004  0.0038506
   Beta_1 |  -0.0019126 -8.94e-004 -0.0018338  0.0107178


   Robust Asymptotic Correlation Matrix of Estimates
   *************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1    Gamma_1     Beta_1
        ------------------------------------------------
Intercept |   1.0000000
  Alpha_1 |   0.1328165  1.0000000
  Gamma_1 |   1.81e-004 -0.0204093  1.0000000
   Beta_1 |  -0.4122337 -0.0656023 -0.2854585  1.0000000


    ************************************************
    Parameter Estimates and Asympt. Standard Errors
    ************************************************

  Parameter   Estimate    AStdErr    T Value   P Value
  Intercept  0.0695121  0.0448144  1.5511109  0.060972
    Alpha_1  0.4511247  0.1316844  3.4258023  0.000350
```

```
Gamma_1 -0.1009519  0.0620534 -1.6268539  0.947584
 Beta_1  0.6444855  0.1035270  6.2252882  0.000000


      **********************************
      Wald Confidence Limits (Alpha=0.05)
      **********************************

   Parameter   Estimate      Lower      Upper
   Intercept  0.0695121 -0.0183225  0.1573467
     Alpha_1  0.4511247  0.1930280  0.7092213
     Gamma_1 -0.1009519 -0.2225744  0.0206706
      Beta_1  0.6444855  0.4415763  0.8473947
    Factor sigma=1 (nobs=300 df=4 vard=300)
```

8. Method 5: AGARCH

```
m = .; q = p = 1;
print "Method 5: AGARCH: Reparametrized Constraints";
optn = [ "cent"            ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(5,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;


      **********************************
      Model 5: AGARCH Modeling: NQ=1 NP=1
      **********************************
            Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .         4
  Mean Y . . . . . .      0.0016
  Standard Deviation Y . . . . . .      1.1442
  Mean Squared Error (MSE) . . . . . .      1.3093
  Number of Back Casts . . . . . .         1
  Error Distribution . . . . . .    Normal
  Optimization Technique . . . . . .    TRUREG


                  ******************
```

Optimization Start
****

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	-0.80893295	-0.0156817
2	Alpha_1	-0.65145797	-8.63e-004
3	Gamma	0.12019977	-10.120248
4	Beta_1	0.08284350	-0.0055090

Value of Objective Function = 455.504

Hessian Matrix
****

Symmetric Matrix: Dense Storage

S	1	2	3	4
1	60.094827	2.1424727	9.0724793	24.498126
2	2.1424727	4.3238422	-2.4711464	1.3176409
3	9.0724793	-2.4711464	63.271550	4.0454594
4	24.498126	1.3176409	4.0454594	12.970194

Trust Region Optimization
Without Parameter Scaling
Gradient Computed by Finite Differences
Hessian Computed by Finite Differences (dense)
(Using Only Function Calls)

Iteration Start:

N. Variables	4		
Criterion	455.5035429	Max Grad Entry	10.12024794
TR Radius	1.000000000		

Iter	rest	nfun	act	optcrit	difcrit	maxgrad	lambda	radius
1	0	2	0	454.7243	0.779193	0.90219	0.00000	1.00000
2	0	3	0	454.6731	0.051214	0.14599	0.00000	0.20814
3	0	4	0	454.6727	4.2e-004	1e-003	0.00000	0.15570
4	0	5	0	454.6727	4.0e-008	1e-006	0.00000	0.01078

```
Successful Termination After     4 Iterations
GCONV convergence criterion satisfied.
Criterion       454.6727129        Max Grad Entry  1.4851e-006
Ridge (lambda)  0.000000000        TR Radius       0.010779429
Act.dF/Pred.dF  0.997041601
N. Function Calls         6        N. Gradient Calls         7
N. Hessian Calls          6        Preproces. Time           0
Time for Method           0        Effective Time            0
```

****
Optimization Results
****

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	-0.82184321	7.21e-008
2	Alpha_1	-0.68460525	1.31e-006
3	Gamma	0.28751079	-2.37e-007
4	Beta_1	0.04295362	1.49e-006

Value of Objective Function = 454.673

Hessian Matrix
****

Symmetric Matrix: Dense Storage

S	1	2	3	4
1	59.097069	3.0961381	12.899948	24.465324
2	3.0961381	4.6931412	5.4377903	2.0330739
3	12.899948	5.4377903	61.523260	6.6195508
4	24.465324	2.0330739	6.6195508	13.145823

Log Likelihood= -454.673
SBC=-886.53 AIC=-901.345

Covariance Matrix 2: H = (NOBS/d) inv(G)
****

Dense Symmetric Matrix (4 by 4)

S	Intercept	Alpha_1	Gamma	Beta_1
Intercept	0.0144828			
Alpha_1	0.0013327	0.0123990		
Gamma	-9.80e-004	-0.0054169	0.0302728	
Beta_1	-0.0109716	-0.0083462	0.0020905	0.0149192

Factor sigma=1 (nobs=300 df=4 vard=300)
Determinant = 1.0427e-008
Matrix has Only Positive Eigenvalues

Robust Asymptotic Covariance Matrix of Estimates
****

Dense Symmetric Matrix (4 by 4)

S	Intercept	Alpha_1	Gamma	Beta_1
Intercept	0.0087005			
Alpha_1	-0.0011078	0.0119599		
Gamma	-3.57e-004	-0.0048228	0.0235804	
Beta_1	-0.0048669	-0.0059886	0.0019479	0.0080261

Robust Asymptotic Correlation Matrix of Estimates
****

Dense Symmetric Matrix (4 by 4)

S	Intercept	Alpha_1	Gamma	Beta_1
Intercept	1.0000000			
Alpha_1	-0.1085972	1.0000000		
Gamma	-0.0249095	-0.2871868	1.0000000	
Beta_1	-0.5824019	-0.6112349	0.1415926	1.0000000

****
Parameter Estimates and Asympt. Standard Errors
****

Parameter	Estimate	AStdErr	T Value	P Value
Intercept	0.4396206	0.0932766	4.7130849	0.000002

```
Alpha_1  0.3351672  0.1093611  3.0647750  0.001190
  Gamma  0.2152178  0.1535592  1.4015295  0.081052
 Beta_1  0.3394524  0.0895886  3.7890126  0.000092


