NAG Library Routine Document

G13EJF

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

G13EJF applies the Unscented Kalman Filter to a nonlinear state space model, with additive noise.

G13EJF uses reverse communication for evaluating the nonlinear functionals of the state space model.

2 Specification

```
SUBROUTINE G13EJF (IREVCM, MX, MY, Y, LX, LDLX, LY, LDLY, X, ST, LDST,     &
                   N, XT, LDXT, FXT, LDFXT, ROPT, LROPT, ICOMM, LICOMM,    &
                   RCOMM, LRCOMM, IFAIL)
INTEGER            IREVCM, MX, MY, LDLX, LDLY, LDST, N, LDXT, LDFXT,       &
                   LROPT, ICOMM(LICOMM), LICOMM, LRCOMM, IFAIL
REAL (KIND=nag_wp) Y(MY), LX(LDLX,*), LY(LDLY,*), X(MX), ST(LDST,*),       &
                   XT(LDXT,*), FXT(LDFXT,*), ROPT(LROPT),                  &
                   RCOMM(LRCOMM)
```

3 Description

G13EJF applies the Unscented Kalman Filter (UKF), as described in Julier and Uhlmann (1997b) to a nonlinear state space model, with additive noise, which, at time t, can be described by:

$$\begin{aligned} x_{t+1} &= F\left(x_t\right) + v_t \\ y_t &= H\left(x_t\right) + u_t \end{aligned}$$

where x_t represents the unobserved state vector of length m_x and y_t the observed measurement vector of length m_y. The process noise is denoted v_t, which is assumed to have mean zero and covariance structure Σ_x, and the measurement noise by u_t, which is assumed to have mean zero and covariance structure Σ_y.

3.1 Unscented Kalman Filter Algorithm

Given $\hat{x}_0$, an initial estimate of the state and P_0 and initial estimate of the state covariance matrix, the UKF can be described as follows:

(a) Generate a set of sigma points (see section Section 3.2):

$$\mathcal{X}_t = \left[\hat{x}_{t-1} \quad \hat{x}_{t-1} + \gamma\sqrt{P_{t-1}} \quad \hat{x}_{t-1} - \gamma\sqrt{P_{t-1}}\right] \tag{1}$$

(b) Evaluate the known model function F:

$$\mathcal{F}_t = F\left(\mathcal{X}_t\right) \tag{2}$$

The function F is assumed to accept the $m_x \times n$ matrix, $\mathcal{X}_t$ and return an $m_x \times n$ matrix, $\mathcal{F}_t$. The columns of both $\mathcal{X}_t$ and $\mathcal{F}_t$ correspond to different possible states. The notation $\mathcal{F}_{t,i}$ is used to denote the ith column of $\mathcal{F}_t$, hence the result of applying F to the ith possible state.

(c) Time Update:

$$\hat{x}_t = \sum_{i=1}^{n} W_i^m \mathcal{F}_{t,i} \tag{3}$$

$$P_t = \sum_{i=1}^{n} W_i^c \left(\mathcal{F}_{t,i} - \hat{x}_t\right)\left(\mathcal{F}_{t,i} - \hat{x}_t\right)^{\mathrm{T}} + \Sigma_x \tag{4}$$

(d) Redraw another set of sigma points (see section Section 3.2):

$$\mathcal{Y}_t = \begin{bmatrix} \hat{x}_t & \hat{x}_t + \gamma\sqrt{P_t} & \hat{x}_t - \gamma\sqrt{P_t} \end{bmatrix} \tag{5}$$

(e) Evaluate the known model function H:

$$\mathcal{H}_t = H\left(\mathcal{Y}_t\right) \tag{6}$$

The function H is assumed to accept the $m_x \times n$ matrix, $\mathcal{Y}_t$ and return an $m_y \times n$ matrix, $\mathcal{H}_t$. The columns of both $\mathcal{Y}_t$ and $\mathcal{H}_t$ correspond to different possible states. As above $\mathcal{H}_{t,i}$ is used to denote the ith column of $\mathcal{H}_t$.

