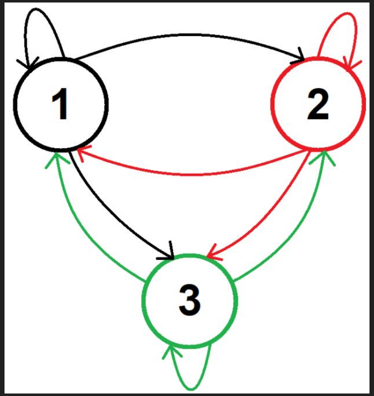
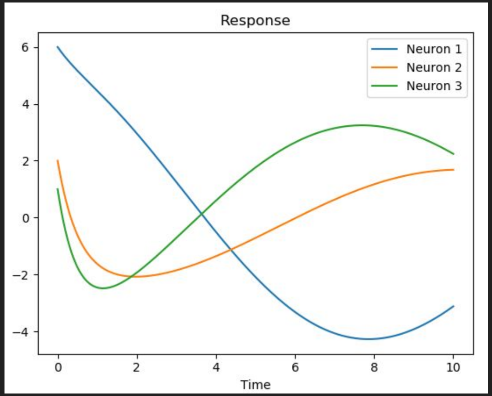
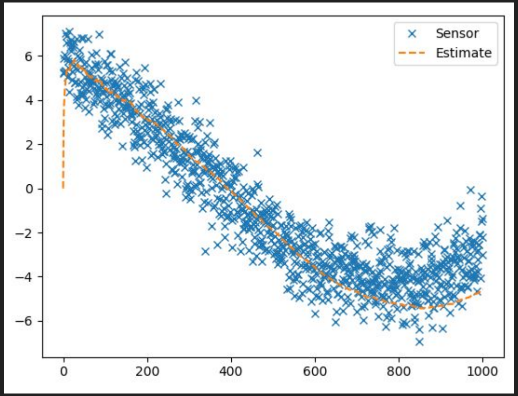
Attractors and Filtering