          **********************************
          Wald Confidence Limits (Alpha=0.05)
          **********************************

        Parameter   Estimate      Lower      Upper
        Intercept  0.4396206  0.2568018  0.6224394
          Alpha_1  0.3351672  0.1208234  0.5495110
            Gamma  0.2152178 -0.0857528  0.5161883
           Beta_1  0.3394524  0.1638619  0.5150429
         Factor sigma=1 (nobs=300 df=4 vard=300)
```

9. Method 6: NAGARCH:

```
m = .; q = p = 1;
print "Method 6: NAGARCH: Reparametrized Constraints";
optn = [ "cent"            ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(6,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;


          **********************************
          Model 6: NAGARCH Modeling: NQ=1 NP=1
          **********************************
                Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .       4
  Mean Y . . . . . .      0.0016
  Standard Deviation Y . . . . . .      1.1442
  Mean Squared Error (MSE) . . . . . .      1.3093
  Number of Back Casts . . . . . .         1
  Error Distribution . . . . . .    Normal
  Optimization Technique . . . . . .    TRUREG
```

```
                  ******************
                  Optimization Start
                  ******************

                 Parameter Estimates
                 -------------------

          Parameter      Estimate   Gradient

           1 Intercept-0.80893295 -0.0156817
           2 Alpha_1  -0.65145797 -8.63e-004
           3 Gamma     0.00000000 -6.3343419
           4 Beta_1    0.08284350 -0.0055090

        Value of Objective Function =      455.504


                    Hessian Matrix
                    **************

          Symmetric Matrix: Dense Storage

        S |           1          2          3          4
        ---------------------------------------------------
        1 |   60.094788  2.1426024  3.5418457  24.498126
        2 |   2.1426024  4.3231317 -1.1918934  1.3178576
        3 |   3.5418457 -1.1918934  26.798373  2.0994381
        4 |   24.498126  1.3178576  2.0994381  12.972177


             Trust Region Optimization
             Without Parameter Scaling
        Gradient Computed by Finite Differences
      Hessian Computed by Finite Differences (dense)
             (Using Only Function Calls)


 Iteration Start:
 N. Variables             4
 Criterion       455.5035429        Max Grad Entry  6.334341932
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1    0    2   0  454.7537 0.749798 0.42360 0.00000  1.00000
   2    0    3   0  454.7329 0.020876 0.01323 0.00000  0.26611
   3    0    4   0  454.7329 3.4e-006  8e-006 0.00000  0.09923
```

```
Successful Termination After     3 Iterations
GCONV convergence criterion satisfied.
Criterion       454.7328650        Max Grad Entry  8.2550e-006
Ridge (lambda)  0.000000000        TR Radius       0.099227568
Act.dF/Pred.dF  0.999981427
N. Function Calls         5        N. Gradient Calls         7
N. Hessian Calls          5        Preproces. Time           0
Time for Method           0        Effective Time            0
```

****
Optimization Results
****

Parameter Estimates

	Parameter	Estimate	Gradient
1	Intercept	-0.79882912	3.31e-007
2	Alpha_1	-0.66456439	-3.95e-006
3	Gamma	0.25256726	-8.25e-006
4	Beta_1	0.04328314	1.30e-006

Value of Objective Function = 454.733

Hessian Matrix
****

Symmetric Matrix: Dense Storage

S	1	2	3	4
1	62.232247	2.7850828	-0.2077795	25.072832
2	2.7850828	4.6852822	1.7877625	1.8948846
3	-0.2077795	1.7877625	21.513094	0.2034453
4	25.072832	1.8948846	0.2034453	13.092182

Log Likelihood= -454.733
SBC=-886.651 AIC=-901.466

Covariance Matrix 2: H = (NOBS/d) inv(G)
****

```
            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1      Gamma     Beta_1
        ------------------------------------------------
Intercept |   0.0144757
  Alpha_1 |   0.0012657  0.0122713
    Gamma |   3.14e-004 -0.0068194  0.0290811
   Beta_1 |  -0.0110178 -0.0081967  0.0017286  0.0149454

         Factor sigma=1 (nobs=300 df=4 vard=300)
               Determinant = 9.23947e-009
          Matrix has Only Positive Eigenvalues


    Robust Asymptotic Covariance Matrix of Estimates
    ************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1      Gamma     Beta_1
        ------------------------------------------------
Intercept |   0.0088774
  Alpha_1 |  -0.0010126  0.0120685
    Gamma |   6.46e-004 -0.0073773  0.0258309
   Beta_1 |  -0.0051584 -0.0060528  0.0024665  0.0083331


   Robust Asymptotic Correlation Matrix of Estimates
   *************************************************

            Dense Symmetric Matrix (4 by 4)

        S |   Intercept    Alpha_1      Gamma     Beta_1
        ------------------------------------------------
Intercept |   1.0000000
  Alpha_1 |  -0.0978309  1.0000000
    Gamma |   0.0426414 -0.4178324  1.0000000
   Beta_1 |  -0.5997496 -0.6035727  0.1681129  1.0000000


    ***********************************************
    Parameter Estimates and Asympt. Standard Errors
    ***********************************************

  Parameter   Estimate    AStdErr    T Value   P Value
  Intercept  0.4498554  0.0942199  4.7745255  0.000001
```

```
Alpha_1  0.3267569  0.1098565  2.9743965  0.001589
  Gamma  0.1986172  0.1607199  1.2357977  0.108757
 Beta_1  0.3372187  0.0912858  3.6940963  0.000131


          **********************************
          Wald Confidence Limits (Alpha=0.05)
          **********************************

        Parameter   Estimate      Lower      Upper
        Intercept  0.4498554  0.2651877  0.6345230
          Alpha_1  0.3267569  0.1114420  0.5420718
            Gamma  0.1986172 -0.1163879  0.5136224
           Beta_1  0.3372187  0.1583017  0.5161357
         Factor sigma=1 (nobs=300 df=4 vard=300)
```

10. Method 7: IGARCH use $\delta = 2$:

```
m = .; q = p = 1;
print "Method 7: IGARCH: Reparametrized Constraints: Delta=2";
optn = [ "cent"            ,
         "delta"        2. ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(7,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;


