navbox+

# -An Unscented Estimation and Adaptive Control Package-

Author: Jason Nezvadovitz

Created: June 1, 2017

This is a work in progress. The ROS package demos and most of the documentation are not finished.

To install the Python package, navigate to this folder and do: `sudo python setup.py install`

This Python package provides an unscented Kalman filter (UKF) for online state and parameter estimation, and a general framework to feed those estimates into an adaptive controller. The only major assumptions are that:

- The physical system is Markovian with respect to some state
- That state lives on a finite-dimensional smooth manifold
- Process noise and sensor noise are independently sampled at each timestep
- Only the mean and covariance of any noises are known / available for use
- The true state's underlying probability distribution is unimodal
- Most uncertainty in the process is parametric

More importantly, this package (will eventually provide) some *usable* demos configuring navbox+ for a variety of robots, including boats and submarines, all integrated with ROS.

## Notation

The physical system under consideration is modeled over time $t$ discretized by a chosen $\Delta t$ as,

$$x(t + \Delta t) = f(x(t), u, \omega_f, \Delta t)$$

where $x \in \mathcal{M}$ is the system state, $u \in \mathbb{R}^{n_u}$ is the input we can control, and $\omega_f \sim (\bar{\omega}_f, C_f)$ is a random vector ("process noise") distributed with mean $\bar{\omega}_f \in \mathbb{R}^{n_{\omega_f}}$ and covariance matrix $C_f$. Of the above variables, we only assume that $u$, $\bar{\omega}_f$, and $C_f$ are known at all times.

The smooth $n_m$-dimensional manifold $\mathcal{M}$ that $x$ lives on (often called the state space) has to be understood a little. We must be capable of computing the exponential mapping between vectors in the tangent space of this manifold and points on the manifold itself. Specifically, we require two special operations: "boxplus" and "boxminus".

The boxplus operation, $\boxplus : \mathcal{M} \times \mathbb{R}^{n_m} \to \mathcal{M}$ perturbs a state on $\mathcal{M}$ by a vector tangent to $\mathcal{M}$. I.e. for any $x \in \mathcal{M}$ and any $v \in \mathcal{T}_x(\mathcal{M})$, the result of $x \boxplus v$ is another point on $\mathcal{M}$ that is the projection of $v$ back onto $\mathcal{M}$. The boxminus operation $\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^{n_m}$ is the inverse operation to boxplus, i.e. $(x \boxplus v) \boxminus x = v$. If your state is just a vector on $\mathbb{R}^{n_m}$ then boxplus and boxminus are just vector addition and subtraction. However, if your state is or includes any non-vector components like quaternions, I highly suggest reading the paper linked above to further understand boxoperations.

The navbox+ package provides a few tools to help you configure your boxoperations. You may also begin to notice the pun in the package name.

Anyway, we also have a suite of $n_h$ memoryless sensors modeled as,

$$z_i = h_i(x, u, \omega_{h_i}), \quad i = 1, 2, \ldots, n_h$$

where $z_i \in \mathbb{R}^{n_{z_i}}$ is the output of the ith sensor, corrupted by sensor noise $\omega_{h_i} \sim (\bar{\omega}_{h_i}, C_{h_i})$. The sensor noise mean and covariance are always known, but the measurements $z_i$ can arrive intermittently.

So here's the deal: $x$ can contain all your hopes and dreams. Typically, $f$ and the $h_i$ require a ton of physical parameters / biases that are difficult to experimentally determine. Or, your model may not even have the exact right form, so treating the parameters as time-varying may be necessary for flexibility through operating modes. To reconcile this,

1. Define $x_q \in \mathcal{M}_q$ as the true system states (not parameters).
2. Define $x_p \in \mathbb{R}^{n_p}$ as a vector of the *distinguishable* unknown parameters in $f$ and the $h_i$.
3. Let $x \in \mathcal{M} = \mathcal{M}_q \times \mathbb{R}^{n_p}$ be the concatenation of $x_q$ and $x_p$, i.e. $|\mathcal{M}| = n_m = n_q + n_p$. Note that $x$ itself can still be described with $n_x \geq n_m$ values when using redundant state representations like quaternions.

So what is meant by "distinguishable" parameters? Well consider some $y = ax + \sin(bx)$. Here $a$ and $b$ are distinguishable. Suppose you ran a system identification finding $a = 4$ and $b = 2$. Then I tell you that $a$ is really a combination of two other parameters, $a_1$ and $a_2$, like perhaps $y = (a_1 + a_2)x + \sin(bx)$ or $y = a_1 a_2 x + \sin(bx)$. The parameters $a_1$ and $a_2$ are indistinguishable because any combination of numbers that sum (first example) or multiply (second example) to $a = 4$ will still satisfy the identification. However, if the function was, say, $y = (a_1+a_2)x+\sin(a_1+bx)$, then $a_1$ and $a_2$ are distinguishable because $a_1$ contributes uniquely elsewhere. In short, indistinguishable parameters are those that connect themselves to the state in an identical way.