Josue Casco-Rodriguez

Overview





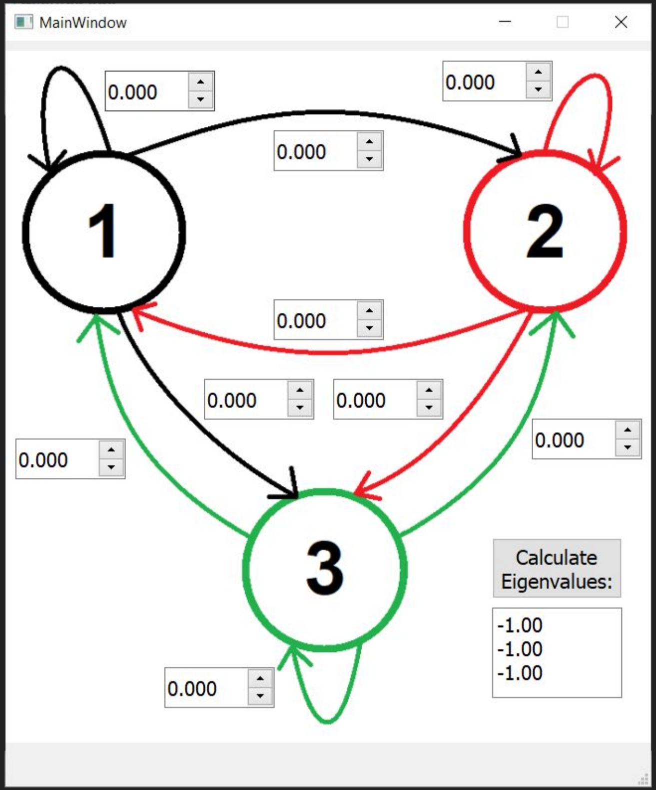


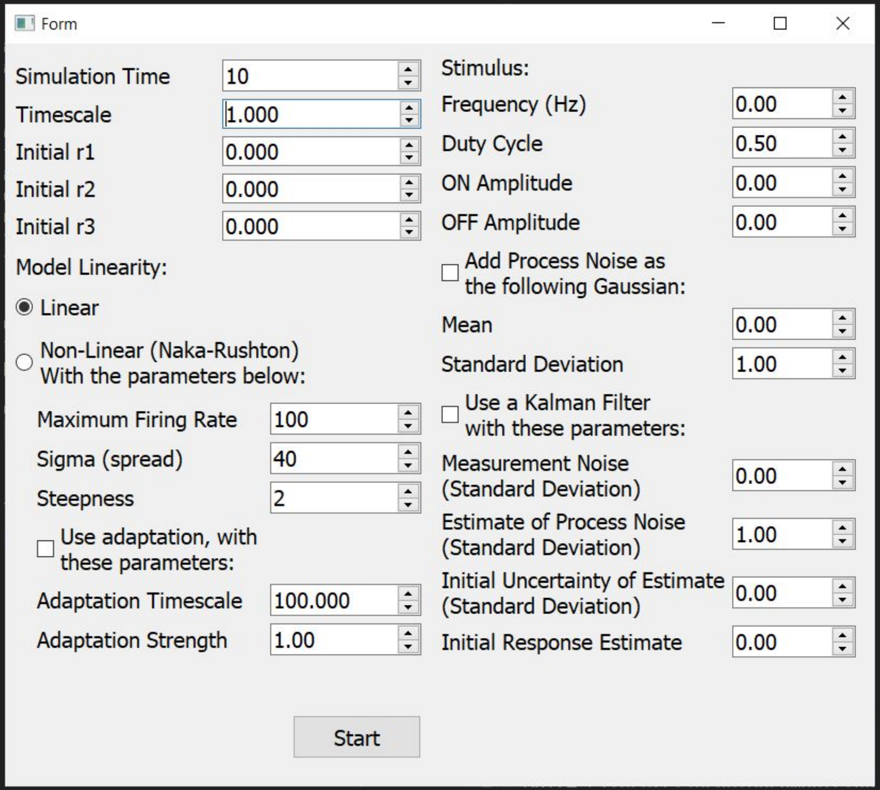
Network → output → noisy observations and Kalman Filter

Attractors (Background)

- $\tau * dr_1/dt = -r_1 + W_{1\to1} * r_1 + W_{2\to1} * r_2 + W_{3\to1} * r_3$
- $\tau * dr_2/dt = -r_2 + W_{1\to2} * r_1 + W_{2\to2} * r_2 + W_{3\to2} * r_3$
- $\tau * dr_3/dt = -r_3 + W_{1\to3} * r_3 + W_{2\to3} * r_3 + W_{3\to3} * r_3$
- This results in $\tau * d\underline{r}/dt = -\underline{r} + W*\underline{r}$.
- Thus, eigenvalues emerge, characterizing attractor's behavior.
  - $\lambda < 0 \to$ decay
  - $\lambda > 0 \to$ growth
  - λ complex $\to$ oscillation
- In general nonlinear case, $\tau * d\underline{r}/dt = -\underline{r} + f(W*\underline{r} + \underline{\text{stimulus}} + \underline{\text{noise}})$

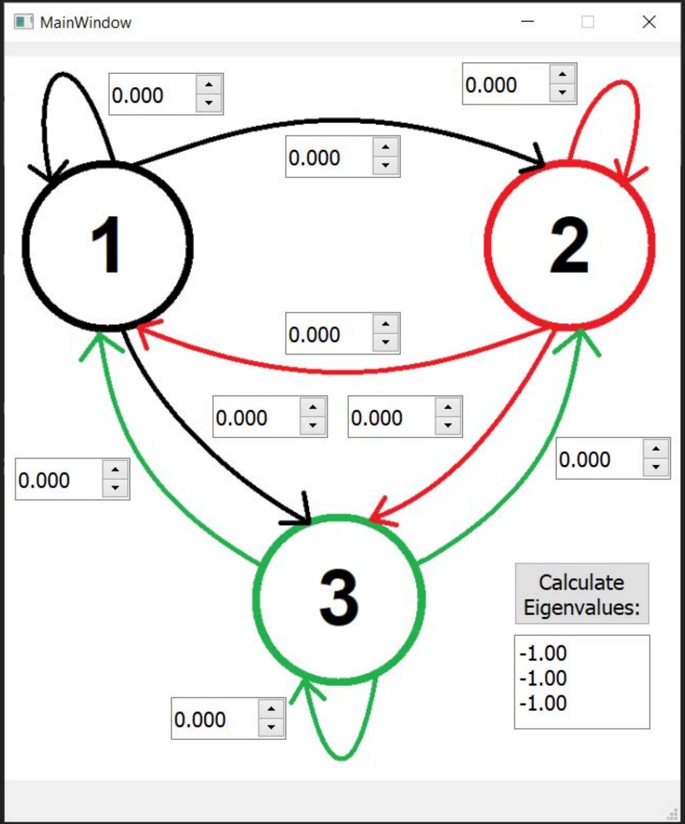
GUI (PyQt)