          **********************************
          Model 7: IGARCH Modeling: NQ=1 NP=1
          **********************************
           Modeling Squared Values (Delta=2)
                Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .       300
  Number of Parameter Estimates. . . .       2
  Mean Y . . . . . .      0.0016
  Standard Deviation Y . . . . . .      1.1442
  Mean Squared Error (MSE) . . . . . .      1.3093
  Number of Back Casts . . . . . .         1
  Error Distribution . . . . . .    Normal
```

```
 Optimization Technique . . . . . .    TRUREG

                  ******************
                  Optimization Start
                  ******************

                 Parameter Estimates
                 -------------------

           Parameter      Estimate   Gradient

            1 Intercept-4.33236311 -0.1900006
            2 Alpha_1  -2.19722238 -1.5994832

       Value of Objective Function =      466.563


                    Hessian Matrix
                    **************

            Symmetric Matrix: Dense Storage

                 S |           1          2
                 --------------------------
                 1 |   1.7309304 -3.4318452
                 2 |  -3.4318452  5.0825081


              Trust Region Optimization
              Without Parameter Scaling
         Gradient Computed by Finite Differences
      Hessian Computed by Finite Differences (dense)
               (Using Only Function Calls)


 Iteration Start:
 N. Variables             2
 Criterion       466.5625441        Max Grad Entry  1.599483213
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
  1*   0    4   0  465.4522 1.110313 1.56396 0.65221  4.01012
  2*   0    5   0  464.5735 0.878697 10.6363 1.09837  2.00506
  3    0    6   0  460.5322 4.041335 5.78896 0.00000  2.00548
  4    0    7   0  460.1095 0.422739 0.11949 0.00000  1.66177
```

```
  5    0    8   0  460.1076 1.9e-003  6e-003 0.00000  0.14209
  6    0    9   0  460.1076 6.1e-007  2e-006 0.00000  0.03490

Successful Termination After     6 Iterations
GCONV convergence criterion satisfied.
Criterion       460.1076084        Max Grad Entry  2.1990e-006
Ridge (lambda)  0.000000000        TR Radius       0.034904588
Act.dF/Pred.dF  0.999530876
N. Function Calls        10        N. Gradient Calls         3
N. Hessian Calls          8        Preproces. Time           0
Time for Method           0        Effective Time            0

                    ********************
                    Optimization Results
                    ********************

                    Parameter Estimates
                    -------------------

          Parameter      Estimate   Gradient

           1 Intercept-1.13300176  7.31e-007
           2 Alpha_1   0.74808044 -2.20e-006

       Value of Objective Function =      460.108

                       Hessian Matrix
                       **************

            Symmetric Matrix: Dense Storage

                 S |           1          2
                 --------------------------
                 1 |   38.015479 -17.841882
                 2 |  -17.841882  11.906938


              Log Likelihood= -460.108
              SBC=-908.808 AIC=-916.215


         Covariance Matrix 2: H = (NOBS/d) inv(G)
         ****************************************

             Dense Symmetric Matrix (2 by 2)
```

```
               S |   Intercept    Alpha_1
                 -------------------------
       Intercept |   0.0092191
         Alpha_1 |   0.0093608  0.0135003

    Factor sigma=1 (nobs=300 df=2 vard=300)
         Determinant = 3.68368e-005
     Matrix has Only Positive Eigenvalues


 Robust Asymptotic Covariance Matrix of Estimates
 ************************************************

        Dense Symmetric Matrix (2 by 2)

               S |   Intercept    Alpha_1
                 -------------------------
       Intercept |   0.0074443
         Alpha_1 |   0.0070108  0.0101716


 Robust Asymptotic Correlation Matrix of Estimates
 *************************************************

        Dense Symmetric Matrix (2 by 2)

               S |   Intercept    Alpha_1
                 -------------------------
       Intercept |   1.0000000
         Alpha_1 |   0.8056720  1.0000000


  ***********************************************
  Parameter Estimates and Asympt. Standard Errors
  ***********************************************

Parameter   Estimate    AStdErr    T Value   P Value
Intercept  0.3220650  0.0862806  3.7327625  0.000113
  Alpha_1  0.6786245  0.1008545  6.7287453  0.000000


       ***********************************
       Wald Confidence Limits (Alpha=0.05)
       ***********************************

    Parameter   Estimate      Lower      Upper
```

```
Intercept  0.3220650  0.1529581  0.4911719
  Alpha_1  0.6786245  0.4809533  0.8762958
Factor sigma=1 (nobs=300 df=2 vard=300)
```

11. Method 7: IGARCH use $\delta = 1$:

```
m = .; q = p = 1;
print "Method 7: IGARCH: Reparametrized Constraints: Delta=1";
optn = [ "cent"            ,
         "delta"        1. ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
< gof,est,ase,cis,cov,scor > = garch(7,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
**********************************
Model 7: IGARCH Modeling: NQ=1 NP=1
**********************************
Modeling Absolute Values (Delta=1)
      Q[1] =    1 P[1] =    1

Number of Observations . . . . . .      300
Number of Parameter Estimates. . . .      2
Mean Y . . . . . .      0.0016
Standard Deviation Y . . . . . .      1.1442
Mean Squared Error (MSE) . . . . . .      1.3093
Number of Back Casts . . . . . .      1
Error Distribution . . . . . .    Normal
Optimization Technique . . . . . .    TRUREG

                  ******************
                  Optimization Start
                  ******************

                 Parameter Estimates
                 -------------------

           Parameter      Estimate   Gradient
```

```
1 Intercept-4.93814434 -8.5560129
2 Alpha_1  -2.94443689  8.2633134

Value of Objective Function =      470.207

                Hessian Matrix
                **************

          Symmetric Matrix: Dense Storage

               S |           1          2
               --------------------------
               1 |   5.4931025 -8.2545947
               2 |  -8.2545947  10.430961


            Trust Region Optimization
            Without Parameter Scaling
       Gradient Computed by Finite Differences
     Hessian Computed by Finite Differences (dense)
            (Using Only Function Calls)


