NAG Library Routine Document

G05PMF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G05PMF simulates from an exponential smoothing model, where the model uses either single exponential, double exponential or a Holt–Winters method.

2 Specification

```
SUBROUTINE G05PMF (MODE, N, ITYPE, P, PARAM, INIT, VAR, R, STATE, E, EN,    &
                   X, IFAIL)

INTEGER             MODE, N, ITYPE, P, STATE(*), EN, IFAIL
REAL (KIND=nag_wp) PARAM(*), INIT(*), VAR, R(*), E(EN), X(N)
```

3 Description

G05PMF 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

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

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 Arguments

1: MODE – INTEGER *Input*

On entry: indicates if G05PMF 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 $= 1$
G05PMF continues from a previous call using values that are supplied in R. R is not updated.

MODE $= 2$
G05PMF continues from a previous call using values that are supplied in R. R is updated.

Constraint: MODE $= 0$, 1 or 2.

2: N – INTEGER *Input*

On entry: the number of terms of the time series being generated.

Constraint: N ≥ 0.

3: ITYPE – INTEGER *Input*

On entry: 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 *Input*

On entry: 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 *Input*

Note: 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.

On entry: 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 *Input*

Note: the dimension of the array INIT must be at least 1 if ITYPE $= 1$, 2 if ITYPE $= 2$ or 3 and at least $2 +$ P if ITYPE $= 4$ or 5.

On entry: 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) *Input*

On entry: 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 *Input/Output*

Note: the dimension of the array R must be at least 13 if ITYPE $= 1$, 2 or 3 and at least $13 +$ P if ITYPE $= 4$ or 5.

On entry: if MODE $= 1$ or 2, R must contain the values as returned by a previous call to G05PMF, R need not be set otherwise.

On exit: if MODE $= 1$, R is unchanged. Otherwise, R contains the information on the current state of smoothing.

Constraint: if MODE $= 1$ or 2, R must have been initialized by at least one call to G05PMF or G13AMF with MODE $\neq 1$, and R must not have been changed since that call.

9: STATE$(*)$ – INTEGER array *Communication Array*

Note: the actual argument supplied **must** be the array STATE supplied to the initialization routines G05KFF or G05KGF.

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

10: E(EN) – REAL (KIND=nag_wp) array *Input*

On entry: 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.

11: EN – INTEGER *Input*

On entry: if EN > 0, then the length of the vector E.

If both VAR ≤ 0.0 and EN ≤ 0 then $\epsilon_t = 0.0$, for $t = 1, 2, \ldots, n$.

12: X(N) – REAL (KIND=nag_wp) array *Output*

On exit: the generated time series, x_t, for $t = 1, 2, \ldots, n$.

13: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, MODE $= \langle value \rangle$.
Constraint: MODE $= 0$, 1 or 2.

IFAIL $= 2$

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= 3$

On entry, ITYPE $= \langle value \rangle$.
Constraint: ITYPE $= 1$, 2, 3, 4 or 5.

IFAIL $= 4$

On entry, P $= \langle value \rangle$.
Constraint: if ITYPE $= 4$ or 5, P ≥ 2.

IFAIL $= 5$

On entry, PARAM($\langle value \rangle$) $= \langle value \rangle$.
Constraint: $0 \leq$ PARAM(i) ≤ 1.

On entry, PARAM($\langle value \rangle$) $= \langle value \rangle$.
Constraint: if ITYPE $= 2$, $0 <$ PARAM(i) ≤ 1.

On entry, PARAM($\langle value \rangle$) $= \langle value \rangle$.
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.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

G05PMF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 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 G13AMF.

G05PMF is then called multiple times to obtain simulated forecast confidence intervals.