(f) Measurement Update:

$$\hat{y}_t = \sum_{i=1}^{n} W_i^m \mathcal{H}_{t,i} \tag{7}$$

$$P_{yy_t} = \sum_{i=1}^{n} W_i^c \left(\mathcal{H}_{t,i} - \hat{y}_t\right)\left(\mathcal{H}_{t,i} - \hat{y}_t\right)^{\mathrm{T}} + \Sigma_y \tag{8}$$

$$P_{xy_t} = \sum_{i=1}^{n} W_i^c \left(\mathcal{F}_{t,i} - \hat{x}_t\right)\left(\mathcal{H}_{t,i} - \hat{y}_t\right)^{\mathrm{T}} \tag{9}$$

$$\mathcal{K}_t = P_{xy_t} P_{yy_t}^{-1} \tag{10}$$

$$\hat{x}_t = \hat{x}_t + \mathcal{K}_t(y_t - \hat{y}_t) \tag{11}$$

$$P_t = P_t - \mathcal{K}_t P_{yy_t} \mathcal{K}_t^{\mathrm{T}} \tag{12}$$

Here $\mathcal{K}_t$ is the Kalman gain matrix, $\hat{x}_t$ is the estimated state vector at time t and P_t the corresponding covariance matrix. Rather than implementing the standard UKF as stated above G13EJF uses the square-root form described in the Haykin (2001).

3.2 Sigma Points

A nonlinear state space model involves propagating a vector of random variables through a nonlinear system and we are interested in what happens to the mean and covariance matrix of those variables. Rather than trying to directly propagate the mean and covariance matrix, the UKF uses a set of carefully chosen sample points, referred to as sigma points, and propagates these through the system of interest. An estimate of the propagated mean and covariance matrix is then obtained via the weighted sample mean and covariance matrix.

For a vector of m random variables, x, with mean μ and covariance matrix Σ, the sigma points are usually constructed as:

$$\mathcal{X}_t = \begin{bmatrix} \mu & \mu + \gamma\sqrt{\Sigma} & \mu - \gamma\sqrt{\Sigma} \end{bmatrix}$$

When calculating the weighted sample mean and covariance matrix two sets of weights are required, one used when calculating the weighted sample mean, denoted W^m and one used when calculated the weighted sample covariance matrix, denoted W^c. The weights and multiplier, γ, are constructed as follows:

$$\begin{aligned}
\lambda &= \alpha^2(L + \kappa) - L \\
\gamma &= \sqrt{L + \lambda} \\
W_{\mathrm{i}}^m &= \begin{cases} \frac{\lambda}{L+\lambda} & i = 1 \\ \frac{1}{2(L+\lambda)} & i = 2, 3, \ldots, 2L+1 \end{cases} \\
W_{\mathrm{i}}^c &= \begin{cases} \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta & i = 1 \\ \frac{1}{2(L+\lambda)} & i = 2, 3, \ldots, 2L+1 \end{cases}
\end{aligned}$$

where, usually $L = m$ and α, β and κ are constants. The total number of sigma points, n, is given by $2L + 1$. The constant α is usually set to somewhere in the range $10^{-4} \leq \alpha \leq 1$ and for a Gaussian distribution, the optimal values of κ and β are $3 - L$ and 2 respectively.

Rather than redrawing another set of sigma points in (d) of the UKF an alternative method can be used where the sigma points used in (a) are augmented to take into account the process noise. This involves replacing equation (5) with:

$$\mathcal{Y}_t = \begin{bmatrix} \mathcal{X}_t & \mathcal{X}_{t,1} + \gamma\sqrt{\Sigma_x} & \mathcal{X}_{t,1} - \gamma\sqrt{\Sigma_x} \end{bmatrix} \tag{13}$$

Augmenting the sigma points in this manner requires setting L to $2L$ (and hence n to $2n-1$) and recalculating the weights. These new values are then used for the rest of the algorithm. The advantage of augmenting the sigma points is that it keeps any odd-moments information captured by the original propagated sigma points, at the cost of using a larger number of points.

4 References

Haykin S (2001) *Kalman Filtering and Neural Networks* John Wiley and Sons

Julier S J (2002) The scaled unscented transformation *Proceedings of the 2002 American Control Conference (Volume 6)* 4555–4559

Julier S J and Uhlmann J K (1997a) A consistent, debiased method for converting between polar and Cartesian coordinate systems *Proceedings of AeroSense97, International Society for Optics and Phonotonics* 110–121

Julier S J and Uhlmann J K (1997b) A new extension of the Kalman Filter to nonlinear systems *International Symposium for Aerospace/Defense, Sensing, Simulation and Controls (Volume 3)* **26**

5 Arguments

Note: this routine uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **IREVCM**. Between intermediate exits and re-entries, **all arguments other than FXT must remain unchanged**.

