NAG Toolbox

nag_tsa_kalman_unscented_state_revcom (g13ej)

1 Purpose

nag_tsa_kalman_unscented_state_revcom (g13ej) applies the Unscented Kalman Filter to a nonlinear state space model, with additive noise.

nag_tsa_kalman_unscented_state_revcom (g13ej) uses reverse communication for evaluating the nonlinear functionals of the state space model.

2 Syntax

```
[irevcm, x, st, xt, icomm, rcomm, ifail] = nag_tsa_kalman_unscented_state_revcom
(irevcm, y, lx, ly, x, st, xt1, fxt1, icomm1, rcomm1, 'mx', mx, 'my', my,
'ropt', ropt)
[irevcm, x, st, xt, icomm, rcomm, ifail] = g13ej(irevcm, y, lx, ly, x, st, xt1,
fxt1, icomm1, rcomm1, 'mx', mx, 'my', my, 'ropt', ropt)
```

3 Description

nag_tsa_kalman_unscented_state_revcom (g13ej) 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 nag_tsa_kalman_unscented_state_revcom (g13ej) 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 Parameters

Note: this function 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**.

5.1 Compulsory Input Parameters

1: **irevcm** – INTEGER

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 nag_tsa_kalman_unscented_state_revcom (g13ej) has been called at least once before at an earlier time step.

On intermediate re-entry: **irevcm** must remain unchanged.

Constraint: **irevcm** $= 0$, 1, 2 or 3.

2: **y**(**my**) – REAL (KIND=nag_wp) array

y_t, the observed data at the current time point.

3: **lx**($ldlx, :$) – REAL (KIND=nag_wp) array

The first dimension of the array **lx** must be at least **mx**.

The second dimension of the array **lx** must be at least **mx**.

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

4: **ly**($ldly, :$) – REAL (KIND=nag_wp) array

The first dimension of the array **ly** must be at least **my**.

The second dimension of the array **ly** must be at least **my**.

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.

5: **x**(**mx**) – REAL (KIND=nag_wp) array

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

On intermediate re-entry: **x** must remain unchanged.

6: **st**($ldst$, :) – REAL (KIND=nag_wp) array

The first dimension of the array **st** must be at least **mx**.

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 re-entry: **st** must remain unchanged.

7: **xt**($ldxt1$, :) – REAL (KIND=nag_wp) array

The first dimension, *ldxt1*, of the array **xt1** must satisfy

if **irevcm** $= 1$ or 2, *ldxt1* $=$ **mx**;
otherwise *ldxt1* ≥ 0.

The second dimension of the array **xt1** must be at least n if **irevcm** $= 1$ or 2, and at least 0 otherwise.

On initial entry: need not be set.

On intermediate re-entry: **xt1** must remain unchanged.

8: **fxt**($ldfxt1$, :) – REAL (KIND=nag_wp) array

The first dimension, *ldfxt1*, of the array **fxt1** must satisfy

if **irevcm** $= 1$, *ldfxt1* $=$ **mx**;
if **irevcm** $= 2$, *ldfxt1* $=$ **my**;
otherwise *ldfxt1* $>= 0$.

The second dimension of the array **fxt1** must be at least n if **irevcm** $= 1$ or 2, and at least 0 otherwise.

On initial entry: need not be set.

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**.

9: **icomm**(:) – INTEGER array

The dimension of the array must be at least 30 if **irevcm** $= 1$, 2 or 3, and at least 0 otherwise

10: **rcomm**(:) – REAL (KIND=nag_wp) array

The dimension of the array must be at least $30 +$ **my** $+$ **mx** $\times$ **my** $+ (1 + 128) \times \max($**mx**, **my**$)$ if **irevcm** $= 1$, 2 or 3, and at least 0 otherwise

5.2 Optional Input Parameters

1: **mx** – INTEGER

Default: the dimension of the array **x** and the first dimension of the arrays **st**, **lx** and the second dimension of the array **lx**. (An error is raised if these dimensions are not equal.)

m_x, the number of state variables.

Constraint: **mx** ≥ 1.

2: **my** – INTEGER

Default: the dimension of the array **y** and the first dimension of the array **ly** and the second dimension of the array **ly**. (An error is raised if these dimensions are not equal.)

m_y, the number of observed variables.

Constraint: **my** ≥ 1.

3: **ropt**$(lropt)$ – REAL (KIND=nag_wp) array

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 $3 -$ **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 $3 - 2 \times$ **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$(2) > -$**mx**;
ropt$(5) > -2 \times$ **mx** when $ldly \neq 0$ and the second set of sigma points are augmented, otherwise **ropt**$(5) > -$**mx**;
ropt$(i) > 0$, for $i = 3, 6$.

