The Cooper Union Department of Electrical Engineering
Prof. Fred L. Fontaine
ECE416 Adaptive Algorithms
Project: Unscented Kalman Filter
November 13, 2023

A Tracking Problem

This example is taken from the following paper:
S. J. Julier, J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proc. IEEE,* vol. 92, no. 3, March 2004, pp. 401-422.

The Continuous-Time Model

From the article: *"A vehicle enters the atmosphere at high altitude and at a very high speed. The position of the body is tracked by a radar which measures range and bearing."* **Note:** range= distance, bearing= angle; essentially, this can be viewed as 2-D in polar coordinates.

Three forces are in effect:

1. Aerodynamic drag.
2. Gravity.
3. Random terms.

A "ballistic trajectory" means motion under constant acceleration. Initially, the trajectory is ballistic, but as the atmospheric density increases the drag effects become more significant and alters the trajectory.

We start with four state variables:

- The 2-D position $(x, y) = (x_1(t), x_2(t))$. These values are with respect to the center of the earth.
- The 2-D velocity $(v_x, v_y) = (x_3(t), x_4(t))$.

We are also interested in estimating the *ballistic coefficient* β. This coefficient is not known precisely, and may even be variable, yet it must always be positive. Our intention is to combine this parameter estimation with the underlying state estimation via JUKF. However, directly defining an auxiliary state variable as $\beta(n)$ is problematic because we need to enforce causality. The idea, then is to represent β in terms of an underlying unconstrained parameter. We choose:

$$\beta(t) = \beta_0 e^{x_5(t)}$$

and hence every value $\beta > 0$ is represented uniquely by a real value x_5, and conversely. Here, β_0 is a prescribed "nominal" value, and hence $x_5(t) = 0$ corresponds to $\beta(t) = \beta_0$.

The result is we introduce a fifth state variable:

$$x_5(t) = \log \frac{\beta(t)}{\beta_0}$$

The dynamical equations are:

$$\begin{aligned}
\dot{x}_1(t) &= x_3(t) + \varepsilon_1^{(x)}(t) \\
\dot{x}_2(t) &= x_4(t) + \varepsilon_2^{(x)}(t) \\
\dot{x}_3(t) &= D(t)\, x_3(t) + G(t)\, x_1(t) + \varepsilon_3^{(x)}(t) \\
\dot{x}_4(t) &= D(t)\, x_4(t) + G(t)\, x_2(t) + \varepsilon_4^{(x)}(t) \\
\dot{x}_5(t) &= \varepsilon_5^{(x)}(t)
\end{aligned}$$

where $\varepsilon_k^{(x)}(t)$ are uncorrelated process disturbances. Note x_5 is driven purely by random disturbance; without the disturbance, $\beta(t) = \beta_0$ would be a known constant.

The nonlinearities enter through the drag D and gravity G:

$$\begin{aligned}
D(t) &= -\beta(t)\, e^{(R_0 - R(t))/H_0} V(t) \\
G(t) &= -\frac{Gm_0}{R^3(t)}
\end{aligned}$$

where R is the distance from the center of the earth, V is the (scalar) speed, and physical constants are:

$$\begin{aligned}
Gm_0 &= 3.986 \times 10^5 \\
R_0 &= 6374 \\
H_0 &= 13.406
\end{aligned}$$

Specifically:

$$\begin{aligned}
\beta(t) &= \beta_0 e^{x_x(t)} \\
R(t) &= \sqrt{x_1^2(t) + x_2^2(t)} \\
V(t) &= \sqrt{x_3^2(t) + x_4^2(t)}
\end{aligned}$$

The radar is located at $(x_r, y_r) = (R_0, 0)$, and measured range and bearing:

$$\begin{aligned}
r(t) &= \sqrt{(x_1(t) - x_r)^2 + (x_2(t) - y_r)^2} + \varepsilon_1^{(y)}(t) \\
\theta(t) &= \tan^{-1} \frac{x_2(t) - y_r}{x_1(t) - x_r} + \varepsilon_2^{(y)}(t)
\end{aligned}$$

where $\varepsilon_k^{(y)}$ are the measurement disturbances, and out output measurement vector is:

$$\mathbf{y}(t) = \begin{bmatrix} r(t) \\ \theta(t) \end{bmatrix}$$

Discretized Model
We assume the signals are sampled at $f_s = 10Hz$, i.e., once every $dt = 0.1$ sec. Thus, $t = ndt$ where n is our discrete-time index.