1: IREVCM – INTEGER *Input/Output*

On initial entry: must be set to 0 or 3.

If IREVCM $= 0$, it is assumed that $t = 0$, otherwise it is assumed that $t \neq 0$ and that G13EJF has been called at least once before at an earlier time step.

On intermediate exit: IREVCM $= 1$ or 2. The value of IREVCM specifies what intermediate values are returned by this routine and what values the calling program must assign to arguments of G13EJF before re-entering the routine. Details of the output and required input are given in the individual argument descriptions.

On intermediate re-entry: IREVCM must remain unchanged.

On final exit: IREVCM $= 3$

Constraint: IREVCM $= 0$, 1, 2 or 3.

2: MX – INTEGER *Input*

On entry: m_x, the number of state variables.

Constraint: MX ≥ 1.

3: MY – INTEGER *Input*

On entry: m_y, the number of observed variables.

Constraint: MY ≥ 1.

4: Y(MY) – REAL (KIND=nag_wp) array *Input*

On entry: y_t, the observed data at the current time point.

5: LX(LDLX, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array LX must be at least MX.

On entry: L_x, such that $L_x L_x^{\mathrm{T}} = \Sigma_x$, i.e., the lower triangular part of a Cholesky decomposition of the process noise covariance structure. Only the lower triangular part of LX is referenced.

If LDLX $= 0$, there is no process noise ($v_t = 0$ for all t) and LX is not referenced.

If Σ_x is time dependent, then the value supplied should be for time t.

6: LDLX – INTEGER *Input*

On entry: the first dimension of the array LX as declared in the (sub)program from which G13EJF is called.

Constraint: LDLX $= 0$ or LDLX $\geq$ MX.

7: LY(LDLY, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array LY must be at least MY.

On entry: L_y, such that $L_y L_y^{\mathrm{T}} = \Sigma_y$, i.e., the lower triangular part of a Cholesky decomposition of the observation noise covariance structure. Only the lower triangular part of LY is referenced.

If Σ_y is time dependent, then the value supplied should be for time t.

8: LDLY – INTEGER *Input*

On entry: the first dimension of the array LY as declared in the (sub)program from which G13EJF is called.

Constraint: LDLY $\geq$ MY.

9: X(MX) – REAL (KIND=nag_wp) array *Input/Output*

On initial entry: $\hat{x}_{t-1}$ the state vector for the previous time point.

On intermediate exit: when

IREVCM $= 1$
X is unchanged.

IREVCM $= 2$
$\hat{x}_t$.

On intermediate re-entry: X must remain unchanged.

On final exit: $\hat{x}_t$ the updated state vector.

10: ST(LDST, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array ST must be at least MX.

On initial entry: S_t, such that $S_{t-1} S_{t-1}^{\mathrm{T}} = P_{t-1}$, i.e., the lower triangular part of a Cholesky decomposition of the state covariance matrix at the previous time point. Only the lower triangular part of ST is referenced.

On intermediate exit: when

IREVCM $= 1$
ST is unchanged.

IREVCM $= 2$
S_t, the lower triangular part of a Cholesky factorization of P_t.

On intermediate re-entry: ST must remain unchanged.

On final exit: S_t, the lower triangular part of a Cholesky factorization of the updated state covariance matrix.

11: LDST – INTEGER *Input*

On entry: the first dimension of the array ST as declared in the (sub)program from which G13EJF is called.

Constraint: LDST $\geq$ MX.

12: N – INTEGER *Input/Output*

On initial entry: the value used in the sizing of the FXT and XT arrays. The value of N supplied must be at least as big as the maximum number of sigma points that the algorithm will use. G13EJF allows sigma points to be calculated in two ways during the measurement update; they can be redrawn or augmented. Which is used is controlled by ROPT.

If redrawn sigma points are used, then the maximum number of sigma points will be $2m_x + 1$, otherwise the maximum number of sigma points will be $4m_x + 1$.

On intermediate exit: the number of sigma points actually being used.

On intermediate re-entry: N must remain unchanged.

On final exit: reset to its value on initial entry.

Constraints: if IREVCM $= 0$ or 3,

if redrawn sigma points are used, N $\geq 2 \times$ MX $+ 1$;
otherwise N $\geq 4 \times$ MX $+ 1$.

13: XT(LDXT, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array XT must be at least max(MY, N).

