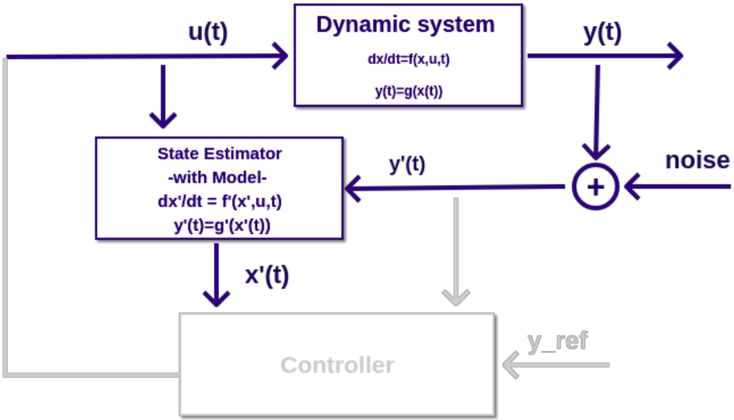
State Estimation
Overview



State Estimation

Kalman Filter Usage

- An initial guess of $\mathbf{x}(t=0)$ and the level of confidence for it, $\mathbf{P}_0$;
- Uses a dynamical model approximation of the system (i.e. f' and g), and a description of the confidence of the approximation, $\mathbf{Q}$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \qquad \dot{\mathbf{x}'} = f'(\mathbf{x}', \mathbf{u}, t) + noise(\mathbf{Q})$$

$$\mathbf{y} = g(\mathbf{x}) \qquad \mathbf{y}' = g(\mathbf{x}')$$

- The noise level expected on the physical sensors, $\mathbf{R}$

State Estimation
Kalman Filter Usage

- An initial guess of $\mathbf{x}(t=0)$ and the level of confidence for it, $\mathbf{P}_0$;

```
\# x = [pos,euler,ve]
x0 = np.array([ meas_pos[0], meas_pos[1], meas_pos[2], 0, 0, 0, 0, 0, 0 ])
P0 = np.diag([100.0, 100.0, 100.0, 0.01, 0.01, 9.0, 9.0, 9.0, 9.0])
```

- A dynamical model approximation of the system (i.e. f' and g), and a description of the confidence of the approximation, $\mathbf{Q}$

```
dfx = rigidbody.quadrotor_dt_kinematic_euler
meas_func = lambda x: x[0:3]
Q = np.diag([0.01, 0.01, 0.01, 0.001, 0.001, 0.001, 0.01, 0.01, 0.01])
```

- The noise level expected on the physical sensors, $\mathbf{R}$

```
R = sigma_gps**2*np.diag([1,1,1])
```

State Estimation

Kalman Filter Usage

- Predict Step: running the model forward in time with the known inputs, also also update the incertitude

```
filter.predict(np.array([meas_ax_av,meas_ay_av,meas_az_av,meas_gx_av,meas_gy_av,meas_gz_av])/
(dt_kf_predict/dt_sim), dt_kf_predict, 1)
```

- Correction Step: take in measurement data, with the given confidence level, and update the model states and incertitude

```
filter.update(meas_pos, 1)
```