MainWindow
0.000
0.000
0.000
1
2
0.000
0.000
0.000
0.000
0.000
3
Calculate Eigenvalues:
-1.00
-1.00
-1.00
0.000


Form
Simulation Time 10
Timescale 1.000
Initial r1 0.000
Initial r2 0.000
Initial r3 0.000
Model Linearity:
Linear
Non-Linear (Naka-Rushton) With the parameters below:
Maximum Firing Rate 100
Sigma (spread) 40
Steepness 2
Use adaptation, with these parameters:
Adaptation Timescale 100.000
Adaptation Strength 1.00
Stimulus:
Frequency (Hz) 0.00
Duty Cycle 0.50
ON Amplitude 0.00
OFF Amplitude 0.00
Add Process Noise as the following Gaussian:
Mean 0.00
Standard Deviation 1.00
Use a Kalman Filter with these parameters:
Measurement Noise (Standard Deviation) 0.00
Estimate of Process Noise (Standard Deviation) 1.00
Initial Uncertainty of Estimate (Standard Deviation) 0.00
Initial Response Estimate 0.00
Start

Attractor Network

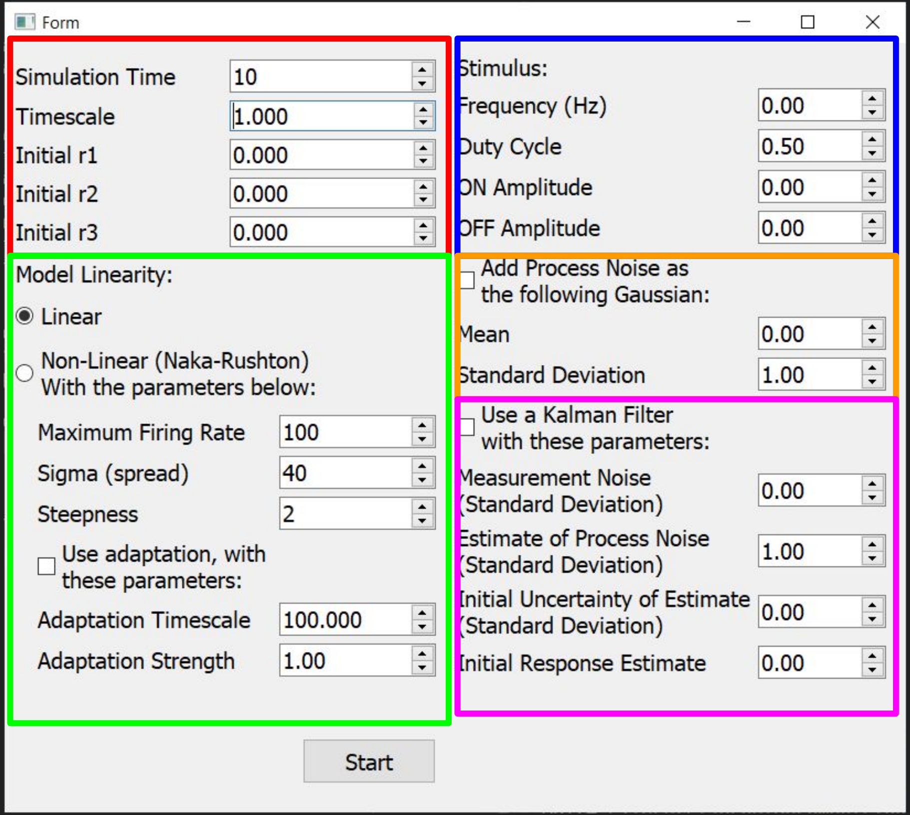
- Eigenvalues are only defined in the linear case.
- Inherent Decay
    - $\tau * d\underline{r}/dt = -\underline{r} + f(\underline{input})$
        - $f(x)$ = activation function
    - In linear case, $d\underline{r}/dt = -\underline{r} + W\underline{r}$
        - $\lambda = -1$ if $W = 0$