On initial entry: need not be set.

On intermediate exit: X_t when IREVCM $= 1$, otherwise Y_t.

For the jth sigma point, the value for the ith parameter is held in XT(i, j), for $i = 1, 2, \ldots,$ MX and $j = 1, 2, \ldots,$ N.

On intermediate re-entry: XT must remain unchanged.

On final exit: the contents of XT are undefined.

14: LDXT – INTEGER *Input*

On entry: the first dimension of the array XT as declared in the (sub)program from which G13EJF is called.

Constraint: LDXT $\geq$ MX.

15: FXT(LDFXT, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array FXT must be at least N + max(MX, MY).

On initial entry: need not be set.

On intermediate exit: the contents of FXT are undefined.

On intermediate re-entry: $F(X_t)$ when IREVCM $= 1$, otherwise $H(Y_t)$ for the values of X_t and Y_t held in XT.

For the jth sigma point the value for the ith parameter should be held in FXT(i, j), for $j = 1, 2, \ldots,$ N. When IREVCM $= 1$, $i = 1, 2, \ldots,$ MX and when IREVCM $= 2$, $i = 1, 2, \ldots,$ MY.

On final exit: the contents of FXT are undefined.

16: LDFXT – INTEGER *Input*

On entry: the first dimension of the array FXT as declared in the (sub)program from which G13EJF is called.

Constraint: LDFXT $\geq \max(\text{MX}, \text{MY})$.

17: ROPT(LROPT) – REAL (KIND=nag_wp) array *Input*

On entry: optional arguments. The default value will be used for ROPT(i) if LROPT $< i$. Setting LROPT $= 0$ will use the default values for all optional arguments and ROPT need not be set.

ROPT(1)
If set to 1 then the second set of sigma points are redrawn, as given by equation (5). If set to 2 then the second set of sigma points are generated via augmentation, as given by equation (13).

Default is for the sigma points to be redrawn (i.e., ROPT$(1) = 1$)

ROPT(2)
κ_x, value of κ used when constructing the first set of sigma points, $\mathcal{X}_t$.

Defaults to $3 - \text{MX}$.

ROPT(3)
α_x, value of α used when constructing the first set of sigma points, $\mathcal{X}_t$.

Defaults to 1.

ROPT(4)
β_x, value of β used when constructing the first set of sigma points, $\mathcal{X}_t$.

Defaults to 2.

ROPT(5)
Value of κ used when constructing the second set of sigma points, $\mathcal{Y}_t$.

Defaults to $3 - 2 \times \text{MX}$ when LDLX $\neq 0$ and the second set of sigma points are augmented and κ_x otherwise.

ROPT(6)
Value of α used when constructing the second set of sigma points, $\mathcal{Y}_t$.

Defaults to α_x.

ROPT(7)
Value of β used when constructing the second set of sigma points, $\mathcal{Y}_t$.

Defaults to β_x.

Constraints:

ROPT$(1) = 1$ or 2;
ROPT$(2) > -\text{MX}$;
ROPT$(5) > -2 \times \text{MX}$ when LDLY $\neq 0$ and the second set of sigma points are augmented, otherwise ROPT$(5) > -\text{MX}$;
ROPT$(i) > 0$, for $i = 3, 6$.

18: LROPT – INTEGER *Input*

On entry: length of the options array ROPT.

Constraint: $0 \leq \text{LROPT} \leq 7$.

19: ICOMM(LICOMM) – INTEGER array *Communication Array*

On initial entry: ICOMM need not be set.

On intermediate exit: ICOMM is used for storage between calls to G13EJF.

On intermediate re-entry: ICOMM must remain unchanged.

On final exit: ICOMM is not defined.

20: LICOMM – INTEGER *Input*

On entry: the length of the array ICOMM. If LICOMM is too small and LICOMM ≥ 2 then IFAIL $= 201$ is returned and the minimum value for LICOMM and LRCOMM are given by ICOMM(1) and ICOMM(2) respectively.

Constraint: LICOMM ≥ 30.

21: RCOMM(LRCOMM) – REAL (KIND=nag_wp) array *Communication Array*

On initial entry: RCOMM need not be set.

On intermediate exit: RCOMM is used for storage between calls to G13EJF.

On intermediate re-entry: RCOMM must remain unchanged.

On final exit: RCOMM is not defined.

