Matlab® hints to Stationary stochastic processes for scientists and engineers

October 22, 2013

1 General advice

All simulation, calculation, and plotting in the textbook was performed by Matlab commands. Useful m-files and data are available in:

Matlab basic: the basic Matlab installation without any extra toolboxes.

Special routines: a package `ssproutines` of selected useful routines that can be downloaded from the book page at Lund university.

Data: A collection of data `sspdata` that can be downloaded from the book page at Lund university.

Matlab toolboxes: `filterdesign`, `ident`, `signal` contain some routines used in the Computer exercises.

Wafo: a large package WAFO of routines and data, designed for analysis, simulation, and statistical analysis of stationary processes. A zip-file `wafo25.7z` can be downloaded from the webpage

`code.google.com/p/wafo/`

It is accompanied by a tutorial (159 pages)

`wafotutor25_rev1.pdf`

that can be downloaded from the same site.

To use these hints you should download the Special routines and Data package. Once you have familiarized yourself with the theory and practised with the routines you should download the WAFO package and start exporing more. Note, however, that some special computation routines are compiled only for Windows. There is also a Python version available on code.google.

2 Special routines

m-files in `ssproutines`

simsum: simulates a simple sum of random cosines

armagui: graphical interface for experiment with ARMA-processes

initarma: help file to `armagui`

kovarians: help file to `armagui`

spekgui: graphical interface for spectral and covariance estimation

initspek: help file to `spekgui`

welch: help file to `spekgui`

dat2cov: estimates a covariance function from data (WAFO)

dat2spec: estimates a spectral density function from data (WAFO)

createspec: routine to create spectral density structures (WAFO)

spec2sdat: routine to simulate stationary processes from spectrum structure (WAFO)

freqtype: identifies the type (frequency/angular) of a spectrum structure (WAFO)

jonswap: gives a spectrum structure of the JONSWAP ocean wave type (WAFO)

oscspec: gives a spectrum structure for the harmonic oscillator (WAFO)

waveplot: plots one or more processes in different colors (WAFO)

covplot: plots a covariance function (WAFO)

specplot: plots a spectral density (WAFO)

---

armagui

```
Call:
armagui

opens a window with point-and-click controls ("control")
for modeling ARMA(p,q)-processes.

Use left/right mouse button to add/remove real or complex poles
and zeros in the unit circle. Move them around by the left button.
Every change will generate a new realization, covariance function
and spectrum. Control "Simulate" gives a new realization.

If the control "Request stability" is alight poles and zeros cannot
be moved outside the unit circle.
```

```
Import A- and C-polynomials by control "Import". For example,
define, in the Matlab window,

>>  model.A=[1 -0.5];
>>  model.C=[1 0.4 0.7];

and import  model  to get an ARMA(1,2)-process.

Export covariance function, spectral density, realization, etc,
to the Matlab window by the control "Export".
Generates a structure (default name  data ) with fields
containing poles, zeros, polynomials, etc.

Seasonal ARIMA-processes can be modeled by setting "Season"
to a positive integer.
```

---

spekgui

```
Call:
spekgui

Opens a window with point-and-click controls ("control")
for spectral and covariance estimation

Import data with the "Import" control from a structure defined
in the Matlab window. For example,

>> somedata.t=linspace(0,50,1001);
>> somedata.x=sin(2*pi*somedata.t)+randn(1,1001)*0.5;

Methods for spectrum estimation are:
   Periodogram; Section 9.2.
   Averaging; Section 9.5.2.
       Parameter = K gives average over  K  subseries
   Welch (with time Hanning window);  Section 9.5.2, Example 9.4.
       Parameter = K gives average over K Hamming windowed subseries
   Time window = Data windowing; Section 9.4.
       Parameter = n is the lag parameter in a  "Hanning window".
   Frequency window = Smoothing over frequencies; Section 9.5.1.
       Parameter = L is the half width of the frequency window.
```

```
Covariance function can be estimated from data or from
the spectrum estimate.

The estimates can be exported to the Matlab window by the
"Export" control.

Exampel:
Construct an ARMA-model by  armagui  and export it  as "data".
Import "data" to  spekgui  and estimate spectrum.
Export the estimates to "estdata"  and compare by

>> plot(data.f,data.R,'r',estdata.f,estdata.R,'b')
```