The discrete-time measurement equation is obtained from the continuous time formula for $\mathbf{y}(t)$ simply by substituting $t = ndt$. The more challenging task is discretizing the process equation. Let us consider the more general form:

$$\dot{x}(t) = f\left(x(t), t, \varepsilon^{(x)}(t), u(t)\right)$$

where $\varepsilon^{(x)}$ is an unknown disturbance and u, if present, is a known input; u may also represent a set of **known** parameter values.

The simplest discretization is called the *Euler approximation:*

$$x_{n+1} = x_n + dt \cdot f\left(x_n, ndt, v_n^{(x)}, u_n\right)$$

In our case, the disturbance is additive, so in the discrete form it is scaled by dt. (Thus for example, given a presumed variance for the continuous-time disturbance, the variance of the discrete-time disturbance should be scaled by $(dt)^2$).

The authors suggest this is not sufficient accurate for the dynamical system at hand. They propose a better approach, called the *midpoint approximation.* This scheme does a two-step update:

$$t \rightarrow t + \frac{1}{2}dt \rightarrow t + dt$$

Of course, one could just cut the sampling period in half, and employ $\frac{1}{2}dt$. But this would double the total number of iterations we need to run over a fixed period of time, and that increases computational complexity significantly. With the midpoint method, two iterations are folded into one; equivalently, not all intermediate values in the additional iterations are computed. In any case, the application of the midpoint method yields:

$$x_{n+1} = x_n + dt \cdot f\left(x_n + \frac{1}{2}dt \cdot f\left(x_n, ndt, 0, u_n\right), n + \frac{1}{2}dt, \varepsilon_n^{(x)}, u_n\right)$$

MATLAB Code for the System Model Equations

- *uhlprocsim* provides either Euler or midpoint simulation of the process equation, calling *uhlproc* internally.
- *uhlmeas* is the code for the measurement equation.

The Sigma Points
In order to either simulate the trajectory or run a UKF on the system, you just need to call *uhlprocsim* and *uhlmeas.* For this project, make sure the **midpoint** technique is used.

For the UKF, I suggest using the sigma point generation method as in the article, summarized here. For the case of an N_x-dimensional random vector X with mean vector $\mathbf{0}$ and covariance matrix I (identity), the $2N_x + 1$ sigma points and weights are:

$$\begin{aligned}
\boldsymbol{\chi}_0 &= \mathbf{0}, \quad w_0 = \frac{1}{3} \\
\boldsymbol{\chi}_n &= \sqrt{\frac{N_x}{1-w_0}}\mathbf{e}_n, \quad w_n = \frac{1-w_0}{2N_x}\ , \quad 1 \leq n \leq N_x \\
\chi_{N_x+n} &= -\sqrt{\frac{N_x}{1-w_0}}\mathbf{e}_n, \quad w_{N_x+n} = \frac{1-w_0}{2N_x}\ , \quad 1 \leq n \leq N_x
\end{aligned}$$

where $\mathbf{e}_n \in \mathbb{R}^{N_x}$ has 1 in the n^{th} coordinate and 0 otherwise.

For a general mean vector $\boldsymbol{\mu}$ and covariance Σ, let $\Sigma^{1/2}$ denote the (lower) Cholesky factor and apply the usual transformation:

$$\chi_n \longrightarrow \boldsymbol{\mu} + \Sigma^{1/2}\boldsymbol{\chi}_n$$

The result is the sigma points are comprised of the mean, and the mean $\pm$ columns of the Cholesky factor scaled by a factor as above.

Observe that each iteration requires computation of Cholesky factors, and this is the primary computational burden of the UKF. This justifies the use of the midpoint method over simply cutting the sample period in half, as it reduces the overall number of requisite Cholesky decompositions roughly in half.