5.3 Output Parameters

1: **irevcm** – INTEGER

On intermediate exit: **irevcm** $= 1$ or 2. The value of **irevcm** specifies what intermediate values are returned by this function and what values the calling program must assign to arguments of nag_tsa_kalman_unscented_state_revcom (g13ej) before re-entering the routine. Details of the output and required input are given in the individual argument descriptions.

On final exit: **irevcm** $= 3$

2: **x**(**mx**) – REAL (KIND=nag_wp) array

On intermediate exit: when

irevcm $= 1$
x is unchanged.

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

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

3: **st**$(ldst, :)$ – REAL (KIND=nag_wp) array

The first dimension of the array **st** will be **mx**.

The second dimension of the array **st** will be **mx**.

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

4: **xt**$(ldxt, :)$ – REAL (KIND=nag_wp) array

The second dimension of the array **xt** will be n.

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 final exit: the contents of **xt** are undefined.

5: **icomm**$(licomm)$ – INTEGER array

$licomm = 30$.

On intermediate exit: **icomm** is used for storage between calls to nag_tsa_kalman_unscented_state_revcom (g13ej).

$licomm = 30$.

On final exit: **icomm** is not defined.

6: **rcomm**$(lrcomm)$ – REAL (KIND=nag_wp) array

$lrcomm = 30 + \mathbf{my} + \mathbf{mx} \times \mathbf{my} + 2 \times \max(\mathbf{mx}, \mathbf{my})$.

On intermediate exit: **rcomm** is used for storage between calls to nag_tsa_kalman_unscented_state_revcom (g13ej).

$lrcomm = 30 + \mathbf{my} + \mathbf{mx} \times \mathbf{my} + 2 \times \max(\mathbf{mx}, \mathbf{my})$.

On final exit: **rcomm** is not defined.

7: **ifail** – INTEGER

On final exit: **ifail** $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 11$

Constraint: **irevcm** $= 0$, 1, 2 or 3.

ifail $= 21$

Constraint: **mx** ≥ 1.

ifail $= 22$

mx has changed between calls.

ifail $= 31$

Constraint: **my** ≥ 1.

ifail $= 32$

my has changed between calls.

ifail $= 61$

Constraint: $ldlx = 0$ or $ldlx \geq$ **mx**.

ifail $= 81$

Constraint: $ldly \geq$ **my**.

ifail $= 111$

Constraint: $ldst \geq$ **mx**.

ifail $= 171$

Constraint: **ropt**$(1) = 1$ or 2.

ifail $= 172$

Constraint: $\kappa > \langle value \rangle$.

ifail $= 173$

Constraint: $\alpha > 0$.

ifail $= 181$

Constraint: $0 \leq lropt \leq 7$.

ifail $= 191$

icomm has been corrupted between calls.

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.

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 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