22: LRCOMM – INTEGER *Input*

On entry: the length of the array RCOMM. If LRCOMM is too small and LICOMM ≥ 2 then IFAIL $= 202$ is returned and the minimum value for LICOMM and LRCOMM are given by ICOMM(1) and ICOMM(2) respectively.

Suggested value: LRCOMM $= 30 + \text{MY} + \text{MX} \times \text{MY} + (1 + nb) \times \max(\text{MX}, \text{MY})$, where nb is the optimal ***block size***. In most cases a ***block size*** of 128 will be sufficient.

Constraint: LRCOMM $\geq 30 + \text{MY} + \text{MX} \times \text{MY} + 2 \times \max(\text{MX}, \text{MY})$.

23: IFAIL – INTEGER *Input/Output*

On initial 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, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On final 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 $= 11$

On entry, IREVCM $= \langle value \rangle$.
Constraint: IREVCM $= 0$, 1, 2 or 3.

IFAIL $= 21$

On entry, MX $= \langle value \rangle$.
Constraint: MX ≥ 1.

IFAIL $= 22$

MX has changed between calls.
On intermediate entry, MX $= \langle value \rangle$.
On initial entry, MX $= \langle value \rangle$.

IFAIL $= 31$

On entry, MY $= \langle value \rangle$.
Constraint: MY ≥ 1.

IFAIL $= 32$

MY has changed between calls.
On intermediate entry, MY $= \langle value \rangle$.
On initial entry, MY $= \langle value \rangle$.

IFAIL $= 61$

On entry, LDLX $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: LDLX $= 0$ or LDLX $\geq$ MX.

IFAIL $= 81$

On entry, LDLY $= \langle value \rangle$ and MY $= \langle value \rangle$.
Constraint: LDLY $\geq$ MY.

IFAIL $= 111$

On entry, LDST $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: LDST $\geq$ MX.

IFAIL $= 121$

On entry, augmented sigma points requested, N $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: N $\geq \langle value \rangle$.

IFAIL $= 122$

On entry, redrawn sigma points requested, N $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: N $\geq \langle value \rangle$.

IFAIL $= 123$

N has changed between calls.
On intermediate entry, N $= \langle value \rangle$.
On intermediate exit, N $= \langle value \rangle$.

IFAIL $= 141$

On entry, LDXT $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: LDXT $\geq$ MX.

IFAIL $= 161$

On entry, LDFXT $= \langle value \rangle$ and MX $= \langle value \rangle$.
Constraint: if IREVCM $= 1$, LDFXT $\geq$ MX.

IFAIL $= 162$

On entry, LDFXT $= \langle value \rangle$ and MY $= \langle value \rangle$.
Constraint: if IREVCM $= 2$, LDFXT $\geq$ MY.

IFAIL $= 171$

On entry, ROPT$(1) = \langle value \rangle$.
Constraint: ROPT$(1) = 1$ or 2.

IFAIL $= 172$

On entry, ROPT$(\langle value \rangle) = \langle value \rangle$.
Constraint: $\kappa > \langle value \rangle$.

IFAIL $= 173$

On entry, ROPT$(\langle value \rangle) = \langle value \rangle$.
Constraint: $\alpha > 0$.

IFAIL $= 181$

On entry, LROPT $= \langle value \rangle$.
Constraint: $0 \leq$ LROPT ≤ 7.

IFAIL $= 191$

ICOMM has been corrupted between calls.

IFAIL $= 201$

On entry, LICOMM $= \langle value \rangle$.
Constraint: LICOMM ≥ 2.
ICOMM is too small to return the required array sizes.

IFAIL $= 202$

On entry, LICOMM $= \langle value \rangle$ and LRCOMM $= \langle value \rangle$.
Constraint: LICOMM ≥ 30 and LRCOMM $\geq 30 +$ MY $+$ MX $\times$ MY $+ 2 \times \max(\text{MX}, \text{MY})$.
The minimum required values for LICOMM and LRCOMM are returned in ICOMM(1) and ICOMM(2) respectively.

IFAIL $= 211$

RCOMM has been corrupted between calls.

IFAIL $= 301$

A weight was negative and it was not possible to downdate the Cholesky factorization.

IFAIL $= 302$

Unable to calculate the Kalman gain matrix.

IFAIL $= 303$

Unable to calculate the Cholesky factorization of the updated state covariance matrix.