If your equations ($f$ and the $h_i$) are linear in their parameters, it is really easy to spot and consolidate indistinguishable parameters. Fortunately, most robot models are linear in their parameters. While navbox+ can work on systems nonlinear in their parameters, things can go wrong because indistinguishability may be lurking within your parameterization.

Now then, if you can manage to get a good model with navbox+ online system identification, then you are poised to construct a great adaptive controller. The controller is a function,

$$u = g(r(t), r(t + dt), \hat{x}, C_x, dt)$$

where $r \in \mathcal{M}_q$ is the desired value of the non-parameter states (i.e. the "reference") and $C_x$ is the covariance of $\hat{x}$, our current estimate of $x$. This controller is adaptive because it makes use of parameters identified in realtime (i.e. the $\hat{x}_p$ part of $\hat{x}$). If the parameter estimates were perfect, then $g$ could simply be $f$ with $r$ plugged into $x_q$ and then solved for $u$. However perfection is impractical, so remember to be safe and always wear a feedback term.

One more thing! If you happen to have a state derivative sensor too (like an accelerometer), you can still incorporate it in a variety of ways. The one we suggest is to append this measured derivative to the state vector and treat the sensor as an update for it. Cross-correlation between these derivative values and the other true states will couple the accelerometer information to the rest of the filter.

A table is provided on the following page to summarize most of the notation used in this package.

...

**Table of Notation**

| Symbol | Space / Args | Meaning | Code |
|---|---|---|---|
| $t$ | $\mathbb{R}$ | Time | `t` |
| $\Delta t$ | $\mathbb{R}$ | Discrete timestep | `dt` |
| $f$ | $\mathcal{M} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_{\omega_f}} \times \mathbb{R} \mapsto \mathcal{M}$ | State advance function | `f` |
| $\hat{x}$ | $\mathcal{M} = \mathcal{M}_q \times \mathbb{R}^{n_p}$ | Full state estimate (contains $n_x \geq n_m$ values) | `x` |
| $\hat{x}_q$ | $\mathcal{M}_q$ | Non-parameter part of the state estimate | `xq` |
| $\hat{x}_p$ | $\mathbb{R}^{n_p}$ | Parameter part of the state estimate | `xp` |
| $C_x$ | $\mathbb{R}^{n_m \times n_m}$ | Full state estimate covariance matrix | `Cx` |
| $u$ | $\mathbb{R}^{n_u}$ | Control input | `u` |
| $\omega_f$ | $\mathbb{R}^{n_{\omega_f}}$ | Process noise | `wf` |
| $\bar{\omega}_f$ | $\mathbb{R}^{n_{\omega_f}}$ | Process noise mean | `wf0` |
| $C_f$ | $\mathbb{R}^{n_{\omega_f} \times n_{\omega_f}}$ | Process noise covariance matrix | `Cf` |
| $z_i$ | $\mathbb{R}^{n_{z_i}}$ | Measurement from a sensor | `z` |
| $h_i$ | $\mathcal{M} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_{\omega_{h_i}}} \mapsto \mathbb{R}^{n_{z_i}}$ | Model of a sensor | `h` |
| $\omega_{h_i}$ | $\mathbb{R}^{n_{\omega_{h_i}}}$ | Noise in a sensor | `wh` |
| $\bar{\omega}_{h_i}$ | $\mathbb{R}^{n_{\omega_{h_i}}}$ | Mean of the noise in a sensor | `wh0` |
| $C_{h_i}$ | $\mathbb{R}^{n_{\omega_{h_i}} \times n_{\omega_{h_i}}}$ | Covariance matrix of a sensor's noise | `Ch` |
| $g$ | $\mathcal{M}_q \times \mathcal{M}_q \times \mathcal{M} \times \mathbb{R}^{n_m \times n_m} \times \mathbb{R} \mapsto \mathbb{R}^{n_u}$ | Controller function | `g` |
| $r$ | $\mathcal{M}_q$ | Desired non-parameter state | `r` |
| $\boxplus$ | $\mathcal{M} \times \mathbb{R}^{n_m} \mapsto \mathcal{M}$ | Boxplus | `xplus` |
| $\boxminus$ | $\mathcal{M} \times \mathcal{M} \mapsto \mathbb{R}^{n_m}$ | Boxminus | `xminus` |

# Configuration / Usage

pass

# References

pass