NAG Toolbox

nag_rand_times_smooth_exp (g05pm)

1 Purpose

nag_rand_times_smooth_exp (g05pm) simulates from an exponential smoothing model, where the model uses either single exponential, double exponential or a Holt–Winters method.

2 Syntax

```
[r, state, x, ifail] = nag_rand_times_smooth_exp(mode, n, itype, p, param, init, var, r, state, e, 'en', en)
[r, state, x, ifail] = g05pm(mode, n, itype, p, param, init, var, r, state, e, 'en', en)
```

3 Description

nag_rand_times_smooth_exp (g05pm) returns $\{x_t : t = 1, 2, \ldots, n\}$, a realization of a time series from an exponential smoothing model defined by one of five smoothing functions:

Single Exponential Smoothing

$$\begin{aligned} x_t &= m_{t-1} + \epsilon_t \\ m_t &= \alpha x_t + (1-\alpha) m_{t-1} \end{aligned}$$

Brown Double Exponential Smoothing

$$\begin{aligned} x_t &= m_{t-1} + \tfrac{r_{t-1}}{\alpha} + \epsilon_t \\ m_t &= \alpha x_t + (1-\alpha) m_{t-1} \\ r_t &= \alpha(m_t - m_{t-1}) + (1-\alpha) r_{t-1} \end{aligned}$$

Linear Holt Exponential Smoothing

$$\begin{aligned} x_t &= m_{t-1} + \phi r_{t-1} + \epsilon_t \\ m_t &= \alpha x_t + (1-\alpha)(m_{t-1} + \phi r_{t-1}) \\ r_t &= \gamma(m_t - m_{t-1}) + (1-\gamma)\phi r_{t-1} \end{aligned}$$

Additive Holt–Winters Smoothing

$$\begin{aligned} x_t &= m_{t-1} + \phi r_{t-1} + s_{t-1-p} + \epsilon_t \\ m_t &= \alpha\big(x_t - s_{t-p}\big) + (1-\alpha)(m_{t-1} + \phi r_{t-1}) \\ r_t &= \gamma(m_t - m_{t-1}) + (1-\gamma)\phi r_{t-1} \\ s_t &= \beta(x_t - m_t) + (1-\beta) s_{t-p} \end{aligned}$$

Multiplicative Holt–Winters Smoothing

$$\begin{aligned} x_t &= (m_{t-1} + \phi r_{t-1}) \times s_{t-1-p} + \epsilon_t \\ m_t &= \alpha x_t / s_{t-p} + (1-\alpha)(m_{t-1} + \phi r_{t-1}) \\ r_t &= \gamma(m_t - m_{t-1}) + (1-\gamma)\phi r_{t-1} \\ s_t &= \beta x_t / m_t + (1-\beta) s_{t-p} \end{aligned}$$

where m_t is the mean, r_t is the trend and s_t is the seasonal component at time t with p being the seasonal order. The errors, ϵ_t are either drawn from a normal distribution with mean zero and variance σ^2 or randomly sampled, with replacement, from a user-supplied vector.

4 References

Chatfield C (1980) *The Analysis of Time Series* Chapman and Hall

5 Parameters

5.1 Compulsory Input Parameters

1: **mode** – INTEGER

Indicates if nag_rand_times_smooth_exp (g05pm) is continuing from a previous call or, if not, how the initial values are computed.

mode $= 0$
Values for m_0, r_0 and s_{-j}, for $j = 0, 1, \ldots, p-1$, are supplied in **init**.

mode $= 1$
nag_rand_times_smooth_exp (g05pm) continues from a previous call using values that are supplied in **r**. **r** is not updated.

mode $= 2$
nag_rand_times_smooth_exp (g05pm) continues from a previous call using values that are supplied in **r**. **r** is updated.

Constraint: **mode** $= 0$, 1 or 2.

2: **n** – INTEGER

The number of terms of the time series being generated.

Constraint: **n** ≥ 0.

3: **itype** – INTEGER

The smoothing function.

itype $= 1$
Single exponential.

itype $= 2$
Brown's double exponential.

itype $= 3$
Linear Holt.

itype $= 4$
Additive Holt–Winters.

itype $= 5$
Multiplicative Holt–Winters.

Constraint: **itype** $= 1$, 2, 3, 4 or 5.

4: **p** – INTEGER

If **itype** $= 4$ or 5, the seasonal order, p, otherwise **p** is not referenced.

Constraint: if **itype** $= 4$ or 5, **p** > 1.

5: **param**(:) – REAL (KIND=nag_wp) array

The dimension of the array **param** must be at least 1 if **itype** $= 1$ or 2, 3 if **itype** $= 3$ and at least 4 if **itype** $= 4$ or 5

The smoothing parameters.

If **itype** $= 1$ or 2, **param**$(1) = \alpha$ and any remaining elements of **param** are not referenced.

If **itype** $= 3$, **param**$(1) = \alpha$, **param**$(2) = \gamma$, **param**$(3) = \phi$ and any remaining elements of **param** are not referenced.

