NAG Toolbox

nag_tsa_kalman_unscented_state (g13ek)

1 Purpose

nag_tsa_kalman_unscented_state (g13ek) applies the Unscented Kalman Filter (UKF) to a nonlinear state space model, with additive noise.

nag_tsa_kalman_unscented_state (g13ek) uses forward communication for evaluating the nonlinear functionals of the state space model.

2 Syntax

```
[x, st, user, ifail] = nag_tsa_kalman_unscented_state(y, lx, ly, f, h, x, st,
'mx', mx, 'my', my, 'user', user)
```

```
[x, st, user, ifail] = g13ek(y, lx, ly, f, h, x, st, 'mx', mx, 'my', my, 'user',
user)
```

3 Description

nag_tsa_kalman_unscented_state (g13ek) 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:

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

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 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:

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

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 3.2):

$$\mathcal{Y}_t = \left[\hat{x}_t \quad \hat{x}_t + \gamma\sqrt{P_t} \quad \hat{x}_t - \gamma\sqrt{P_t}\right] \tag{5}$$

(e) Evaluate the known model function H:

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

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 (g13ek) 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:

$$\mathcal{X}_t = \left[\mu \quad \mu + \gamma\sqrt{\Sigma} \quad \mu - \gamma\sqrt{\Sigma}\right]$$

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:

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

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.

The constants, κ, α and β are given by $\kappa = 3 - m_x$, $\alpha = 1.0$ and $\beta = 2$. If more control is required over the construction of the sigma points then the reverse communication function, nag_tsa_kalman_unscented_state_revcom (g13ej), can be used instead.

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

5.1 Compulsory Input Parameters

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

y_t, the observed data at the current time point.

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

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.

3: **ly**(**my**, **my**) – REAL (KIND=nag_wp) array

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.

4: **f** – SUBROUTINE, supplied by the user.

The state function, F as described in (b).

```
[fxt, user, info] = f(xt, user, info)
```

Input Parameters

1: **xt**(**mx**, n) – REAL (KIND=nag_wp) array

X_t, the sigma points generated in (a). For the jth sigma point, the value for the ith parameter is held in **xt**(i,j), for $i = 1, 2, \dots, m_x$ and $j = 1, 2, \dots, n$, where m_x is the number of state variables and n is the number of sigma points.

2: **user** – INTEGER array

f is called from nag_tsa_kalman_unscented_state (g13ek) with the object supplied to nag_tsa_kalman_unscented_state (g13ek).

3: **info** – INTEGER

info $= 0$.

Output Parameters

1: **fxt**(**mx**, n) – REAL (KIND=nag_wp) array

$F\,(X_t)$.

For the jth sigma point the value for the ith parameter should be held in **fxt**(i,j), for $i = 1,2,\ldots,m_x$ and $j = 1,2,\ldots,n$, where m_x is the number of observed variables and n is the number of sigma points supplied in **xt**.

2: **user** – INTEGER array

3: **info** – INTEGER

Set **info** to a nonzero value if you wish nag_tsa_kalman_unscented_state (g13ek) to terminate with **ifail** $= 61$.

5: **h** – SUBROUTINE, supplied by the user.

The measurement function, H as described in (e).

```
[hyt, user, info] = h(yt, user, info)
```

Input Parameters

1: **yt**(**mx**, n) – REAL (KIND=nag_wp) array

Y_t, the sigma points generated in (d). For the jth sigma point, the value for the ith parameter is held in **yt**(i,j), for $i = 1,2,\ldots,m_x$ and $j = 1,2,\ldots,n$, where m_x is the number of state variables and n is the number of sigma points.

2: **user** – INTEGER array

h is called from nag_tsa_kalman_unscented_state (g13ek) with the object supplied to nag_tsa_kalman_unscented_state (g13ek).

3: **info** – INTEGER

info $= 0$.

Output Parameters

1: **hyt**(**my**, n) – REAL (KIND=nag_wp) array

$H\,(Y_t)$.

For the jth sigma point the value for the ith parameter should be held in **hyt**(i,j), for $i = 1,2,\ldots,m_y$ and $j = 1,2,\ldots,n$, where m_y is the number of observed variables and n is the number of sigma points supplied in **yt**.

2: **user** – INTEGER array

3: **info** – INTEGER

Set **info** to a nonzero value if you wish nag_tsa_kalman_unscented_state (g13ek) to terminate with **ifail** $= 71$.

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

The dimension of the array **x** must be at least **mx**

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

7: **st**(**mx**, **mx**) – REAL (KIND=nag_wp) array

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.

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 arrays **st**, **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: **user** – INTEGER array

user is not used by nag_tsa_kalman_unscented_state (g13ek), but is passed to **f** and **h**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Output Parameters

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

The dimension of the array **x** will be **mx**

$\hat{x}_t$ the updated state vector.

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

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

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

3: **user** – INTEGER array

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 $= 11$

Constraint: **mx** ≥ 1.

ifail $= 21$

Constraint: **my** ≥ 1.

ifail $= 61$

User requested termination in **f**.

ifail $= 71$

User requested termination in **h**.