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

9.1 Program Text

```
function g13ej_example

fprintf('g13ej example results\n\n');

% Cholesky factorisation of the covariance matrix for
% the process noise
lx = 0.1 * eye(3);

% Cholesky factorisation of the covariance matrix for
% the observation noise
ly = 0.01 * eye(2);

% Initial state vector
ix = zeros(size(lx,1),1);
x = ix;

% Cholesky factorisation of the initial state covariance matrix
st = 0.1 * eye(3);

% Constant terms in the state space model
r = 3;
d = 4;
Delta = 5.814;
A = 0.464;

% Observed data, y = (delta, alpha)
y = [ 5.262 5.923; 4.347 5.783; 3.818 6.181;
      2.706 0.085; 1.878 0.442; 0.684 0.836;
      0.752 1.300; 0.464 1.700; 0.597 1.781;
      0.842 2.040; 1.412 2.286; 1.527 2.820;
      2.399 3.147; 2.661 3.569; 3.327 3.659];

% Number of time points to run the system for
ntime = size(y,1);

% phi_r and phi_l (these are the same across all time points in
% this example)
phi_r = ones(ntime,1) * 0.4;
phi_l = ones(ntime,1) * 0.1;

mx = numel(x);
my = size(ly,1);

% Reserve some space to hold the state
cx = zeros(mx,ntime);

irevcm = nag_int(0);
xt = [];
fxt = [];
icomm = nag_int([]);
rcomm = [];

% Loop over each time point
for t = 1:ntime
  % Observed data at time point t
  y_t = y(t,:);
  phi_rt = phi_r(t);
  phi_lt = phi_l(t);

  % Call the Unscented Kalman Filter routine
  while true
    [irevcm,x,st,xt,icomm,rcomm,ifail] = ...
```

```
    g13ej( ...
           irevcm,y_t,lx,ly, x,st,xt,fxt,icomm,rcomm);
    switch irevcm
      case 1
        % Evaluate F(x)
        fxt = f(xt,r,d,phi_rt,phi_lt);
      case 2
        % Evaluate H(x)
        fxt = h(xt,Delta,A);
      otherwise
        % irevcm = 3, finished
        break;
    end
  end

  % Store the current state
  cx(:,t) = x(:);
end

% Print the results
ttext = ['  Time  ' blanks(ceil((11*mx- 16)/2)) ' Estimate of State' ...
                   blanks(ceil((11*mx -16)/2))];
fprintf('%s\n',ttext);
ttext(:) = '-';
fprintf('%s\n',ttext);
for t = 1:ntime
  fprintf('  %3d   ', t);
  fprintf(' %10.3f', cx(1:mx,t));
  fprintf('\n');
end

fprintf('\nEstimate of Cholesky Factorisation of the State\n');
fprintf('Covariance Matrix at the Last Time Point\n');
for i=1:mx
  for j=1:i
    fprintf(' %10.2e',st(i,j));
  end
  fprintf('\n');
end

% Plot the results
fig1 = figure;

% calculate and plot the position and facing of the robot as if there
% were no slippage in the wheels
pos_no_slippage(:,1) = ix;
rot_mat = [r/2 r/2; 0 0;r/d -r/d];
for t=1:ntime
  v_r = rot_mat * [phi_r(t); phi_l(t)];
  theta = pos_no_slippage(3,t);
  T = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
  pos_no_slippage(:,t+1) = pos_no_slippage(:,t) + T*v_r;
end

% formula (of the form y = a + b x) for the position of the wall
b = -cos(A) / sin(A);
a = Delta * (sin(A) + cos(A)^2/sin(A));

% actual position and facing of the robot
% (this would usually be unknown, but this example
% is based on a simulation and hence we know the answer)
pos_actual = [0.000 0.617 1.590 2.192 ...
              3.238 3.947 4.762 4.734 ...
              4.529 3.955 3.222 2.209 ...
              2.047 1.137 0.903 0.443;
              0.000 0.000 0.101 0.079 ...
              0.474 0.908 1.947 1.850 ...
              2.904 3.757 4.675 5.425 ...
              5.492 5.362 5.244 4.674;
              0.000 0.103 0.036 0.361 ...
              0.549 0.906 1.299 1.763 ...
```

```
              2.164 2.245 2.504 2.749 ...
              3.284 3.610 4.033 4.123];

% produce the plot
h(1) = plot_robot(pos_no_slippage,'s','green','green');
hold on
h(2) = plot_robot(pos_actual,'c','red','red');
h(3) = plot_robot([zeros(3,1) cx],'c','blue','none');
hold off

% Add reference line for the wall
yl = ylim;
line([(yl(1) - a)/b (yl(2) - a) / b],yl,'Color','black');
xlim([-0.5 7]);

% Add title
title({'{\bf g13ej Example Plot}',
       'Illustration of Position and Orientation',
       ' of Hypothetical Robot'});

% Add legend
label = ['Initial' 'Actual' 'Updated'];
h(4) = legend(h,'Initial','Actual','Updated','Location','NorthEast');
set(h(4),'FontSize',get(h(4),'FontSize')*0.8);

% Add text to indicate wall
text(4.6,3.9,'Wall','Rotation',-63);

function [fxt] = f(xt,r,d,phi_rt,phi_lt)
  fxt = zeros(size(xt));

  t1 = 0.5*r*(phi_rt+phi_lt);
  t3 = (r/d)*(phi_rt-phi_lt);

  fxt(1,:) = xt(1,:) + cos(xt(3,:))*t1;
  fxt(2,:) = xt(2,:) + sin(xt(3,:))*t1;
  fxt(3,:) = xt(3,:) + t3;

function [hyt] = h(yt,Delta,A)
  hyt(1,:) = Delta - yt(1,:)*cos(A) - yt(2,:)*sin(A);
  hyt(2,:) = yt(3,:) - A;

  % Make sure that the theta is in the same range as the observed
  % data, which in this case is [0, 2*pi)
  hyt(2,(hyt(2,:) < 0)) = hyt(2,(hyt(2,:) < 0)) + 2 * pi;

function [h] =  plot_robot(x,symbol,colour,fill)
  alen = 0.3;
  h = scatter(x(1,:),x(2,:),60,colour,symbol,'MarkerFaceColor',fill);
  aend = [x(1,:)+alen*cos(x(3,:)); x(2,:)+alen*sin(x(3,:))];
  line([x(1,:); aend(1,:)],[x(2,:); aend(2,:)],'Color',colour);
```