10.1 Program Text

```
    Program g05pmfe

!     G05PMF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01amf, g01faf, g05kff, g05pmf, g13amf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: ad, alpha, dv, tmp, var, z
      Integer                          :: en, genid, i, ifail, itype, k, le,   &
                                          linit, lparam, lr, lstate, mode, n,  &
                                          nf, nsim, p, smode, subid
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: blim(:,:), bsim(:,:), e(:), fse(:),  &
                                          fv(:), glim(:,:), gsim(:,:),         &
                                          init(:), param(:), r(:), res(:),     &
```

```
                                        tsim1(:), tsim2(:), y(:), yhat(:)
      Real (Kind=nag_wp)               :: q(2)
      Integer                          :: seed(lseed)
      Integer, Allocatable             :: state(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05PMF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in the base generator information and seed
      Read (nin,*) genid, subid, seed(1)

!     Initial call to initializer to get size of STATE array
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Reallocate STATE
      Deallocate (state)
      Allocate (state(lstate))

!     Initialize the generator to a repeatable sequence
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Read in the initial arguments and check array sizes
      Read (nin,*) mode, itype, n, nf, nsim, alpha

      Select Case (itype)
      Case (1)
        lparam = 1
        p = 0
        linit = 1
      Case (2)
        lparam = 2
        p = 0
        linit = 2
      Case (3)
        lparam = 3
        p = 0
        linit = 2
      Case Default
        lparam = 4

!       Read in seasonal order
        Read (nin,*) p

        linit = p + 2
      End Select
      lr = 13 + p
!     Not using the E array for the bootstrap
      le = 0
      Allocate (param(lparam),init(linit),r(lr),e(le),fv(nf),fse(nf),yhat(n),  &
        res(n),blim(2,nf),glim(2,nf),tsim1(nf),tsim2(nf),gsim(nsim,nf),        &
        bsim(nsim,nf),y(n))

!     Read in series to be smoothed
      Read (nin,*) y(1:n)

!     Read in parameters
      Read (nin,*) param(1:lparam)

!     Read in the MODE dependent arguments (skipping headings)
      Select Case (mode)
      Case (0)
!       User supplied initial values
        Read (nin,*) init(1:linit)
      Case (1)
```

```
!       Continuing from a previously saved R
        Read (nin,*) r(1:(p+13))
      Case (2)
!       Initial values calculated from first K observations
        Read (nin,*) k
      End Select

!     Fit a smoothing model (parameter R in G05PMF and STATE in G13AMF
!     are in the same format)
      ifail = 0
      Call g13amf(mode,itype,p,param,n,y,k,init,nf,fv,fse,yhat,res,dv,ad,r,    &
        ifail)

!     Simulate forecast values from the model, and don't update R
      smode = 2
      var = dv*dv

!     Simulate NSIM forecasts
      Do i = 1, nsim
!       Not using E array for Gaussian errors
        en = 0

!       Simulations assuming Gaussian errors
        ifail = 0
        Call g05pmf(smode,nf,itype,p,param,init,var,r,state,e,en,tsim1,ifail)

!       For bootstrapping error, we are using RES from call to G13AMF as the
!       errors, and length of RES is N
        en = n

!       Bootstrapping errors
        ifail = 0
        Call g05pmf(smode,nf,itype,p,param,init,0.0E0_nag_wp,r,state,res,en,   &
          tsim2,ifail)

!       Copy and transpose the simulated values
        gsim(i,1:nf) = tsim1(1:nf)
        bsim(i,1:nf) = tsim2(1:nf)
      End Do

!     Calculate CI based on the quantiles for each simulated forecast
      q(1) = alpha/2.0E0_nag_wp
      q(2) = 1.0E0_nag_wp - q(1)
      Do i = 1, nf
        ifail = 0
        Call g01amf(nsim,gsim(1,i),2,q,glim(1,i),ifail)
        ifail = 0
        Call g01amf(nsim,bsim(1,i),2,q,blim(1,i),ifail)
      End Do

!     Display the forecast values and associated prediction intervals
      Write (nout,*) 'Initial values used:'
      Write (nout,99998) init(1:linit)
      Write (nout,*)
      Write (nout,99999) 'Mean Deviation     = ', dv
      Write (nout,99999) 'Absolute Deviation = ', ad
      Write (nout,*)
      Write (nout,*) '         Observed      1-Step'
      Write (nout,*) ' Period   Values      Forecast      Residual'
      Write (nout,*)
      Write (nout,99997)(i,y(i),yhat(i),res(i),i=1,n)
      Write (nout,*)
      Write (nout,*) '                                          ' //           &
        '      Simulated CI         Simulated CI'
      Write (nout,*) 'Obs.  Forecast      Estimated CI    ' //                 &
        '     (Gaussian Errors)    (Bootstrap Errors)'
      z = g01faf('L',q(2),ifail)
      Do i = 1, nf
        tmp = z*fse(i)
        Write (nout,99996) n + i, fv(i), fv(i) - tmp, fv(i) + tmp, glim(1,i),  &
          glim(2,i), blim(1,i), blim(2,i)
```