 Iteration Start:
 N. Variables              2
 Criterion       470.2067871        Max Grad Entry  8.556012903
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1*   0    3   0  467.5957 2.611061 4.07006 9.42808  0.49607
   2*   0    7   0  465.6413 1.954411 1.35533 0.32023  5.91678
   3    0    9   0  464.8224 0.818955 0.33783 0.94923  0.85436
   4    0   10   0  464.7775 0.044887 0.07254 0.00000  0.90206
   5    0   11   0  464.7774 1.2e-004  4e-004 0.00000  0.25776
   6    0   12   0  464.7774 1.7e-009  3e-008 0.00000  0.01028

 Successful Termination After     6 Iterations
 GCONV convergence criterion satisfied.
 Criterion       464.7773539        Max Grad Entry  2.6987e-008
 Ridge (lambda)  0.000000000        TR Radius       0.010284096
 Act.dF/Pred.dF  0.999686729
 N. Function Calls        13        N. Gradient Calls         3
 N. Hessian Calls          8        Preproces. Time           0
 Time for Method           0        Effective Time            0
```

```
********************
Optimization Results
********************

Parameter Estimates
-------------------

    Parameter      Estimate   Gradient

     1 Intercept-2.69415957 -8.89e-009
     2 Alpha_1  -1.26091968 -2.70e-008

Value of Objective Function =      464.777


             Hessian Matrix
             **************

      Symmetric Matrix: Dense Storage

          S |           1          2
          --------------------------
          1 |   48.759932 -46.781661
          2 |  -46.781661  47.834734


        Log Likelihood= -464.777
        SBC=-918.147 AIC=-925.555


  Covariance Matrix 2: H = (NOBS/d) inv(G)
  ****************************************

      Dense Symmetric Matrix (2 by 2)

             S |   Intercept    Alpha_1
             -------------------------
     Intercept |   0.0015155
       Alpha_1 |   0.0037710  0.0100023

   Factor sigma=1 (nobs=300 df=2 vard=300)
        Determinant = 9.37374e-007
    Matrix has Only Positive Eigenvalues


Robust Asymptotic Covariance Matrix of Estimates
************************************************
```

```
                Dense Symmetric Matrix (2 by 2)

                       S |   Intercept    Alpha_1
                       --------------------------
                Intercept |   0.0071639
                  Alpha_1 |   0.0190380  0.0512523


          Robust Asymptotic Correlation Matrix of Estimates
          ************************************************

                Dense Symmetric Matrix (2 by 2)

                       S |   Intercept    Alpha_1
                       --------------------------
                Intercept |   1.0000000
                  Alpha_1 |   0.9935531  1.0000000


           ********************************************
           Parameter Estimates and Asympt. Standard Errors
           ********************************************

         Parameter   Estimate    AStdErr    T Value   P Value
         Intercept  0.0675992  0.0846398  0.7986692  0.212559
           Alpha_1  0.2207715  0.2263898  0.9751828  0.165130


                 *********************************
                 Wald Confidence Limits (Alpha=0.05)
                 *********************************

              Parameter   Estimate      Lower      Upper
              Intercept  0.0675992 -0.0982917  0.2334901
                Alpha_1  0.2207715 -0.2229444  0.6644873
               Factor sigma=1 (nobs=300 df=2 vard=300)
```

12. Method 8: FIGARCH: Truncated Lags=100

```
m = .; q = p = 1;
print "Method 8: FIGARCH: Reparametrized Constraints: TRUNCL=100";
optn = [ "cent"           ,
         "truncl"      100 ,
         "tech"       "tr" ,
         "error"    "norm" ,
         "print"         3 ];
```

```
< gof,est,ase,cis,cov,scor > = garch(8,two,.,m,q,p,optn);
print "GOF=",gof;
print "EST=",est;
print "ASE=",ase;
print "CIs=",cis;
print "COV=",cov;
```

```
            ************************************
            Model 8: FIGARCH Modeling: NQ=1 NP=1
            ************************************
                  Q[1] =    1 P[1] =    1

  Number of Observations . . . . . .        300
  Number of Parameter Estimates. . . .        4
  Mean Y . . . . . .                      0.0016
  Standard Deviation Y . . . . . .        1.1442
  Mean Squared Error (MSE) . . . . . .    1.3093
  Number of Weights in ARCH(oo). . . .      100
  Error Distribution . . . . . .         Normal
  Optimization Technique . . . . . .     TRUREG

                   ******************
                   Optimization Start
                   ******************

                  Parameter Estimates
                  -------------------

            Parameter      Estimate   Gradient

             1 Intercept-0.88588323  0.3649173
             2 Phi       1.38629436 -0.4245310
             3 D        -1.38629436  0.2307636
             4 Beta      0.00000000  1.1283417

          Value of Objective Function =      457.638


                     Hessian Matrix
                     **************

           Symmetric Matrix: Dense Storage

          S |          1          2          3          4
          ---------------------------------------------------
```

```
  1 |   19.991728  0.3046813  10.991260 -0.7993952
  2 |   0.3046813  0.2911064  0.5745450 -0.2276935
  3 |   10.991260  0.5745450  7.0245531 -0.1297626
  4 |  -0.7993952 -0.2276935 -0.1297626  2.8163192


                Trust Region Optimization
                Without Parameter Scaling
          Gradient Computed by Finite Differences
        Hessian Computed by Finite Differences (dense)
                (Using Only Function Calls)


 Iteration Start:
 N. Variables              4
 Criterion       457.6376615        Max Grad Entry  1.128341659
 TR Radius       1.000000000


Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius
   1    0    2   0  457.1540 0.483708 0.21628 0.22278  1.00000
   2    0    5   0  456.9815 0.172470 0.51927 0.01525  2.30785
   3    0    6   0  456.8819 0.099625 0.15624 0.03489  1.00857
   4    0    7   0  456.8402 0.041700 0.15303  8e-003  1.01691
   5    0    8   0  456.8237 0.016415  9e-003 0.00000  1.01747
   6    0    9   0  456.8183 5.5e-003  7e-003 0.00000  1.05950
   7    0   10   0  456.8162 2.1e-003  4e-003 0.00000  0.98833
   8    0   11   0  456.8155 7.6e-004  2e-003  2e-004  0.99528
   9    0   12   0  456.8152 2.8e-004  4e-004  3e-004  0.99560
  10    0   13   0  456.8151 4.6e-005  1e-004 0.00000  0.99560
  11    0   14   0  456.8151 4.6e-005  1e-004 0.00000  1.09166
  12    0   15   0  456.8151 3.6e-005  2e-004 0.00000  1.16833
  13    0   16   0  456.8151 5.8e-006  4e-005 0.00000  0.66855

