DSP Evaluation Project

Youngjae Kim

Monostatic Pulse Radar

- Mono-static: the transmitter and receiver are collocated
- Pulse: short and powerful pulses followed by the silent period during which the echo signals are received
- Metrics for performance of non-coherent detection
  - Probability of False Alarm
  - Probability of Missed Detection = 1 - Probability of Detection

Received Power Computation

- From Friis transmission equation,

$$P_r = \frac{P_t G_t}{4\pi r^2}\sigma\frac{1}{4\pi r^2}A_{\text{eff}}$$

where

- P_t = transmitter's input power (watts)
- G_t = gain of the radar transmit antenna (dimensionless)
- r = distance from the radar to the target (meters)
- σ = radar cross-section of the target (meters squared)
- A_{eff} = effective area of the radar receiving antenna (meters squared)
- P_r = power received back from the target by the radar (watts)

- Assuming lossless isotropic Rx antenna,

$$A_{eff} = \frac{\lambda^2}{4\pi}$$

Noise Power Consideration

- Thermal noise (in dBm) = -174 + 10*log10(Fs)
  - Fs is the sampling frequency
  - Pulse BW is selected as Fs, which is Nyquist rate.
  - Pulse BW = 1/pulse width = c/(2*range_resolution)
    - c is a light speed
    - required range resolution is given as 50m

Readme for Matlab file

- Run “radar.m”
- The following parameters are hard-coded
  - Fc = 100MHz
  - Tx Power = 5kW
  - 10 repetitions of pulses
  - detection threshold = 0.2e-13
- It tries a single pulse 100000 times for PFA and PD computation

```
>> radar
At SNR = 4.82 (dB)
Probability of Detection = 0.98.....PASSED
Probability of False Alarm = 0.00.....PASSED
```

Gaussian Noise Generator

- Basically, it is an inverse CDF-based generator
  - Pick a random number between [0,1)
  - Apply ICDF to generate the corresponding noise sample
  - When applying ICDF, use 2nd-order polynomial approximation

Implementation Details

- Since ICDF is symmetric, only the positive part needs to be generated. One sign bit determines if we need to negate the result at the end
- Segment generation
  - A total of 128 segments(=7bits) are generated (32 from LZD and 4 from OFFSET)
  - Leading zero detector is for making $1/2^5$ probability segment
  - Further refine each segment to 4 equal pieces
- Generated segments and polynomial coefficients are stored in the lookup tables

Readme for Matlab file

- Run “[grn_val, test_vec] = GNG”
- Gaussian Noise Sample with zero-mean and unit-variance will be generated

```
>> [grn_val, test_vec] = GNG

grn_val =

    1.5341


test_vec =

  struct with fields:

     URNG_OUT: [1×64 double]
         SIGN: 0
       OFFSET: [1 0]
    LZD_INPUT: [1×61 double]
      SEGMENT: [1 0 1 1 0 0 0]
      MASK_IN: [0 1 1 1 1 0 0 1 1 0 1 1 1 0 0]
     MASK_OUT: [0 1 1 1 1 0 0 1 1 0 1 1 1 0 0]
      MULT_IN: [0 0 1 1 1 0 1 1 0 0 1 1 1 1 0]
        COEF0: [0 0 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 0 0]
        COEF1: [0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0]
        COEF2: [0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0]
       INTERM: [0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0]
      POLYOUT: [0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0]
       GRNOUT: [0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0]
```

Kalman Filter Basics

- It consists of three stages
  - Projection
  - Kalman Gain Computation
  - New estimation and update covariance matrix

Description	Equation
Kalman Gain	$K_k = P_k' H^T \left(H P_k' H^T + R \right)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}_k' + K_k \left(z_k - H \hat{x}_k' \right)$
Update Covariance	$P_k = \left(I - K_k H \right) P_k'$
Project into $k+1$	$\hat{x}_{k+1}' = \Phi \hat{x}_k$ $P_{k+1} = \Phi P_k \Phi^T + Q$

Extended KF

- Extended KF is KF's extension to a non-linear system by taking the first order linear approximation to get matrix Φ. or H.

$$\tilde{x} = f(\hat{x}, u) \qquad \tilde{y} = h(\tilde{x}, u)$$

I) $$F = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}} \qquad H = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}}$$

II) $$\tilde{P} = F\hat{P}F^T + Q$$

III) $$K = \tilde{P}H^T\left[H\tilde{P}H^T + R\right]^{-1}$$

IV) $$\hat{x} = \tilde{x} + K(y - \tilde{y})$$

V) $$\hat{P} = \tilde{P} - KH\tilde{P}$$

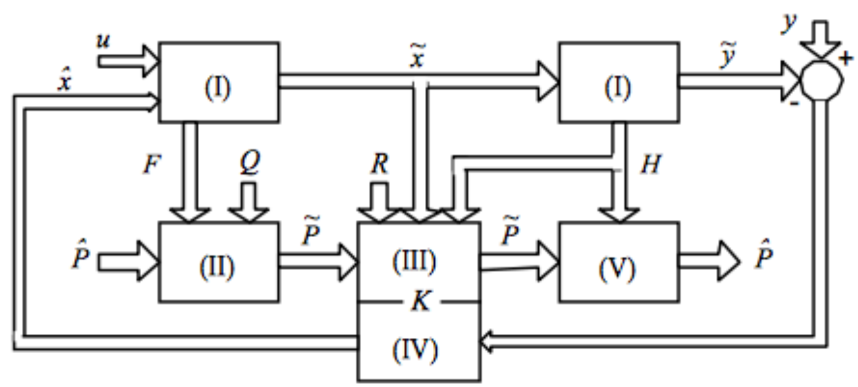

Fig. 1. EKF recursive computational scheme.

Rotor with Kalman Filter Representation

- Substituting the following variables to the previous equations yield the results on the next slide

$$\boldsymbol{x}_k = [cos\,\theta_k \quad sen\,\theta_k \quad \omega_k]^T \qquad \boldsymbol{y}_k = [cos\,\theta_k \quad sen\,\theta_k]^T \qquad \boldsymbol{u}_k = [0 \quad 0]^T \tag{5}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \mathrm{k}) = \begin{pmatrix} x_1 \cdot cos(\omega_b T_{sc} \cdot x_3) - x_2 \cdot sin(\omega_b \mathrm{T}_{sc} \cdot x_3) \\ x_1 \cdot sin(\omega_b \mathrm{T}_{sc} \cdot x_3) + x_2 \cdot cos(\omega_b \mathrm{T}_{sc} \cdot x_3) \\ x_3 \end{pmatrix} \qquad \boldsymbol{y}_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \boldsymbol{x}_k \tag{6}$$

$$\boldsymbol{Q} = q\boldsymbol{I}_{3x3} \qquad \boldsymbol{R} = r\boldsymbol{I}_{2x2} \tag{7}$$

Readme for Matlab file

- Run “rotor”