Initial Conditions

Assume initial conditions for the vehicle are:

$$x\left(0\right) = \begin{bmatrix} 6400.4 \\ 349.14 \\ -1.8093 \\ -6.7967 \\ 0.6932 \end{bmatrix}$$

and the disturbance covariance matrices are:

$$\begin{aligned}
Q_x &= diag\left\{10^{-8}, 10^{-8}, 2.404 \times 10^{-5}, 2.404 \times 10^{-5}, 10^{-8}\right\} \\
Q_y &= diag\left\{1, 17 \times 10^{-3}\right\}
\end{aligned}$$

Assume the initial conditions for the UKF are:

$$\hat{x}\left(1|0\right) = \begin{bmatrix} 6400 \\ 350 \\ -2 \\ -7 \\ 0.65 \end{bmatrix}$$

which represents that we have approximate knowledge of the true initial conditions, and:

$$K\left(1,0\right) = diag\left\{10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 1\right\}$$

It is not unreasonable to assume we have approximate knowledge of the initial position and velocity. The ballistic coefficient is harder to determine, hence we are building in a larger initial error in it. Also, setting the corresponding value in the initial $K(1,0)$ larger enables the UKF to start with a wider "net" if you will (thing of the spread of the sigma points), corresponding to the larger initial uncertainty we have in that parameter.

Finally, the simulation should be over 50 seconds, which means 500 time steps.

The Assignment: Theory and Setup

1. Following the work in the lecture notes, assuming 0 means and covariance I, show that the empirical mean of the proposed sigma points is 0, the empirical covariance is I, the empirical skewness of each component is 0, and the empirical Kurtosis of each component is:
$$\frac{N_x}{1-w_0}$$
In this case, $w_0 = 1/3$ and $N_x = 5$. Evaluate this, compare to the Kurtosis of Gaussian and comment. The dimension $N_x = 5$ is fixed. Show that selecting w_0 to get the Kurtosis of 3 would require $w_0 < 0$, which is problematic. This set of sigma points does not match Gaussian Kurtosis, but avoids negative weights.

2. Write a function:
$$[w, sig] = sigmapoints(mu, Cchol)$$
that generates the sigma points and weights using the above formulas, given input mean vector and Cholesky factor (i.e., not the direct covariance matrix). **Remark:** Don't use *chol* as a variable name, as that is the name of the MATLAB function that computes Cholesky factors!

3. Write a function:
$$sigmu = sigmamean\,(w, sig)$$
that computes the empirical mean from given sigma points and weights.

4. Write a function called:
$$sigcov = sigmacov(w, sigx, sigy)$$
that will compute the empirical cross-covariance for X, Y prescribed by sigma points $sigx, sigy$ respectively, with common weights w. This function, when called simply as $sigmacov\,(w, sigx)$ should compute the auto-covariance matrix. Note that *sigmacov* should internally call *sigmamean.*

5. Write a function that performs one iteration of the UKF:
$$[xpnew, Kpnew, xe, Ke] = ukfilter(xp, Kp, Qx, Qy, ymeas, n, dt)$$
Here, xp, Kp are initial predicted state and prediction error covariance matrix, respectively, and it outputs both the estimated values xe, Ke (estimated state and associated covariance matrix), and updated predicted values $xpnew, Kpnew$. The process and

measurement covariance matrices are Qx, Qy. The measured vector is $ymeas$, n is the discrete iteration index, and dt is the parameter for the discretization. Internally, it should call *uhlprocsim,uhlmeas,sigmapts,sigmamean,sigmacov.*

The Assignment: Experiment

You should simulate the system and simultaneously run the UKF. As you update each state and produce the measurement (r, θ), run the UKF to keep up with estimated/predicted states and covariance matrices. Then create three graphs:

1. Plot the actual position and estimated trajectory (different colors, superimposed). That is, plot state variable x_1 versus state variable x_2. Place a marker and text label at the initial point, so we can see where it starts.

2. Plot the actual velocity trajectory and its estimate (different colors, superimposed). That is, plot state variable x_3 versus state variable x_4. Again, place a marker at the starting point.

3. Plot the actual $\beta = \beta_0 e^{x_r}$ and the estimated value (i.e., using the estimated x_5 state variable), superimposed. Although we are not directly performing a parameter estimation with a JUKF, in effect inclusion of β as a state variable (actually, parametrized in a way to enforce positivity in β, which is a common technique) **is** in fact how parameter estimation would be done anyway! As a remark, since this is embedded with the other states, the covariance matrices the UKF generates implies we are not assuming the uncertainty in β is uncorrelated with that of the other states. Hence, this is closer in form to a JUKF [Joint UKF] than a DUKF [Dual UKF] (which would run a separate UKF for the β and for the other states).

Now put a whole wrapper around the whole thing and repeat it 5 times- in the end there will be 5 sets of 3 graphs. [You don't need to store things between runs] You are doing this to see how much variability there may be, informally. Comment on any observations you can make in this regard.