 Successful Termination After    13 Iterations
 GCONV convergence criterion satisfied.
 Criterion       456.8150627        Max Grad Entry  3.8784e-005
 Ridge (lambda)  0.000000000        TR Radius       0.668550915
 Act.dF/Pred.dF  1.832806390
 N. Function Calls        17        N. Gradient Calls         3
 N. Hessian Calls         15        Preproces. Time           0
 Time for Method           1        Effective Time            1


                    ********************
                    Optimization Results
                    ********************
```

Parameter Estimates

Parameter		Estimate	Gradient
1	Intercept	-0.46297451	-3.88e-005
2	Phi	11.8851377	-3.17e-005
3	D	-2.60537231	-3.74e-007
4	Beta	-0.20489682	3.46e-005

Value of Objective Function = 456.815

Hessian Matrix

Symmetric Matrix: Dense Storage

S	1	2	3	4
1	47.152502	6.17e-005	11.418389	-2.6275098
2	6.17e-005	1.26e-004	7.51e-005	7.49e-005
3	11.418389	7.51e-005	2.8779569	-0.5813992
4	-2.6275098	7.49e-005	-0.5813992	3.0395612

Log Likelihood= -456.815
SBC=-890.815 AIC=-905.63

Covariance Matrix 2: H = (NOBS/d) inv(G)

Dense Symmetric Matrix (4 by 4)

S	Intercept	Phi	D	Beta
Intercept	-0.0345733			
Phi	-0.0806959	-0.0525482		
D	0.0294820	0.0394115	-0.0175934	
Beta	-0.0364651	-0.0143724	0.0169448	0.0052774

Factor sigma=1 (nobs=300 df=4 vard=300)
Determinant = -3.18551e-008
Matrix has 1 Negative Eigenvalue(s)

Robust Asymptotic Covariance Matrix of Estimates

```
*********************************************

          Dense Symmetric Matrix (4 by 4)

       S |   Intercept        Phi          D       Beta
       ---------------------------------------------------
Intercept |   0.1283577
      Phi |   0.0360338  0.0295502
        D |  -0.0518958 -0.0182783  0.0226552
     Beta |   0.0053435  0.0109712 -0.0034809  0.0075612


  Robust Asymptotic Correlation Matrix of Estimates
  *************************************************

          Dense Symmetric Matrix (4 by 4)

       S |   Intercept        Phi          D       Beta
       ---------------------------------------------------
Intercept |   1.0000000
      Phi |   0.5850856  1.0000000
        D |  -0.9623608 -0.7064356  1.0000000
     Beta |   0.1715219  0.7339742 -0.2659561  1.0000000


   *********************************************
   Parameter Estimates and Asympt. Standard Errors
   *********************************************

  Parameter   Estimate    AStdErr    T Value   P Value
  Intercept  0.6294087  0.3582705  1.7567975  0.039993
        Phi  0.4656001  0.1719017  2.7085257  0.003576
          D  0.0687935  0.1505163  0.4570499  0.323985
       Beta  0.2399182  0.0869549  2.7591106  0.003078


         ***********************************
         Wald Confidence Limits (Alpha=0.05)
         ***********************************

      Parameter   Estimate      Lower      Upper
      Intercept  0.6294087 -0.0727886  1.3316059
            Phi  0.4656001  0.1286790  0.8025211
              D  0.0687935 -0.2262131  0.3638000
           Beta  0.2399182  0.0694897  0.4103468
       Factor sigma=1 (nobs=300 df=4 vard=300)
```

5.7 Function `jarbera`

prob = jarbera(y<,optn>)

< prob,stat > = jarbera(y<,optn>)

Purpose: Implements the Jarque-Bera test for normally distributed data.

Input: y The first input object y must be either a n vector or a $n \times nc$ matrix. If y is a vector the scalar value of the probability of the CDF is returned, if y is a matrix then a vector of nc probabilities is returned by using each column of y separately.

optn this is a vector of options which should be initialized with missing values for defaults:

1. (int) amount of printed output (default is no output);
2. (real) probability for test (default is 0.05), must be in $[0, 1]$;
3. (int) number of depedent variables (default=2).

Output: prob The first result is either a scalar or an object of the test probabilities.

stat The `stat` result has the same size as `prob` and returns the test statistics (only for `optn[3]=0`).

Restrictions: 1. Missing value are returned if the input y contains missing data.

2.

Relationships: berkow(), ksprob(), kstest(), shapwilk()

Examples: 1. Test $N = 1000$ random uniform generated data:

```
   options NOECHO;
#include "..\\tdata\\matlb.dat"
   options ECHO;


optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           2 ]; /* nvar */
< p1,s1 > = jarbera(uf1000,optn);
print "JARBERA P:",p1;
print "JARBERA S:",s1;
```

```
     Hypothesis H0: Data are uniformally distributed versus
       Hypothesis H1: Data are not uniformally distributed
    H0 rejected: Probability=7.38734e-014 Statistics=60.4728
```

2. Test $N = 1000$ random $(0, 1)$-normal generated data:

```
nr = 500; nc = 5;
print "\n *** nr,nc=",nr,nc," ***\n";
options RANDUNI;
mu1 = rand(nr,nc,'g',"norm");
```

```
     Hypothesis H0: Data are uniformally distributed versus
       Hypothesis H1: Data are not uniformally distributed