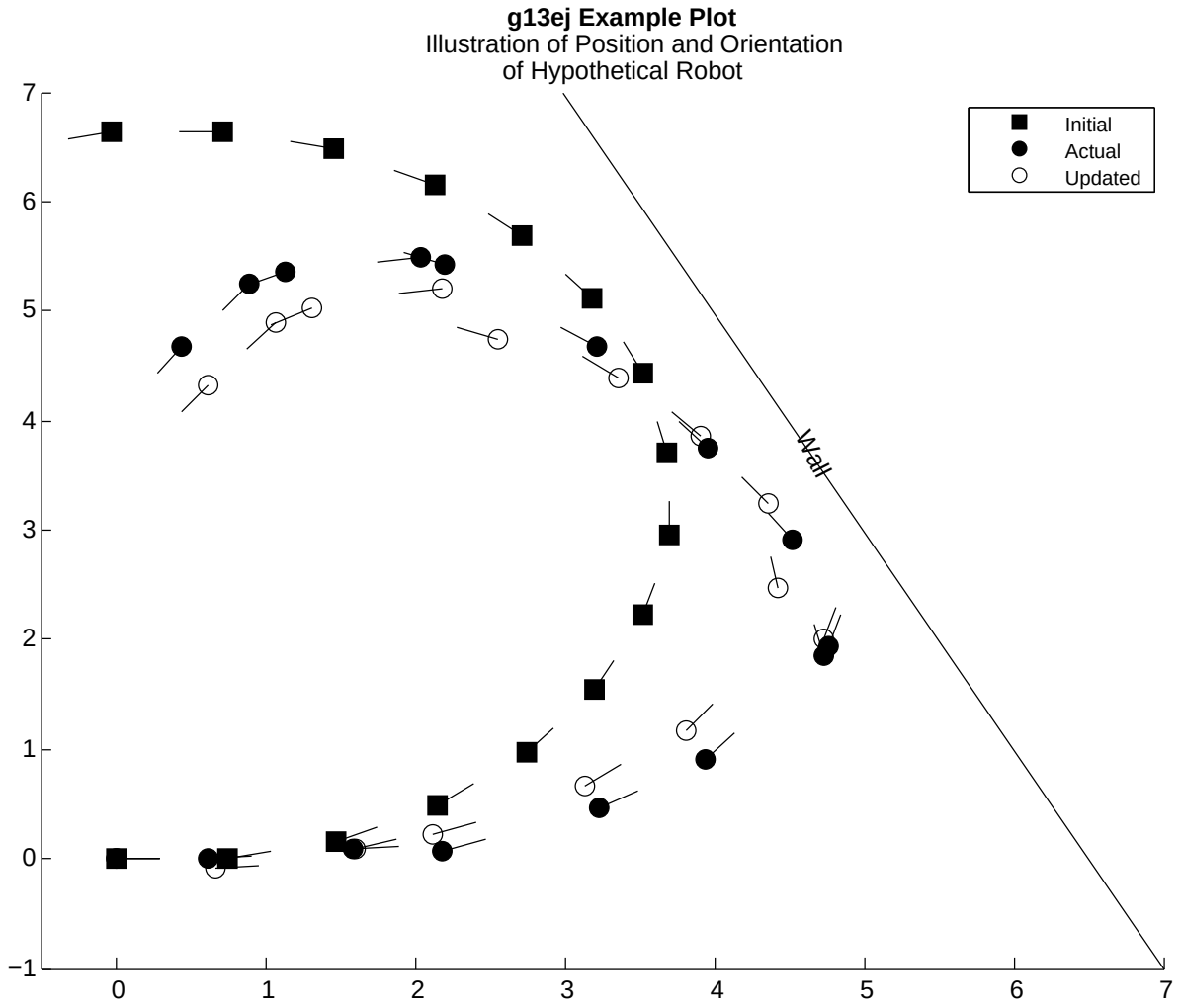
9.2 Program Results

```
     g13ej example 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 Factorisation of the State
Covariance Matrix at the Last Time Point
    1.92e-01
   -3.82e-01    2.22e-02
    1.58e-06    2.23e-07    9.95e-03
```



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.