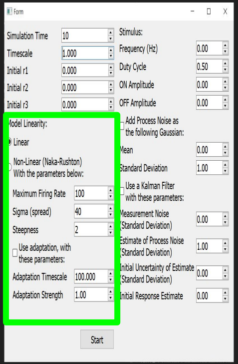
Parameters

- Red - Common parameters
    - Total time
    - Timescale
    - Initial response values
- Green - Linearity vs. Non-Linearity
- Blue - Stimulus (optional)
- Orange - Process noise (optional)
- Purple - Kalman Filter (optional)

Simulation is done using Euler's method with dt = 0.01.

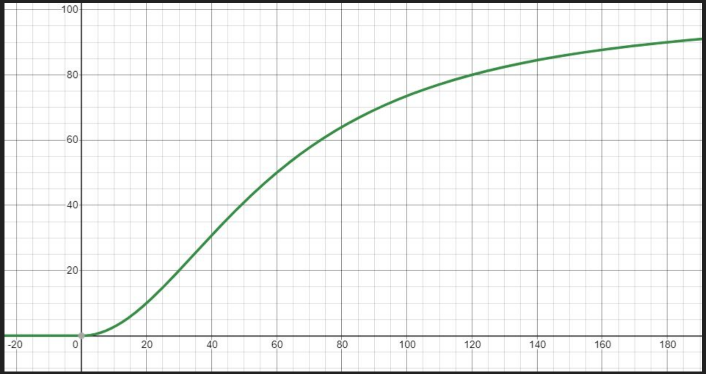


Activation Functions: Naka-Rushton Function

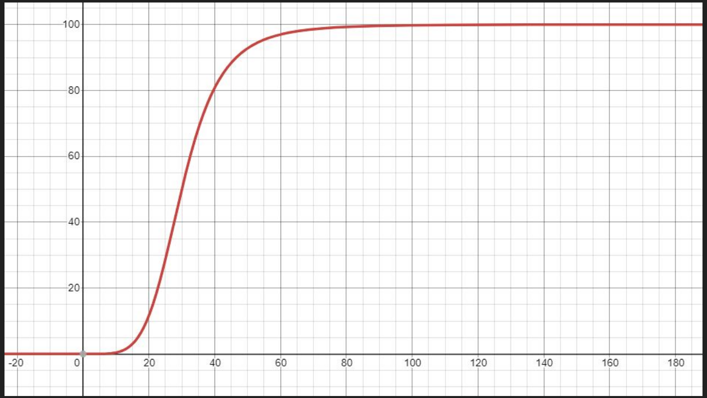


- Naka-Rushton Function
  - W = input
  - σ = semi-saturation constant (spread)
  - u = Maximum response
  - N = steepness

$$S(W) = \begin{cases} \dfrac{uW^N}{\sigma^N + W^N} & \text{for } W \geq 0 \\ 0 & \text{for } W < 0 \end{cases}$$