If **itype** $= 4$ or 5, **param**$(1) = \alpha$, **param**$(2) = \gamma$, **param**$(3) = \beta$ and **param**$(4) = \phi$ and any remaining elements of **param** are not referenced.

Constraints:

if **itype** $= 1$, $0.0 \leq \alpha \leq 1.0$;
if **itype** $= 2$, $0.0 < \alpha \leq 1.0$;
if **itype** $= 3$, $0.0 \leq \alpha \leq 1.0$ and $0.0 \leq \gamma \leq 1.0$ and $\phi \geq 0.0$;
if **itype** $= 4$ or 5, $0.0 \leq \alpha \leq 1.0$ and $0.0 \leq \gamma \leq 1.0$ and $0.0 \leq \beta \leq 1.0$ and $\phi \geq 0.0$.

6: **init**(:) – REAL (KIND=nag_wp) array

The dimension of the array **init** must be at least 1 if **itype** $= 1$, 2 if **itype** $= 2$ or 3 and at least $2 + \mathbf{p}$ if **itype** $= 4$ or 5

If **mode** $= 0$, the initial values for m_0, r_0 and s_{-j}, for $j = 0, 1, \ldots, p-1$, used to initialize the smoothing.

If **itype** $= 1$, **init**$(1) = m_0$ and any remaining elements of **init** are not referenced.

If **itype** $= 2$ or 3, **init**$(1) = m_0$ and **init**$(2) = r_0$ and any remaining elements of **init** are not referenced.

If **itype** $= 4$ or 5, **init**$(1) = m_0$, **init**$(2) = r_0$ and **init**(3) to **init**$(2 + p)$ hold the values for s_{-j}, for $j = 0, 1, \ldots, p-1$. Any remaining elements of **init** are not referenced.

7: **var** – REAL (KIND=nag_wp)

The variance, σ^2 of the Normal distribution used to generate the errors ϵ_i. If **var** ≤ 0.0 then Normally distributed errors are not used.

8: **r**(:) – REAL (KIND=nag_wp) array

The dimension of the array **r** must be at least 13 if **itype** $= 1$, 2 or 3 and at least $13 + \mathbf{p}$ if **itype** $= 4$ or 5

If **mode** $= 1$ or 2, **r** must contain the values as returned by a previous call to nag_rand_times_smooth_exp (g05pm), **r** need not be set otherwise.

Constraint: if **mode** $= 1$ or 2, **r** must have been initialized by at least one call to nag_rand_times_smooth_exp (g05pm) or nag_tsa_uni_smooth_exp (g13am) with **mode** $\neq 1$, and **r** must not have been changed since that call.

9: **state**(:) – INTEGER array

Note: the actual argument supplied **must** be the array **state** supplied to the initialization routines nag_rand_init_repeat (g05kf) or nag_rand_init_nonrepeat (g05kg).

Contains information on the selected base generator and its current state.

10: **e**(**en**) – REAL (KIND=nag_wp) array

If **en** > 0 and **var** ≤ 0.0, a vector from which the errors, ϵ_t are randomly drawn, with replacement.

If **en** ≤ 0, **e** is not referenced.

5.2 Optional Input Parameters

1: **en** – INTEGER

Default: the dimension of the array **e**.

If **en** > 0, then the length of the vector **e**.

If both **var** ≤ 0.0 and **en** ≤ 0 then $\epsilon_t = 0.0$, for $t = 1, 2, \ldots, n$.

5.3 Output Parameters

1: **r**(:) – REAL (KIND=nag_wp) array

The dimension of the array **r** will be 13 if **itype** = 1, 2 or 3 and at least 13 + **p** if **itype** = 4 or 5

If **mode** = 1, **r** is unchanged. Otherwise, **r** contains the information on the current state of smoothing.

2: **state**(:) – INTEGER array

Contains updated information on the state of the generator.

3: **x**(**n**) – REAL (KIND=nag_wp) array

The generated time series, x_t, for $t = 1, 2, \ldots, n$.

4: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

Constraint: **mode** = 0, 1 or 2.

ifail = 2

Constraint: **n** ≥ 0.

ifail = 3

Constraint: **itype** = 1, 2, 3, 4 or 5.

ifail = 4

Constraint: if **itype** = 4 or 5, **p** ≥ 2.

ifail = 5

Constraint: 0 ≤ **param**(i) ≤ 1.

Constraint: if **itype** = 2, 0 < **param**(i) ≤ 1.

Constraint: **param**(i) ≥ 0.

ifail = 8

On entry, some of the elements of the array **r** have been corrupted or have not been initialized.

ifail = 9

On entry, **state** vector has been corrupted or not initialized.

ifail = 12

Model unsuitable for multiplicative Holt–Winter, try a different set of parameters.

ifail = −99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example reads 11 observations from a time series relating to the rate of the earth's rotation about its polar axis and fits an exponential smoothing model using nag_tsa_uni_smooth_exp (g13am).

nag_rand_times_smooth_exp (g05pm) is then called multiple times to obtain simulated forecast confidence intervals.