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

G13EJF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

As well as implementing the Unscented Kalman Filter, G13EJF can also be used to apply the Unscented Transform (see Julier (2002)) to the function F, by setting LDLX $= 0$ and terminating the calling sequence when IREVCM $= 2$ rather than IREVCM $= 3$. In this situation, on initial entry, X and ST would hold the mean and Cholesky factorization of the covariance matrix of the untransformed sample and on exit (when IREVCM $= 2$) they would hold the mean and Cholesky factorization of the covariance matrix of the transformed sample.

10 Example

This example implements the following nonlinear state space model, with the state vector x and state update function F given by:

$$\begin{aligned} m_x &= 3 \\ x_{t+1} &= \begin{pmatrix} \xi_{t+1} & \eta_{t+1} & \theta_{t+1} \end{pmatrix}^{\mathrm{T}} \\ &= F\left(x_t\right) + v_t \\ &= x_t + \begin{pmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5r & 0.5r \\ 0 & 0 \\ r/d & -r/d \end{pmatrix} \begin{pmatrix} \phi_{Rt} \\ \phi_{Lt} \end{pmatrix} + v_t \end{aligned}$$

where r and d are known constants and ϕ_{Rt} and ϕ_{Lt} are time-dependent knowns. The measurement vector y and measurement function H is given by:

$$\begin{aligned} m_y &= 2 \\ y_t &= \left(\delta_t, \alpha_t\right)^{\mathrm{T}} \\ &= H\left(x_t\right) + u_t \\ &= \begin{pmatrix} \Delta - \xi_t \cos A - \eta_t \sin A \\ \theta_t - A \end{pmatrix} + u_t \end{aligned}$$

where A and Δ are known constants. The initial values, x_0 and P_0, are given by

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad P_0 = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$$

and the Cholesky factorizations of the error covariance matrices, L_x and L_x by

$$L_x = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}, \quad L_y = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}.$$