High spread, low steepness



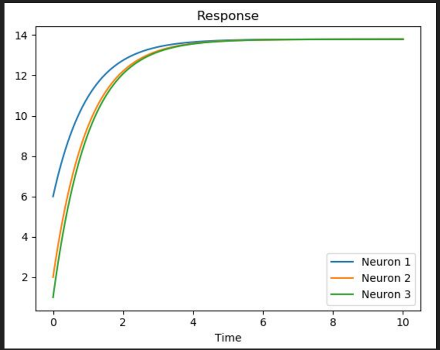
Low spread, high steepness

Activation Functions: Adaptation

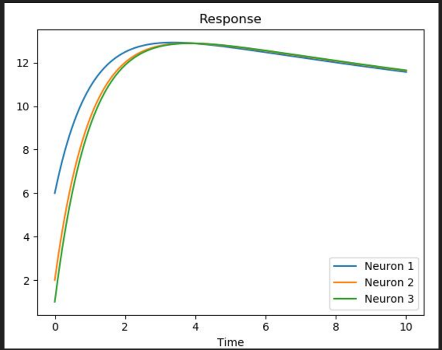
- Naka-Rushton Function
  - W = input
  - σ = semi-saturation constant (spread)
  - u = Maximum response
  - N = steepness
- Adaptation
  - A = adaptation factor
  - Adaptation strength = 0.7 in this case.

$$S(W) = \begin{cases} \frac{uW^N}{(\sigma+A)^N+W^N} & \text{for } W \geq 0 \\ 0 & \text{for } W < 0 \end{cases}$$

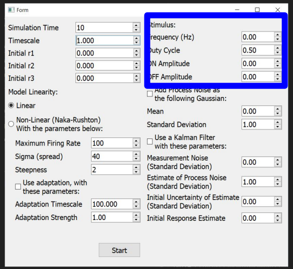
$$\frac{dA}{dt} = \frac{(-A + .7a)}{\tau_A}$$



Adaptation (Exaggerated)

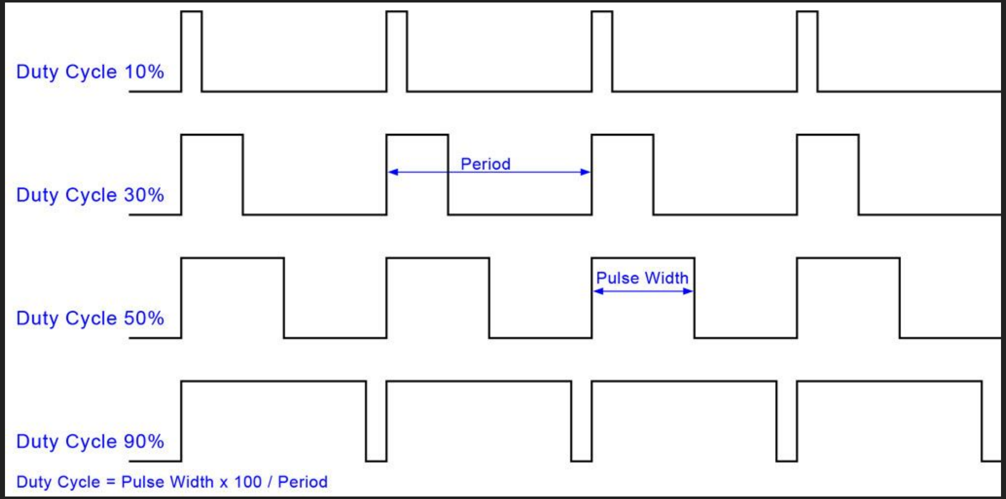


Stimulus (Pulse Wave)

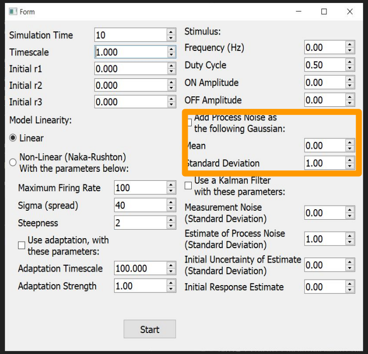


Pulse Wave can simulate:

- Square wave (duty cycle = 0.5)
- Pulse train (duty cycle ≤ 0.1)
- Constant input (frequency = 0)
- Monophasic/Biphasic input
    - Deep Brain Stimulation