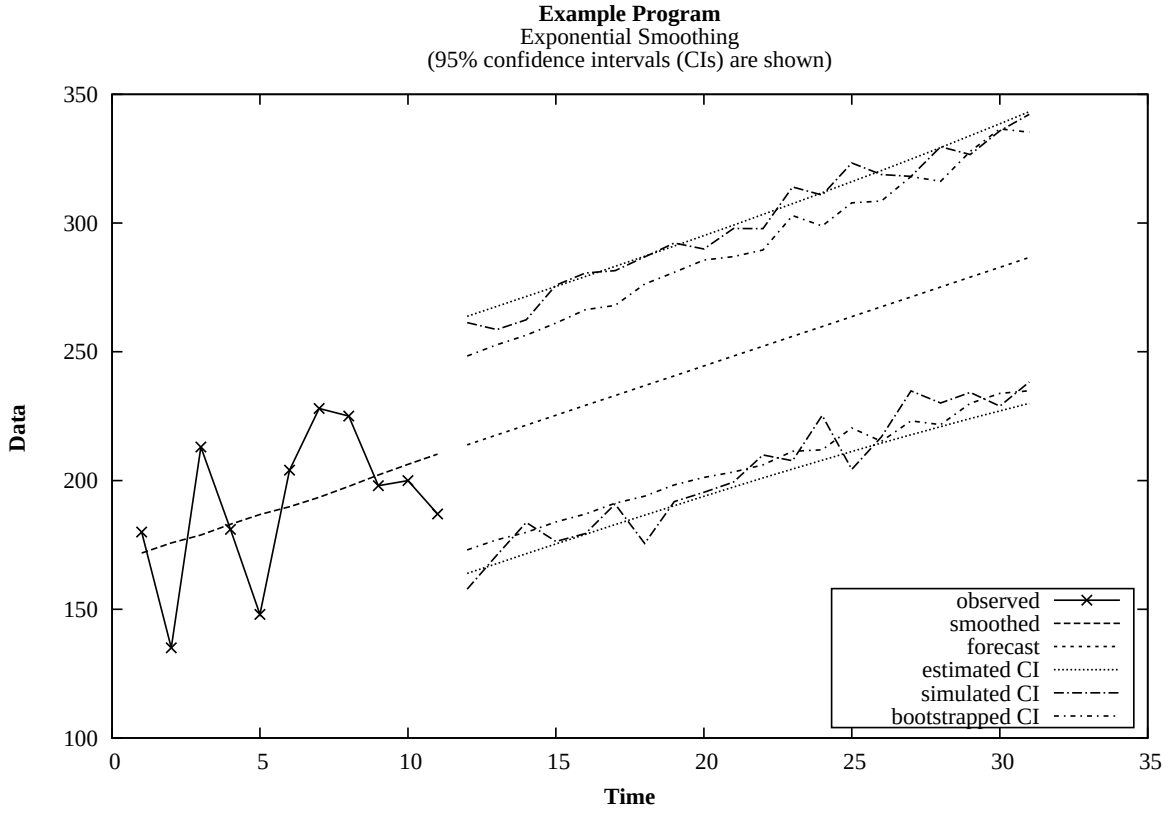
```
      End Do
      Write (nout,99995) 100.0E0_nag_wp*(1.0E0_nag_wp-alpha),                  &
        '% CIs were produced'

99999 Format (A,E12.4)
99998 Format (F12.3)
99997 Format (I4,1X,F12.3,1X,F12.3,1X,F12.3)
99996 Format (I3,7(1X,F10.3))
99995 Format (1X,F5.1,A)
    End Program g05pmfe
```

10.2 Program Data

```
G05PMF Example Program Data
1  1  1762543                                 :: GENID,SUBID,SEED(1)
2 3 11 5 100 0.05                             :: MODE,ITYPE,N,NF,NSIM,ALPHA
180 135 213 181 148 204 228 225 198 200 187 :: Y
0.01 1.0 1.0                                  :: PARAM
11                                            :: K
```

10.3 Program Results

```
 G05PMF Example Program Results

 Initial values used:
     168.018
       3.800

Mean Deviation     =   0.2547E+02
Absolute Deviation =   0.2123E+02

          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    161.431    258.001    173.073    248.363
 13    217.685    167.748    267.622    172.660    262.100    177.311    252.638
 14    221.516    171.556    271.475    169.259    263.107    179.344    256.921
 15    225.346    175.347    275.345    180.721    272.776    183.672    260.804
 16    229.177    179.115    279.238    184.790    263.591    186.398    264.173
  95.0% CIs were produced
```


Example Program
Exponential Smoothing
(95% confidence intervals (CIs) are shown)
Data
Time
observed
smoothed
forecast
estimated CI
simulated CI
bootstrapped CI
350
300
250
200
150
100
0
5
10
15
20
25
30
35