10.1 Program Text

```
!   G13EJF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module g13ejfe_mod

!     G13EJF Example Program Module:
!     User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                            :: f, h, read_problem_data
!     .. Parameters ..
      Integer, Parameter, Public        :: mx = 3, my = 2, nin = 5, nout = 6
!     .. Derived Type Definitions ..
      Type, Public                      :: g13ej_problem_data
        Real (Kind=nag_wp)              :: delta, a, r, d
        Real (Kind=nag_wp)              :: phi_rt, phi_lt
      End Type g13ej_problem_data
    Contains
      Subroutine f(n,xt,fxt,dat)

!       .. Scalar Arguments ..
        Type (g13ej_problem_data), Intent (In) :: dat
        Integer, Intent (In)            :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: fxt(:,:)
        Real (Kind=nag_wp), Intent (In) :: xt(:,:)
!       .. Local Scalars ..
        Real (Kind=nag_wp)              :: t1, t3
        Integer                         :: i
!       .. Intrinsic Procedures ..
        Intrinsic                       :: cos, sin
!       .. Executable Statements ..
        Continue

        t1 = 0.5_nag_wp*dat%r*(dat%phi_rt+dat%phi_lt)
        t3 = (dat%r/dat%d)*(dat%phi_rt-dat%phi_lt)

        Do i = 1, n
          fxt(1,i) = xt(1,i) + cos(xt(3,i))*t1
          fxt(2,i) = xt(2,i) + sin(xt(3,i))*t1
          fxt(3,i) = xt(3,i) + t3
        End Do

        Return
      End Subroutine f
      Subroutine h(n,yt,hyt,dat)

!       .. Use Statements ..
        Use nag_library, Only: x01aaf
!       .. Scalar Arguments ..
        Type (g13ej_problem_data), Intent (In) :: dat
        Integer, Intent (In)            :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: hyt(:,:)
        Real (Kind=nag_wp), Intent (In) :: yt(:,:)
!       .. Local Scalars ..
        Real (Kind=nag_wp)              :: tmp
```

```
        Integer                       :: i
!       .. Intrinsic Procedures ..
        Intrinsic                     :: cos, sin
!       .. Executable Statements ..
        Continue

        Do i = 1, n
          hyt(1,i) = dat%delta - yt(1,i)*cos(dat%a) - yt(2,i)*sin(dat%a)
          hyt(2,i) = yt(3,i) - dat%a

!         Make sure that the theta is in the same range as the observed
!         data, which in this case is [0, 2*pi)
          If (hyt(2,i)<0.0_nag_wp) Then
            hyt(2,i) = hyt(2,i) + 2*x01aaf(tmp)
          End If
        End Do

        Return
      End Subroutine h
      Subroutine read_problem_data(t,dat,read_ok)
!       Read in any data specific to the F and H subroutines

!       .. Scalar Arguments ..
        Type (g13ej_problem_data), Intent (Inout) :: dat
        Integer, Intent (In)          :: t
        Logical, Intent (Out)         :: read_ok
!       .. Local Scalars ..
        Integer                       :: tt
!       .. Executable Statements ..
        Continue

        If (t==0) Then
!         Read in the data that is constant across all time points
          Read (nin,*) dat%r, dat%d, dat%delta, dat%a
          read_ok = .True.
        Else
!         Read in data for time point t
          Read (nin,*) tt, dat%phi_rt, dat%phi_lt
          If (tt/=t) Then
!           Sanity check
            Write (nout,99999) 'Expected to read in data for time point ', t
            Write (nout,99999) 'Data that was read in was for time point ', tt
99999       Format (A,E22.15)
            read_ok = .False.
          Else
            read_ok = .True.
          End If
        End If
      End Subroutine read_problem_data
    End Module g13ejfe_mod

    Program g13ejfe

!     .. Use Statements ..
      Use nag_library, Only: g13ejf, nag_wp
      Use g13ejfe_mod, Only: f, g13ej_problem_data, h, mx, my, nin, nout,      &
                             read_problem_data
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Type (g13ej_problem_data)       :: dat
      Integer                         :: i, ifail, irevcm, ldfxt, ldlx, ldly, &
                                         ldst, ldxt, licomm, lrcomm, lropt,   &
                                         n, ntime, t
      Logical                         :: read_ok
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: fxt(:,:), lx(:,:), ly(:,:),          &
                                         rcomm(:), ropt(:), st(:,:), x(:),    &
                                         xt(:,:), y(:)
      Integer, Allocatable            :: icomm(:)
!     .. Intrinsic Procedures ..
```

```
      Intrinsic                        :: abs, max, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G13EJF Example Program Results'
      Write (nout,*)

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

!     Using default optional arguments
      lropt = 0
      Allocate (ropt(lropt))

!     Allocate arrays
      n = 2*mx + 1
      If (lropt>=1) Then
        If (abs(ropt(1)-2.0_nag_wp)<=0.0_nag_wp) Then
          n = n + 2*mx
        End If
      End If
      ldlx = mx
      ldly = my
      ldst = mx
      ldxt = mx
      ldfxt = max(mx,my)
      licomm = 30
      lrcomm = 30 + my + mx*my + 2*max(mx,my)
      Allocate (lx(ldlx,mx),ly(ldly,my),x(mx),st(ldst,mx),xt(ldxt,max(my,      &
        n)),fxt(ldfxt,n+max(mx,my)),icomm(licomm),rcomm(lrcomm),y(my))

!     Read in the Cholesky factorization of the covariance matrix for the
!     process noise
      Do i = 1, mx
        Read (nin,*) lx(i,1:i)
      End Do

!     Read in the Cholesky factorization of the covariance matrix for the
!     observation noise
      Do i = 1, my
        Read (nin,*) ly(i,1:i)
      End Do

!     Read in the initial state vector
      Read (nin,*) x(1:mx)

!     Read in the Cholesky factorization of the initial state covariance
!     matrix
      Do i = 1, mx
        Read (nin,*) st(i,1:i)
      End Do

!     Read in the number of time points to run the system for
      Read (nin,*) ntime

!     Read in any problem specific data that is constant
      Call read_problem_data(0,dat,read_ok)
      If (.Not. read_ok) Then
        Go To 100
      End If

!     Title for first set of output
      Write (nout,*) ' Time   ', repeat(' ',(11*mx-16)/2), 'Estimate of State'
      Write (nout,*) repeat('-',7+11*mx)

!     Loop over each time point
      irevcm = 0
      Do t = 1, ntime

!       Read in any problem specific data that is time dependent
        Call read_problem_data(t,dat,read_ok)
        If (.Not. read_ok) Then
          Go To 100
```

```
        End If

!       Read in the observed data for time t
        Read (nin,*) y(1:my)

!       Call Unscented Kalman Filter routine
ukf_lp: Do
          ifail = 0
          Call g13ejf(irevcm,mx,my,y,lx,ldlx,ly,ldly,x,st,ldst,n,xt,ldxt,fxt,  &
            ldfxt,ropt,lropt,icomm,licomm,rcomm,lrcomm,ifail)
          Select Case (irevcm)
          Case (1)
!           Evaluate F(X)
            Call f(n,xt,fxt,dat)

          Case (2)
!           Evaluate H(X)
            Call h(n,xt,fxt,dat)

          Case Default
!           IREVCM = 3, finished
            Exit ukf_lp
          End Select
        End Do ukf_lp

!       Display the some of the current state estimate
        Write (nout,99999) t, x(1:mx)
      End Do

      Write (nout,*)
      Write (nout,*) 'Estimate of Cholesky Factorization of the State'
      Write (nout,*) 'Covariance Matrix at the Last Time Point'
      Do i = 1, mx
        Write (nout,99998) st(i,1:i)
      End Do

100   Continue

99999 Format (1X,I3,4X,10(1X,F10.3))
99998 Format (10(1X,E10.3))
    End Program g13ejfe
```