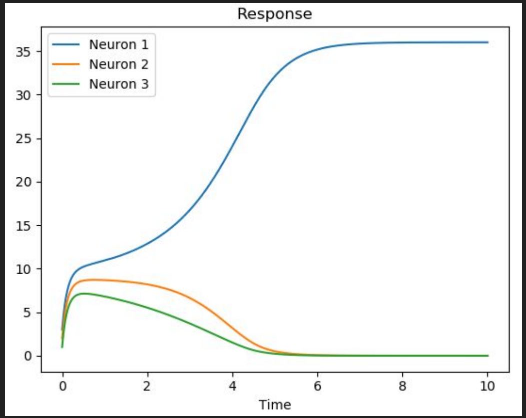


Process Noise

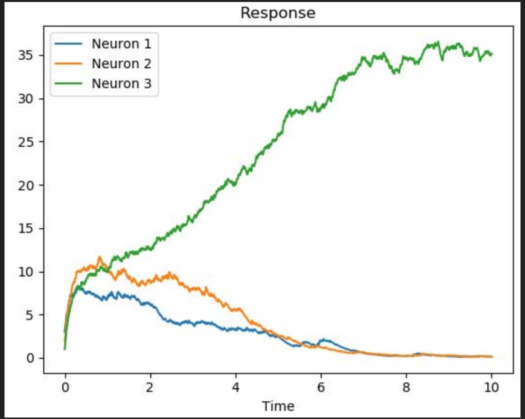


$\tau * d\underline{r}/dt = -\underline{r} + f(W\underline{r} + \underline{\text{process noise}})$

The noise is an adjustable Gaussian distribution.



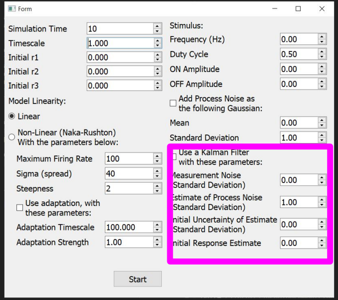
Heavy noise



Winner-take-all network with dominant Neuron 1

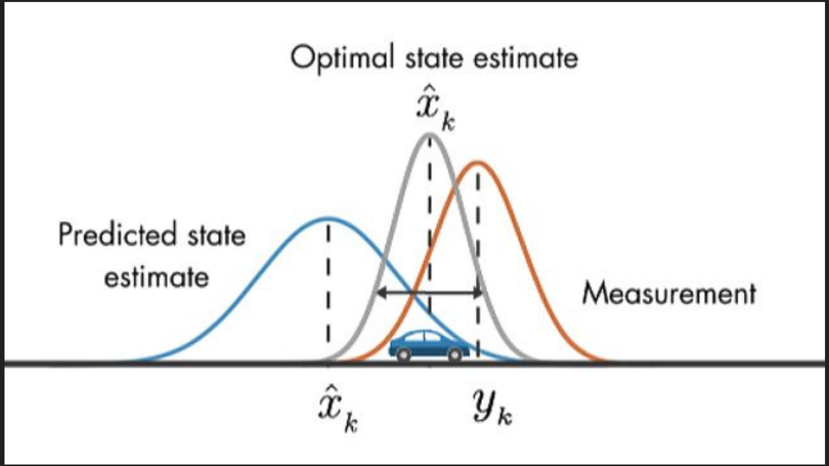
Same network, but Neuron 3 wins by chance

Kalman Filter



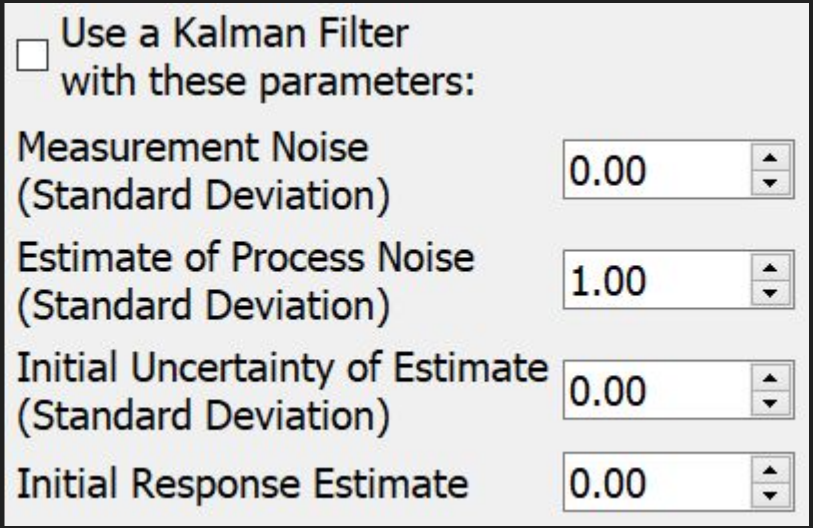
- Beliefs are represented as Gaussians.
    - Mean = estimated state
    - Variance = uncertainty