9.1 Program Text

```
function g05pm_example

fprintf('g05pm example results\n\n');

% Initialize the generator to a repeatable sequence
seed  = [nag_int(1762543)];
genid = nag_int(1);
subid =  nag_int(1);
[state, ifail] = g05kf( ...
                        genid, subid, seed);

mode  = nag_int(2);
n     = 11;
k     = nag_int(11);
nf    = nag_int(20);
nsim  = 100;
alpha = 0.05;
y     = [180; 135; 213; 181; 148; 204; 228; 225; 198; 200; 187];
itype = nag_int(3);
inits = [0; 0];
p     = nag_int(0);
r     = zeros(p+13,1);
param = [0.01; 1; 1];
e     = [];
gsim = zeros(nsim, nf);
bsim = zeros(nsim, nf);

% Fit a smoothing model (parameter r in g05pm and state in g13am
% are in the same format)
[inits, fv, fse, yhat, res, dv, ad, r, ifail] =  ...
  g13am( ...
         mode, itype, p, param, y, k, inits, nf, r);

% Simulate forecast values from the model, and no update of r
smode = nag_int(2);
var   = dv*dv;
% Simulate nsim forecasts
for i = 1:nsim
  % Simulations assuming gaussian errors
  [r, state, gsim(i, :), ifail] = ...
```

```
    g05pm( ...
           smode, nf, itype, p, param, inits, var, r, state, e);
  % Bootstrapping errors
  [r, state, bsim(i, :), ifail] = ...
    g05pm( ...
           smode, nf, itype, p, param, inits, 0, r, state, res);
end

% Calculate CI based on the quantiles for each simulated forecast
q = [alpha/2; 1-alpha/2];
glim = zeros(2, nf);
blim = zeros(2, nf);
for i = 1:nf
  [glim(:, i), ifail] = g01am(gsim(:,i), q);
  [blim(:, i), ifail] = g01am(bsim(:,i), q);
end

% Display the forecast values and associated prediction intervals
fprintf('\n Initial values used:\n');
for i = 1:numel(inits);
  fprintf(' %d %12.3f\n', i, inits(i));
end
fprintf('\nMean Deviation     = %12.4e\n', dv);
fprintf('Absolute Deviation = %12.4e\n\n', ad);
fprintf('         Observed      1-Step\n');
fprintf(' Period   Values      Forecast      Residual\n');
for i=1:n
  fprintf('%4d %12.3f %12.3f %12.3f\n', i, y(i,1), yhat(i,1), res(i,1));
end
fprintf('\n%52s%20s\n','Simulated CI','Simulated CI');
fprintf('%14s%17s%22s%21s\n', 'Obs.  Forecast', 'Estimated CI', ...
        '(Gaussian Errors)', '(Bootstrap Errors)');

[z, ifail] = g01fa(q(2));

for i = 1:5
  tmp = z*fse(i);
  fprintf('%3d', n+i);
  fprintf('%10.3f', fv(i), fv(i)-tmp, fv(i)+tmp, glim(1:2, i), blim(1:2, i));
  fprintf('\n');
end
fprintf('\n %5.1f%% CIs were produced\n', 100*(1-alpha));

fig1 = figure;
hold on;
plot([1:n], y, 'x-r');
plot([1:n], yhat,'-g');
plot([n+1:n+20], fv,'-b');
plot([n+1:n+20], fv-z*fse,'-c');
plot([n+1:n+20], fv+z*fse,'-c');
plot([n+1:n+20], glim(1,:),'-m');
plot([n+1:n+20], glim(2,:),'-m');
plot([n+1:n+20], blim(1,:),'-k');
plot([n+1:n+20], blim(2,:),'-k');
hold off
legend('observed', 'smoothed', 'forecast', 'estimated CI', ...
       'simulated CI', 'bootsrtapped CI', 'Location', 'SouthEast');
xlabel('Time');
ylabel('Data');
title({'Exponential smoothing', ...
       '95%% confidence intervals (CIs) are shown'});
```

9.2 Program Results

```
     g05pm example results

 Initial values used:
 1      168.018
 2        3.800
```

```
Mean Deviation     =   2.5473e+01
Absolute Deviation =   2.1233e+01

         Observed      1-Step
 Period   Values      Forecast      Residual
   1      180.000      171.818        8.182
   2      135.000      175.782      -40.782
   3      213.000      178.848       34.152
   4      181.000      183.005       -2.005
   5      148.000      186.780      -38.780
   6      204.000      189.800       14.200
   7      228.000      193.492       34.508
   8      225.000      197.732       27.268
   9      198.000      202.172       -4.172
  10      200.000      206.256       -6.256
  11      187.000      210.256      -23.256

                                        Simulated CI        Simulated CI
Obs.  Forecast     Estimated CI     (Gaussian Errors)   (Bootstrap Errors)
 12   213.854   163.928   263.781   157.822   261.312   173.073   248.363
 13   217.685   167.748   267.622   171.006   258.662   176.994   252.738
 14   221.516   171.556   271.475   183.687   262.402   179.870   256.473
 15   225.346   175.347   275.345   176.347   275.832   183.923   261.128
 16   229.177   179.115   279.238   179.421   280.585   187.008   266.360

  95.0% CIs were produced
```