10.2 Program Data

```
G13EJF Example Program Data
0.1
0.0 0.1
0.0 0.0 0.1           :: End of LX
0.01
0.0 0.01              :: End of LY
0.0 0.0 0.0           :: Initial value for X
0.1
0.0 0.1
0.0 0.0 0.1           :: End of initial value for ST
15                    :: Number of time points
3.0 4.0 5.814  0.464  :: r, d, Delta, A
 1    0.4    0.1
      5.262  5.923
 2    0.4    0.1
      4.347  5.783
 3    0.4    0.1
      3.818  6.181
 4    0.4    0.1
      2.706  0.085
 5    0.4    0.1
      1.878  0.442
 6    0.4    0.1
      0.684  0.836
 7    0.4    0.1
      0.752  1.300
```

```
 8    0.4    0.1
      0.464  1.700
 9    0.4    0.1
      0.597  1.781
10    0.4    0.1
      0.842  2.040
11    0.4    0.1
      1.412  2.286
12    0.4    0.1
      1.527  2.820
13    0.4    0.1
      2.399  3.147
14    0.4    0.1
      2.661  3.569
15    0.4    0.1
      3.327  3.659    :: t, phi_rt, phi_lt, Y = (delta_t, alpha_a)
```

10.3 Program Results

```
G13EJF Example Program Results

 Time           Estimate of State
----------------------------------------
   1          0.664     -0.092      0.104
   2          1.598      0.081      0.314
   3          2.128      0.213      0.378
   4          3.134      0.674      0.660
   5          3.809      1.181      0.906
   6          4.730      2.000      1.298
   7          4.429      2.474      1.762
   8          4.357      3.246      2.162
   9          3.907      3.852      2.246
  10          3.360      4.398      2.504
  11          2.552      4.741      2.750
  12          2.191      5.193      3.281
  13          1.309      5.018      3.610
  14          1.071      4.894      4.031
  15          0.618      4.322      4.124

Estimate of Cholesky Factorization of the State
Covariance Matrix at the Last Time Point
 0.192E+00
-0.382E+00  0.222E-01
 0.158E-05  0.223E-06  0.995E-02
```

The example described above can be thought of as relating to the movement of a hypothetical robot. The unknown state, x, is the position of the robot (with respect to a reference frame) and facing, with (ξ, η) giving the x and y coordinates and θ the angle (with respect to the x-axis) that the robot is facing. The robot has two drive wheels, of radius r on an axle of length d. During time period t the right wheel is believed to rotate at a velocity of ϕ_{Rt} and the left at a velocity of ϕ_{Lt}. In this example, these velocities are fixed with $\phi_{Rt} = 0.4$ and $\phi_{Lt} = 0.1$. The state update function, F, calculates where the robot should be at each time point, given its previous position. However, in reality, there is some random fluctuation in the velocity of the wheels, for example, due to slippage. Therefore the actual position of the robot and the position given by equation F will differ.

In the area that the robot is moving there is a single wall. The position of the wall is known and defined by its distance, Δ, from the origin and its angle, A, from the x-axis. The robot has a sensor that is able to measure y, with δ being the distance to the wall and α the angle to the wall. The measurement function H gives the expected distance and angle to the wall if the robot's position is given by x_t. Therefore the state space model allows the robot to incorporate the sensor information to update the estimate of its position.

Example Program
Illustration of Position and Orientation
of Hypothetical Robot