---

dat2cov

```
CALL:  R = dat2cov(x,L,plotflag,dT,flag)

       R = a structure containing:
             R = ACF vector length L+1
             t = time lags  length L+1
             stdev = estimated large lag standard deviation of the
                     estimate assuming x is a Gaussian process:
                     if R(k)=0 for lags k>q then an approximation
                     of the variance for large samples
                     due to Bartlett is given by
      var(R(k))=1/N*(R(0)^2+2*R(1)^2+2*R(2)^2+ ..+2*R(q)^2)
                     for  k>q and where  N=length(x). Special case
                     white noise where it equals R(0)^2/N for k>0
             h = water depth (default inf)
             tr = [], transformation
             type = 'none'
             norm = 0 indicating that R is not normalized
         x  = a column data vector or
              two column data matrix with sampled times and values.
        L  = the maximum time-lag for which the ACF is estimated.
             (Default L=n-1)
  plotflag = 1 then the ACF is plotted vs lag
             2 then the ACF is plotted vs lag
             3 then the ACF is plotted vs lag and vs lag
        dT = time-step between data points
                 (default xn(2,1)-xn(1,1) or 1 Hz).
     flag = 'biased': scales the raw cross-correlation by 1/n.
```

```
                (default)
          'unbiased':scales the raw correlation by 1/(n-abs(k)),
                        where k is the index into the result.

  Note:  - flag may be given anywhere after x.
         - x may contain NaN's  (i.e. missing values).
```

---

dat2spec

```
CALL:
S = dat2spec(x,L,g,plotflag,p,method,dflag,ftype)

          S = A structure containing:
              S    = spectral density
              w    = angular frequency
              tr   = transformation g
              h    = water depth (default inf)
              type = 'freq'
              note = Memorandum string
              date = Date and time of creation
              L    = maximum lag size of the window function.
              CI   = lower and upper confidence constant
              p    = confidence level. (Default 0.95).
              Bw   = Bandwidth of the smoothing window used
                     in the estimated spectrum. (rad/sec or Hz)
          x = m column data matrix with sampled times in the
              first column and values the next columns.
          L = maximum lag size of the window function.
              If no value is given the lag size is set to
              be the lag where the auto correlation is less than
              2 standard deviations. (maximum 300)
          g = the transformation assuming that x is a sample of a
              transformed Gaussian process. If g is empty then
              x  is a sample of a Gaussian process (Default)
   plotflag = 1 plots the spectrum, S,
              2 plot 10log10(S) and
              3 plots both the above plots
   Method   = 'cov' Frequency smoothing using a parzen window
                  on the estimated autocovariance function. (Default)
              'psd' Welch's averaged periodogram method with
                  non-overlapping batches
              'psdo'  Welch's averaged periodogram method with
```

```
                overlapping batches
              'pmem'  Maximum Entropy Method (psd using the
                   Yule-Walker AR method)
              'pburg' Burg's method
    dflag    = specifies a detrending performed on the signal
               before estimation. 'mean','linear'
               or 'ma' (= moving average)  (Default 'mean')
    ftype    = frequency type, 'w' or 'f'  (Default 'w')

If Method == 'cov','psd' then:
   As L decreases the estimate becomes smoother and Bw increases.
    If we want to resolve peaks in S which is Bf (Hz or rad/sec)
    apart then Bw < Bf.
If Method == 'pmem','pburg' then:
   L denotes the order of the AR (AutoRegressive) model.
```

---

createspec

```
CALL:
S          = createspec(stype,freqtype)

stype  = 'freq'    Frequency spectrum (default)
         'dir'     Directional spectrum
         'k1D'     Wave number spectrum 1D
         'k2D'     Wave number spectrum 2D
         'encdir'  Encountered directional spectrum
         'enc'     Encountered frequency spectrum

freqtype = 'w' angular frequency (rad/sec) (default)
           'f' frequency         (Hz)

Example: Create a structure with proper field names
for directional spectrum
S=createspec('dir')
```

---

jonswap

```
CALL:
S = jonswap(w,sdata,plotflag);
```

```
  S     = a struct containing the spectral density.
  w     = angular frequency        (default linspace(0,wc,257))
  wc    = angular cutoff frequency (default 33/Tp)
  sdata = [Hm0 Tp gamma sa sb A], where
          Hm0   = significant wave height (default 7 (m))
          Tp    = peak period (default 11 (sec))
          gamma = peakedness factor determines the concentraton
                  of the spectrum on the peak frequency,
                  1 <= gamma <= 7.
          sa,sb = spectral width parameters (default 0.07 0.09)
          Ag    = normalization factor used when gamma>1
                  Ag<0  : Ag calculated by integration so that
                          int S dw = Hm0^2/16 (default)
                  Ag==0 : Ag = (1+1.00*log(gamma)^1.16)/gamma
                  Ag>0  : Ag = Ag
      plotflag = 0, do not plot the spectrum (default).
                 1, plot the spectrum.

JONSWAP returns a spectrum object defined as

        S(w) = A * Gf * G0 * wn^(-N)*exp(-N/(M*wn^M))
   where
        G0  = Normalizing factor related to Bretschneider form
        A   = Ag * (Hm0/4)^2 / wp     (Normalization factor)
        Gf  = j^exp(-.5*((wn-1)/s)^2) (Peak enhancement factor)
        wn  = w/wp
        wp  = angular peak frequency
        s   = sa      for wn <= 1
              sb      for 1  <  wn
        j   = gamma,  (j=1, => Bretschneider spectrum)
```