1. Predict
    a. Prediction = estimate$_{prior}$ + change$_{model}$
    b. Adding Gaussians increases variance
2. Update
    a. estimate$_{posterior}$ = prediction * observation
    b. Multiplying Gaussians decreases variance



Kalman Filter (parameters)

- Measurement noise (tuning variable)
  - $\underline{\text{observation}} = \underline{r} + \underline{\text{measurement noise}}$
  - Gaussian
- Process noise (tuning variable)
  - $\tau * d\underline{r}/dt = -\underline{r} + f(W\underline{r} + \underline{\text{process noise}})$
  - Gaussian
- Initial Parameters
  - Beliefs, including the initial one, are Gaussians
  - Mean = Initial Response Estimate
  - Variance = Initial Uncertainty



Kalman Filter (Technical Details)

- $\underline{x} = [r, dr/dt]$
- $\underline{z} = [r]$
- $P_{initial} = I_2 * \sigma^2_{initial}$
  - P = Uncertainty Covariance Matrix
- F = [[1 dt] [0 1]]
  - $r_{prediction} = r_{prior} + dt * (dr/dt)_{prior}$
  - $(dr/dt)_{prediction} = (dr/dt)_{prior}$
- B = [0 0]
  - No control
- H = [1 0]
  - Response (r) is noisily observed
  - Change (dr/dt) is a hidden variable
- $Q = E[\ \underline{noise_{process}} * \underline{noise_{process}}^T\ ]$
- $R = \sigma^2_{measurement}$

Predict Step

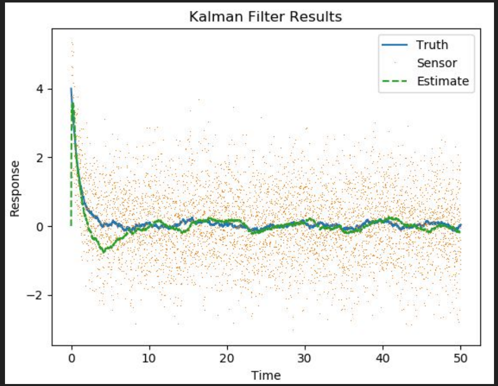
$$\bar{\mathbf{x}} = \mathbf{Fx} + \mathbf{Bu}$$
$$\bar{\mathbf{P}} = \mathbf{FPF}^\mathsf{T} + \mathbf{Q}$$

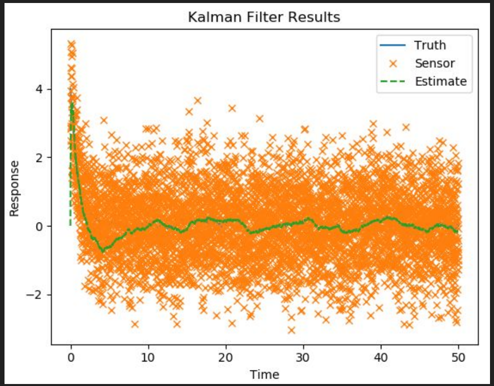
Update Step

$$\mathbf{S} = \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R}$$
$$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}$$
$$\mathbf{y} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}}$$
$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Ky}$$
$$\mathbf{P} = (\mathbf{I} - \mathbf{KH})\bar{\mathbf{P}}$$