               Col  Probability   Statistics Rej/Acc
                 1  0.870686046  0.276947638  Accept
                 2  0.575339062  1.105591478  Accept
                 3  0.732156146  0.623522948  Accept
                 4  0.745872441  0.586401367  Accept
                 5  0.677028247  0.780084567  Accept
```

5.8 Function `mucomp`

< gof,BF > = mucomp(data,nres,modl<,optn>)

< gof,lik,est,fpbar,BF > = mucomp(data,modres,order,rest<,optn>)

Purpose: The `mucomp` function implements three algorithms for comparison of means (confirmatory ANOVA) as it is available in the Fortran program by Kuiper, Klugkist, & Hojtink (2009). The data set is usually a vector x of N observations grouped into g groups. The common 1-way ANOVA model,

$$y_{ij} = \mu_i d_{ij} + \epsilon_{ij}, \quad i = 1, \ldots, g, j = 1, \ldots, N_i$$

where d_{ij} is observation j belonging to group i, and the error is normal distributed $\epsilon_{ij} \sim \mathcal{N}(\prime, \sigma^{\in})$, tests the null hypothesis that all g means are equal $\mu_1 = \ldots = \mu_g$ against the alternative that the means are not equal: $\mu_1, \ldots, \mu_g$. This method however, was extended to permit a wider variety of order relations among the means, e.g. testing models of the form:

$$\mu_5 = \mu_3 > (\mu_1, \mu_4) < \mu_2$$

As in the Fortran program by Kuiper, Klugkist, & Hojtink three different algorithms are implemented:

Fbar an algorithm for estimating the `Fbar` and `pFbar` criterion as shown in Silvapulle & Sen (2005).

ORIC an algorithm for computing the order restricted information criterion ORIC proposed by Anraku (1999).

BMS an algorithm for Bayesian model selection, see Klugkist, Laudy, & Hoijtink (2005).

For the Fbar and ORIC algorithms a set of optimal estimates (means satisfying the constraints) and log likelihood values for each model is computed. The ORIC and BMS algorithms permit to rank the models.

The method can be extended to deal with 2 or more way classification problems.

Input: data The data matrix consists of either one or two columns.

2 column data : One of the two columns (specified by an option) must contain an integer ID variable indicating the group membership. This ID variable must not have consecutive values. Groups are specified by identical ID values.

1 column data : Here we assume the total number of observations N is a multiple of the number g of groups. For the group membership the data must then be arranged by one of the orders :

block blockwise: [1 1 1 2 2 2 3 3 3],

split splitwise: [1 2 3 1 2 3 1 2 3],

random random: [3 1 2 2 1 3 3 2 1].

The user may specify in which order the data are arranged.

nres This must be a m vector nr of positive integers specifying how many restrictions are specified for each of the models $k = 1, \ldots, m$ with the third input argument. Let $NR = \sum_k^m nr_k$ be the total number of restrictions for all models.

modl There are two different forms of the specification of restrictions on the means μ_j, the *tensor* and the easier *string* specification:

tensor The tensor `tmod[2,NR,g]` consists of two matrices of integers and is the same as the matrix form developed and described in detail by Kuiper, Klugkist, & Hoijtink (2009).

1. The first $NR \times g$ matrix of integers specifies the order of groups in the NR restrictions. The rows of this matrix have to ordered as specified by the `modres` input argument.
2. The second $NR \times g$ matrix of integers specifies the NR model restrictions.

The meaning of the integers in the restriction matrix is the following:

- -1: defines the ¡ relationship;
- -3: defines the ¿ relationship;
- 0: defines a singleton (unused by the restriction);
- positive integers: specify the group number of equality constrained clusters of variables.

string This must be a vector of NR strings where each of the strings formulates restrictions in a clearly readable form.

Even though the string seems to be much nicer for applications, the matrix or tensor specification may have some use when the input arguments are created by some other software rather than writing them down by the user. See the example below for more details.

optn The `optn` argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content. Many of the options have default values and must not be specified for many applications. However, using the string input of the model requires to specify the number g of groups, which must not necessarily be determined with the model input.

Option	Second Column	Meaning
"block"		block sorted 1-column input data
"bms"		perform BMS algorithm (not default)
"delta"	real	only for BMS: $Delta \geq 0$, default=0
"fbar"		perform Fbar algorithm (is default)
"group"		specifies number g of groups (string model)
"idvar"	int	column number containing ID variable default is first column
"nopr"		suppress all printed output
"oric"		perform ORIC algorithm (not default)
"nsamp"	int	number of random samples for Fbar and ORIC default is $n_s = 100,000$
"pinp"		print the model specification in matrix form
"print"	int	amount of printed output =0 specifies no printed output, default is 1
"pv"	real	only for BMS: prior vagueness, $pv > 0$, default=2
"qpnusp"		perform null space QP algorithm
"qppogi"		perform Powell's Goldfarb and Idnani QP algorithm
"rand"		random sorted 1-column input data
"split"		split sorted 1-column input data
"seed"	int	seed for random generator, default=time of day
"sing"	real	singularity criterion, default=1.e-8

Output: **gof** a vector of scalar results, see below for content;

lik return only for the Fbar and ORIC methods:

- for Fbar method: the $m \times 2$ matrix containing the log likelihood values and the residual sum of squares for all models in its two columns;
- for ORIC method: $m \times 4$ matrix with an additional two matrices containing the penalty term and the ORIC value (which is the sume of log likelihood and penalty term). The value of the penalty term is in $[2, g+1]$, the lower value for the null hypothesis (all means are equal) und $g + 1$ for the unconstrained solution.

est return only for Fbar and ORIC: this is the $m \times g$ matrix of parameter estimates for the m models which should satisfy the specified restrictions.

fpbar only for Fbar: contains the Fbar and corresponding p values for the hypotheses.

BF only for BMS: this is the $m \times 2$ matrix of Bayes factors BF_k in its first column and the *posterior model probabilities* pmp_k, $k = 1, \ldots, m$, which follow directly from the Bayes factors:

$$pmp_k = BF_k / \sum_k^M BF_k \quad , \quad k = 1, \ldots, m$$

That means $0 \leq pmp_k \leq 1$ and the model with the highest pmp_k value should be the most fitting w.r.t. to the data.

Restrictions: 1. The input data must be numeric and cannot contain any string or complex data.

2. Currently the input data must not have any missing values.

Relationships: anova(), hbanova()

Examples: 1. Example by Lucas (2003): Tensor Input:

```
data = [ 1   3.58, 1  -0.15, 1   0.67, 1   2.22, 1   2.56, 1   1.7 , 1  -0.45, 1
         1   4.83, 1   2.44, 1   6.74, 1   1.4 , 1  -0.03, 1   0.71, 1   4.3 , 1
         1   2.44, 1   0.81, 1   3.65, 1   1.41, 1  -0.38, 1   3.11, 1   0.21, 1
         1   2.63, 1   4.74, 1   2.12, 1   4.36, 1   3.26, 1   0.94, 2   1.67, 2
         ..........................................................................
         5   1.38, 5   4.58, 5   3.48, 5   0.39, 5   3.89, 5   4.18, 5   3.72, 5
         5   3.63, 5   1.55, 5  -0.46, 5   2.04, 5   2.26, 5   4.27, 5   5.67, 5
         5   2.17, 5   2.27, 5   4.1 , 5   3.74, 5   4.54, 5   1.71, 5   4.74, 5
         5   4.1 , 5   4.8 , 5   2.9 , 5   2.35, 5   4.09, 5   1.09 ];
```

The model specification contains one restriction row for all models (hypotheses) except for the second which takes two rows:

```
nres = [ 1  2  1  1 ];
nmod = dim(nres);
```

The following is the string model specification:

```
print "String Model Specification without Singletons";
modl = [ "1 = 2 = 3 = 4 = 5",
         "5 = 3 > 1 > 2 ",
         "3 > 4 > 2 ",
         "3 > 1 > 4 = 5 > 2",
         " " ];
```

The following string specification is equivalent but closer to the matrix spaecification used in Kuiper, Klugkist, and Hojtink (2009);

```
print "Model specification with Singletons";
modl = [ "1 = 2 = 3 = 4 = 5",
         "5 = 3 > 1 > 2,  4",
         "3 > 4 > 2,  1,  5",
         "3 > 1 > 4 = 5 > 2",
         "1 , 2 , 3 , 4 , 5" ];
```

The group number must be specified with the string model specification. The default sample size for the simulation with Fbar (p value) and ORIC is 100,000 but is here specified to only 1000. The `"seed"` specification ensures replicable results.

```
print "Fbar, ORIC, and BMS together";
optns = [ "bms"            ,
          "fbar"           ,
          "oric"           ,
          "group"        5 ,
          "samp"      1000 ,
          "seed"       123 ,
          "delt"       0.3 ,
          "pv"         2.0 ,
          "pinput"         ,
          "print"        2 ];
```

The function starts with the output of some data properties, where the means are the free (unconstrained) parameter estimates (as with specified model 4):

```
                    **********************
                    Summary of Input Data
                    **********************

               Group    Nobs        Mean      StdDev

                   1      30  2.32933333  1.86012223
                   2      30  1.32800000  1.14947484
                   3      30  3.20000000  1.78986418
                   4      30  2.23066667  1.45044805
                   5      30  3.22900000  1.50012264
```

The following is the output generated for the `pinp` option:

```
      ************************************************
      Ordering and Restrictions for 4 Models and 5 Groups
      ************************************************

               Model  NR Orders and Restrictions

                   1   1   1   2   3   4   5
                           1   1   1   1   1
                   2   1   5   3   1   2   4
                           1   1  -3  -3   0
                       2   3   4   2   1   5
```

```
                1  -3  -3   0   0
3   1   3   1   4   5   2
                1  -3  -3   3  -3
4   1   1   2   3   4   5
                0   0   0   0   0
```

With Fbar and ORIC the optimal (constrained) estimates are computed:

```
                    Estimated Coefficients
                    **********************

                     Dense Matrix (4 by 5)

        |      Group1     Group2     Group3     Group4     Group5
      ------------------------------------------------------------
 Model1 |   2.4634000  2.4634000  2.4634000  2.4634000  2.4634000
 Model2 |   2.3293333  1.3280000  3.2145000  2.2306667  3.2145000
 Model3 |   2.5963333  1.3280000  3.2000000  2.5963333  2.5963333
 Model4 |   2.3293333  1.3280000  3.2000000  2.2306667  3.2290000
```

The program

```
  Fbar Test for Null Hypothesis 1 and Unconstrained Hypothesis 4
    Hypothesis 1 versus 4:  Fbar=       30.27 p-value=  0.0000


                  ******************************
                  "Ordered Alternative" Fbar Test
                  ******************************

                  Hypothesis        Fbar     p-value

                    H0 vs  2  30.2643855  0.00000000
                    H0 vs  3  22.9115843  0.00000000


                     ************************
                     "Ordered Null" Fbar Test
                     ************************

                  Hypothesis        Fbar     p-value

                     2 vs Ha  0.00511179  0.98600000
                     3 vs Ha  7.35791301  0.05600000
```

```
          Fbar and Pvalues for Null and Alt Models
          ****************************************

                    Dense Matrix (4 by 5)

            |       Model1      Model2      Model3      Model4
           -------------------------------------------------------
H0_FbarProb |       .        0.0051118   7.3579130      .
    H0_Pval |       .        0.9860000   0.0560000      .
    Ha_Fbar |       .        30.264385   22.911584      .
    Ha_Pval |       .        0.0000000   0.0000000      .


            |  H0_vs_Hator
           ---------------
H0_FbarProb |       .
    H0_Pval |       .
    Ha_Fbar |    30.269497
    Ha_Pval |    0.0000000
```

After Fbar the pecified ORIC is computed. The model with the smallest ORIC value is the most preferred:

```
     ******************************************************
     The value of the Order-Restricted Information Criterion
             ORIC = -2 * log likelihood + 2 * penalty
     ******************************************************

            Model      LogLik     Penalty        ORIC

                1 -292.267801  2.00000000  588.535603
                2 -278.051118  3.17300000  562.448236
                3 -281.760877  3.15300000  569.827753
                4 -278.048474  6.00000000  568.096948


   The preferred hypothesis, according to the Order-Restricted
 Information Criterion of the hypotheses be compared is hypothesis
        number 2 with the following ordering(s) of means:

                        5     3     1     2     4
                        3     4     2     1     5
                 Corresponding Order of Restrictions:

                        1     1    -3    -3     0
                        1    -3    -3     0     0
```

After ORIC the pecified BMS is computed. Here is it necessary to specify the `delta` and `pv` options which may have a huge effect on the computer time needed for the bayes estimation.