---

oscspec

```
CALL:
S  = oscspec(sdata,z,b,s);

Output:
S   =  the spectral density (structure array)

Input:
sdata  = the data vector [wl wu n], where
```

```
wl  = lower truncation frequency  (default 4/257)
wu  = upper truncation frequency  (default 4)
n   = number of evaluation points (default 257)
z,b,s  = parameters in the equation for the oscillator.
                 (default z=0.1, b=1, s=1)

Let   W   be white noise. Then the oscillator  X  is defined by

X''(t) + 2bz X'(t) + b^2 X(t) = s W(t)

The angular peak frequency is given by wp = b/sqrt(1-4*z^2).
Important parameter values to a normalized linear oscillator
with  Var(X(t))=Var(X'(t))=1:

0<z<1, b=1, s = 2*sqrt(z) :

Example:
data = [0.01 4 275];
S    = oscspec(data,[],2.5);  % Peak frequency at w=2.5
```

---

simsum

```
CALL:
SIMSUM simulates  N  realizations of

X(t) = A0 + sum Ak cos(2 pi fk t + phik)

A0 is normal with variance sigma2(0)
Ak is Rayleigh with parameter sigma2(k)

rayamp  = simsum(f,sigma2_,N,T,plotid)

Output:
rayamp  = Gaussian process X(t)

Input:
f       = 1 x K vector with frequencies in increasing order
sigma2_ = 1 x K vector with energy per frequency
N       = number of realizations
t       = time

Exampel:
```

```
f = [5 10];
sigma2_ = [2 2];
N = 1000;
dt = 1/(2*max(f)+1);
t = 0:dt:20;
rayamp=simsum(f,sigma2_,N,t);

For use in  spekgui
data.x = rayamp(:,5);
data.dt = dt;
```

---

spec2sdat

```
CALL:
[xs, xsder] = spec2sdat(S,[np cases],dt,iseed,method);

         xs    = a cases+1 column matrix  ( t,X1(t) X2(t) ...).
         xsder = a cases+1 column matrix  ( t,X1'(t) X2'(t) ...).
         S     = a spectral density structure
         np    = giving np load points.  (default length(S)-1=n-1).
                 If np>n-1 it is assummed that R(k)=0 for all k>n-1
         cases = number of cases (replicates), (default=1)
            dt = step in grid (default set by the Nyquist freq)
         iseed = starting seed for the random number generator
                 (default none is set)
        method = 'exact'  : simulation using cov2sdat
                 'random' : random phase and amplitude (default)

SPEC2SDAT performs a fast and exact simulation of stationary
zero mean Gaussian process through circulant embedding of
the covariance matrix or by summation of sinus functions with
random amplitudes and random phase angle.

If the spectrum has a non-empty field .tr, then the transformation
is applied to the simulated data, the result is a simulation of
a transformed Gaussian process.

NB! The method 'exact' simulation may give high frequency ripple when
used with a small dt. In this case the method 'random' works better.

Example to generate two time series:
np =100; dt = .2;
```

```
[x1 x2] = spec2sdat(jonswap,np,dt);
waveplot(x1, 'r', x2, 'g',1,1)
```

Plotting

The routine waveplot plots one or more processes with time in first column and data in next columns.

The routines covplot and specplot plot covariance functions and spectral densities; see help text.

3 Data

Data available in sspdata

Matlab mat-files:

afric: wave data in the Atlantic ocean

cello: Cello playing "a"

covProc: 2000 simulated data from a stationary process

data1: 100 data from white noise with mean m and standard deviation σ, (to be estimated)

eegdata12: file with three sets of eeg-measurements from one person; sampling frequency 5Hz. One set is taken with stimulus with frequency 12Hz.

eegdatax: file with three sets of eeg-measurements, one with stimulus of unknown frequency.

initarma: data for use in armagui

initspek: data for use in spekgui

lowpass: coefficients of a Butterworth filter for import in armagui

pulsedata: EKG and fingertip pulse data

trombone: Trombone playing "a"

yearMeanSpots: yearly sunspot index for years 1749 to 1984

wav files:

cello: recorded wav-file for cello data

trombone: recorded wav-file for trombone data