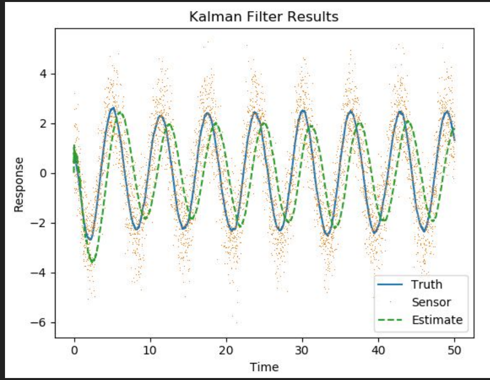
Performance (Linear)

Current iteration of Kalman Filter does well with constantly-changing values, but struggles with oscillations or sharp changes.

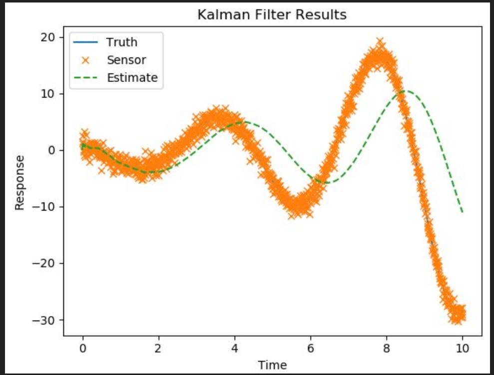




Exponential Decay



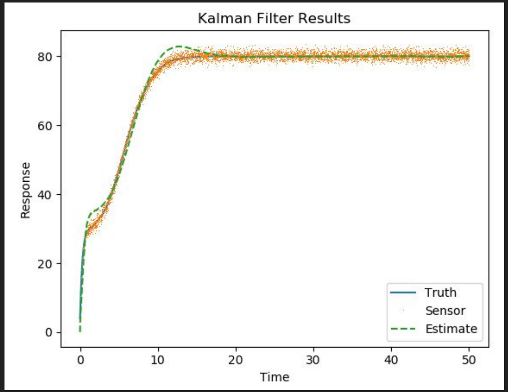
Constant Oscillation



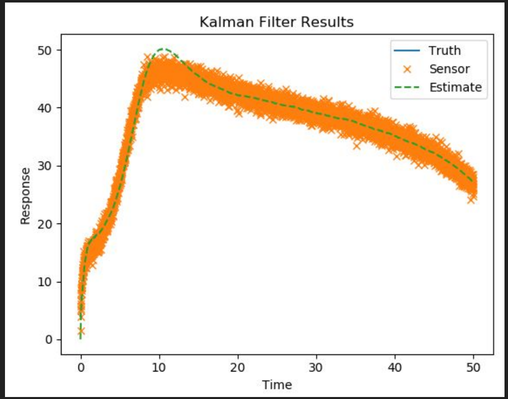
Unstable Oscillation

Performance (Continued)

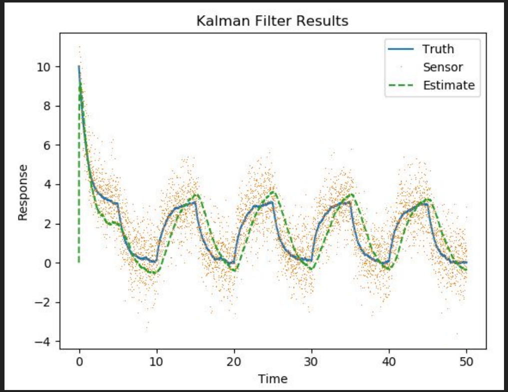
Even in nonlinear or unknown-stimuli cases, the current Kalman Filter does well with constantly changing values but struggles with sharp changes.



Winner-take-all



Winner-take-all with adaptation



Linear decay w/ stimulus

Potential Future Work

- Unscented (Nonlinear) Kalman Filter
- More detailed state transition matrix (F)?
- More states? x'', x''', etc.
- Control (non-zero B)

Sources

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4930057/

https://nbviewer.jupyter.org/github/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/table_of_contents.ipynb

https://www.simbrain.net/Documentation/docs/Pages/Network/neuron/Naka-Rushton.html

Code: https://github.com/JosueImet/Neuron-GUI-2021