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:

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

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:

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

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

$$L_{\mathrm{x}} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}, \quad L_{\mathrm{y}} = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}$$

9.1 Program Text

```
     function g13ek_example

fprintf('g13ek 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 = [5.2620 4.3470 3.8180 2.7060 1.8780 0.6840 0.7520 ...
     0.4640 0.5970 0.8420 1.4120 1.5270 2.3990 2.6610 3.3270;
     5.9230 5.7830 6.1810 0.0850 0.4420 0.8360 1.3000 ...
     1.7000 1.7810 2.0400 2.2860 2.8200 3.1470 3.5690 3.6590];

% Number of time points to run the system for
ntime = size(y,2);

% 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);

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

% 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);

  % Store the information required by f and h in a cell array
  user = {Delta;A;r;d;phi_rt;phi_lt};
  % Update the state and its covariance matrix
  [x,st,user,ifail] = g13ek( ...
                           y_t,lx,ly,@f,@h,x,st,'user',user);

  % 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 g13ek 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,user,info] = f(xt,user,info)
  r = user{3};
  d = user{4};
  phi_rt = user{5};
  phi_lt = user{6};

  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;

  % Set info nonzero to terminate execution for any reason.
  info = nag_int(0);

function [hyt,user,info] = h(yt,user,info)
  Delta = user{1};
  A = user{2};

  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;

  % Set info nonzero to terminate execution for any reason.
  info = nag_int(0);

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);
```

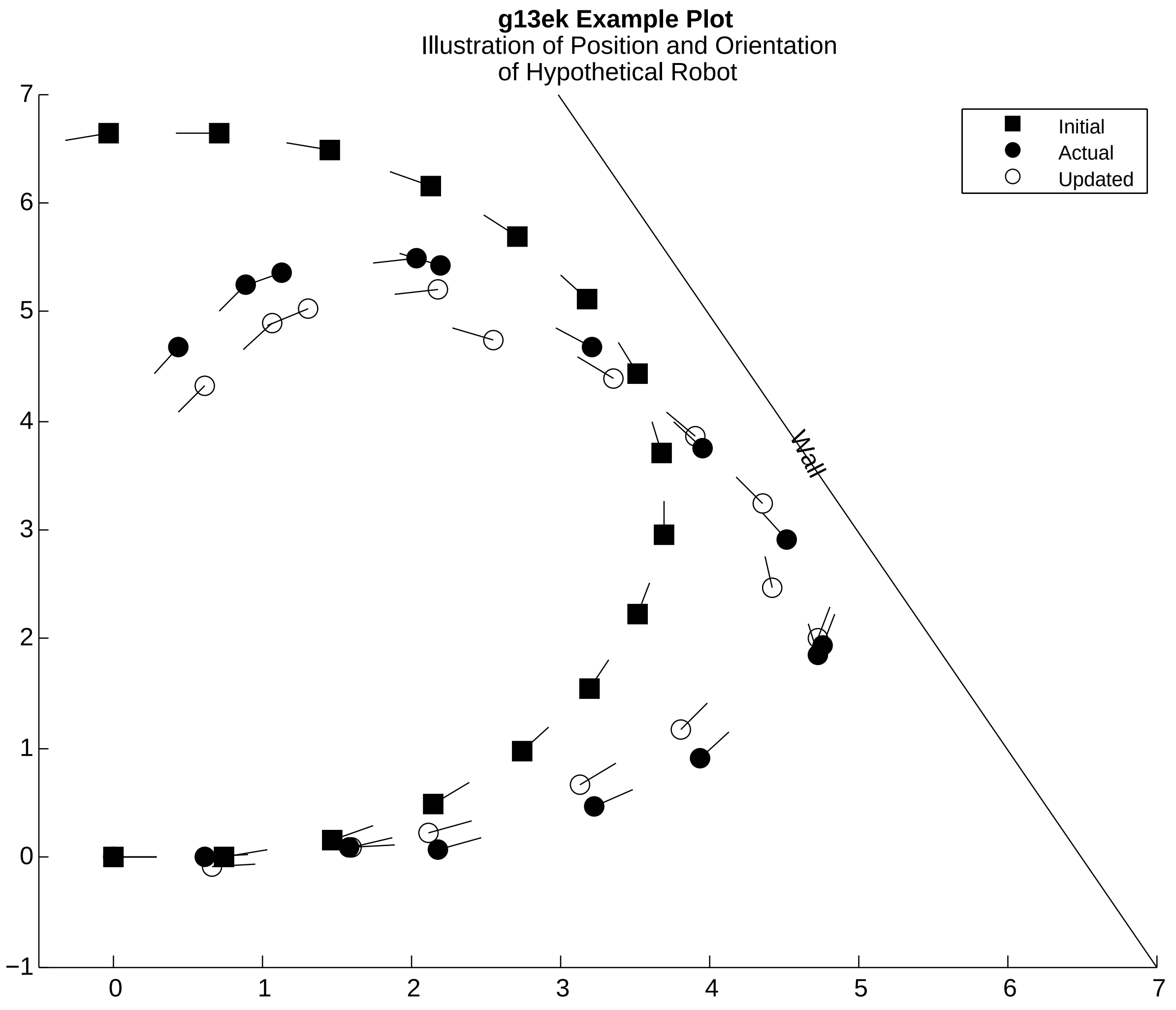
9.2 Program Results

```
     g13ek 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 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.