```
*************************************************************************
  The resulting Bayes factor (BF) values (of the order-restricted
 hypotheses versus the unconstrained hypothesis) and the posterior
 model probabilities (PMP) of the order-restricted hypotheses with
              respect to the whole set of hypotheses:

                 Hypothesis          BF         PMP

                         1  0.02643567  5.797e-004
                         2  39.5893012  0.86820503
                         3  4.98328638  0.10928494
                         4  1.00000000  0.02193029


*************************************************************************
The preferred hypothesis, according to confirmatory Bayesian model
 selection of the hypotheses to be compared is hypothesis number 2
             with the following ordering(s) of means:

                       3     4     2     1     5
                       3     1     4     5     2
                Corresponding Order of Restrictions:

                       1    -3    -3     0     0
                       1    -3    -3     3    -3


*************************************************************************
             Specification of the Encompassing Prior:
For all means, the same normal prior with mean 2.28037 and variance
                          2.3046 is used.
   For the residual variance, a scaled inverse chi-square, with
     degrees of freedom 1 and scale parameter 2.50256 is used.
```

The following is the printed output of the return arguments:

```
< gof,lik,est,fpbar,BF > = mucomp(data,nres,ord,rest,optns);
print "GOF=",gof;
print "LIK=",lik;
print "EST=",est;
print "FPBAR=",fpbar;
print "BF=",BF;
```

GOF=

	g-of-fit
Error	0.000000
Mu_Total	2.46340
SS_Total	357.835
Time_Tot	64.0000
Time_Opt	18.0000
Iter_Tot	17439.0
unused	.
unused	.
unused	.
unused	.

LIK=

	LogLik	ResiSS	PenTrm	ORIC
Model1	-292.27	432.53	2.0000	588.54
Model2	-278.05	357.85	3.1730	562.45
Model3	-281.76	375.99	3.1530	569.83
Model4	-278.05	357.83	6.0000	568.10

EST=

	Group1	Group2	Group3	Group4	Group5
Model1	2.4634	2.4634	2.4634	2.4634	2.4634
Model2	2.3293	1.3280	3.2145	2.2307	3.2145
Model3	2.5963	1.3280	3.2000	2.5963	2.5963
Model4	2.3293	1.3280	3.2000	2.2307	3.2290

FPBAR=

	Model1	Model2	Model3
H0_FbarProb	.	0.0051	7.3579
H0_Pval	.	0.98600	0.0560
Ha_Fbar	.	30.264	22.912
Ha_Pval	.	0.00000	0.00000

	Model4	H0_vs_Hator
H0_FbarProb	.	.
H0_Pval	.	.
Ha_Fbar	.	30.269

```
     Ha_Pval |          .           0.00000

BF=
         |   BayesFactor   PostModProb
-------------------------------------
  Model1 |        0.0264        0.0006
  Model2 |        39.589       0.86821
  Model3 |        4.9833       0.10928
  Model4 |       1.00000        0.0219
```

2. Example by Lucas (2003): Tensor Input:
   The following specification is equivalent with that of the Kuiper, Klugkist, & Hojtink (2009) program: The first matrix specifies the order of variables in the restrictions (second matrix):

```
ord = [ 1 2 3 4 5 ,
        5 3 1 2 4 ,
        3 4 2 1 5 ,
        3 1 4 5 2 ,
        1 2 3 4 5 ];
print "Ordering of means in restriction",ord;
```

   The second matrix specifies the restrictions of the four models:

```
rest = [ 1  1  1  1  1 ,
         1  1 -3 -3  0 ,
         1 -3 -3  0  0 ,
         1 -3 -3  3 -3 ,
         0  0  0  0  0 ];
print "(Order) Restrictions",rest;
```

   The meaning of the integers in the restriction matrix is the following:

   - -1: defines the ¡ relationship;
   - -3: defines the ¿ relationship;
   - 0: defines a singleton (unused by the restriction);
   - positive integers: specify the group number of equality constrained clusters of variables.

   The two matrices are merged in the tensor `tmodl`:

```
int tmodl[2,5,5];
tmodl[1,,] = ord; tmodl[2,,] = rest;
print "Tensor=",tmodl;
```

The results are identical to those obtained with the string specification of the model.

5.9 Function `shapwilk`

prob = shapwilk(y<,optn>)

< prob,stat > = shapwilk(y<,optn>)

Purpose: Implements the Shapiro-Wilks and the Shapiro-Francia tests for normally distributed data.

Input: y The first input object y must be either a n vector or a $n \times nc$ matrix. If y is a vector the scalar value of the probability of the CDF is returned, if y is a matrix then a vector of nc probabilities is returned by using each column of y separately.

optn this is a vector of options which should be initialized with missing values for defaults:

1. (int) amount of printed output (default is no output);
2. (real) probability for test (default is 0.05), must be in $[0, 1]$;
3. (int) version: =0: Shapiro-Wilks test (default); =1: Shapiro-Francia test.

Output: prob The first result is either a scalar or an object of the test probabilities.

stat The `stat` result has the same size as `prob` and returns the test statistics (only for `optn[3]=0`).

Restrictions: 1. Missing value are returned if the input y contains missing data.

2.

Relationships: berkow(), jarbera(), ksprob(), kstest()

Examples: 1. Shapiro-Wilks Test: $N = 1000$

```
   options NOECHO;
#include "..\\tdata\\matlb.dat"
   options ECHO;


print "Shapiro-Wilks test for Normality";
optn = [   2 ,  /* print */
         .05 ,  /* alpha */
           0 ]; /* vers */
< p1,s1 > = shapwilk(uf1000,optn);
print "SHAPIRO-Wilks P:",p1;
print "SHAPIRO-Wilks S:",s1;
```

```
           Shapiro-Wilks Test for Normal Distribution
        Hypothesis H0: Data are normally distributed versus
         Hypothesis H1: Data are not normally distributed
    H0 rejected: Probability=5.41037e-017 Statistics=8.29541


SHAPIRO-Wilks P: 5.410e-017

SHAPIRO-Wilks S: 8.2954


 print "Shapiro-Francia test for Normality";
 optn = [   2 ,  /* print */
          .05 ,  /* alpha */
            1 ]; /* vers */
 < p1,s1 > = shapwilk(uf1000,optn);
 print "SHAPIRO-Francia P:",p1;
 print "SHAPIRO-Francia S:",s1;


          Shapiro-Francia Test for Normal Distribution
        Hypothesis H0: Data are normally distributed versus
         Hypothesis H1: Data are not normally distributed
     H0 rejected: Probability=0.000253907 Statistics=3.4766


SHAPIRO-Francia P: 0.0002539

SHAPIRO-Francia S: 3.4766
```