Fusion of Engineering, Control, Coding, Machine Learning, and Science

Aleksandar Haber

UNCATEGORIZED

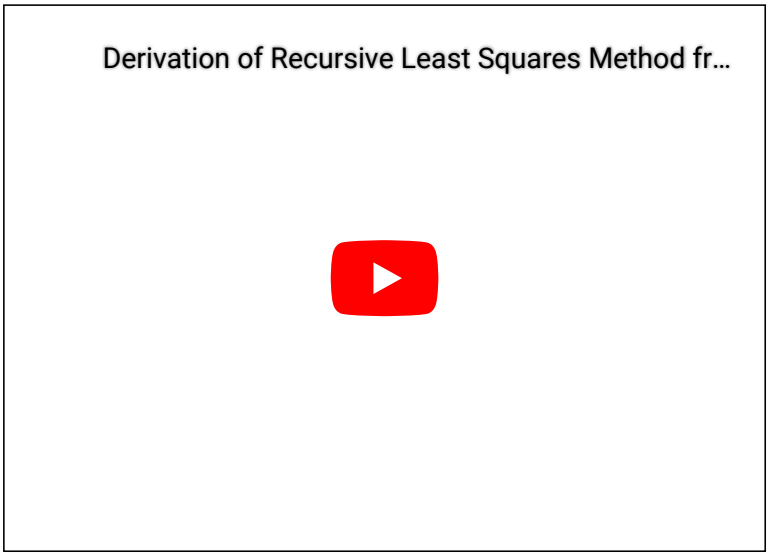
Introduction to Kalman Filter: Derivation of the Recursive Least Squares Method

by admin October 23, 2022

In this post, we derive equations describing the recursive least squares method. The motivation for creating this tutorial, comes from the fact that these equations are used in the derivation of the Kalman filter equations. Apart from this, the recursive least squares method is the basis of many other algorithms, such as adaptive control algorithms. Consequently, it is of paramount importance to properly understand the recursive least-squares method. The derivation of the Kalman filter equations on the basis of the recursive least-squares equations is arguably much simpler and easier to understand than the derivation based on other methods or approaches.

The Python implementation of the derived least-squares method is given here.

The YouTube videos accompanying this post are given below.



Motivational Example

To motivate the need for the recursive least squares method, we consider the following example. We assume that a vehicle is moving from a starting position y_0, with an initial velocity, denoted by v_0, and with an acceleration denoted by a. Our goal is to estimate, y_0, v_0, and a, by only measuring position values $y(t)$ obtained by a distance sensor. Here, t

denotes the time variable. From the basic kinematics equations that relate the position, velocity, and acceleration, we know that the following model fully kinematically describes the motion of the car

$$y(t) = y_0 + v_0 t + \frac{1}{2} a t^2 \tag{1}$$

We assume that we are able to measure the position at the discrete-time instants $t = 0, \Delta t, 2\Delta t, 3\Delta t, \ldots$, where Δt is the sampling period of the sensor. Next, let $y(k\Delta t) := y_k$. Our measurement equation is then

$$y_k = \begin{bmatrix} 1 & k\Delta t & \frac{1}{2}(k\Delta t)^2 \end{bmatrix} \begin{bmatrix} y_0 \\ v_0 \\ a \end{bmatrix} + v_k \tag{2}$$

where v_k is the measurement noise. Clearly, our C_k matrix is then

$$C_k = \begin{bmatrix} 1 & k\Delta t & \frac{1}{2}(k\Delta t)^2 \end{bmatrix} \tag{3}$$

And our vector $\mathbf{x}$ is then

$$\mathbf{x} = \begin{bmatrix} y_0 \\ v_0 \\ a \end{bmatrix} \tag{4}$$

The goal of the recursive least squares method is to estimate the vector $\mathbf{x}$ recursively on the basis of the sequentially obtained data $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \ldots$. That is, we start from some initial guess of $\mathbf{x}$, denoted by $\mathbf{x}_0$. When the initial measurement sample $\mathbf{y}_0$ arrives, we should update the initial estimate. Then, when the next sample $\mathbf{y}_1$ arrives, we should use this sample to update the previously computed estimate. This procedure is repeated every time a new measurement sample arrives.

Issues with the ordinary least squares method

In our previous post which can be found here, we derived the solution to the least squares problem. Here for completeness of this tutorial, we briefly revise the ordinary least squares method.

Consider the following system of linear equations

$$y_i = \sum_{j=1}^{n} C_{i,j} x_j + v_i, \quad i = 1, 2, \ldots, s \tag{5}$$

where $x_j, j = 1, 2, \ldots, n$ are real scalar variables that we want to determine (unknowns), $C_{i,j} \in \mathbb{R}$ are scalars and $v_i \in \mathbb{R}$ is the measurement noise. The noise term is not known. However, we assume that we know the statistical

properties of the noise that can be estimated from the data. We assume that the measurement noise is zero mean and that

$$E[v_i^2] = \sigma_i^2 \tag{6}$$

where $E[\cdot]$ is an expectation operator and σ_i^2 is a variance (σ_i is a standard deviation) . Furthermore, we assume that v_i and v_j are statistically independent for all $i \neq j$.

The equations in (5) can be compactly written as follows

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_s \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \dots & \vdots \\ C_{s1} & C_{s2} & \dots & C_{sn} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_s \end{bmatrix}}_{\mathbf{v}} \tag{7}$$

where $\mathbf{y}, \mathbf{v} \in \mathbb{R}^s$ and $C \in \mathbb{R}^{s \times n}$. In (7) and throughout this post, vector quantities are denoted using a bold font. The last equation can be compactly written as follows:

$$\mathbf{y} = C\mathbf{x} + \mathbf{v} \tag{8}$$

Usually, in the least squares estimation problems $n \ll s$, that is, the number of measurements s is usually much larger than the number of variables to be estimated. Under the assumption that the matrix C has a full column rank and under the assumption that $s \geq n$, the least squares solution is found by solving the following optimization problem

$$\min_{\mathbf{x}} \|\mathbf{y} - C\mathbf{x}\|_2^2 \tag{9}$$

There are a number of computationally efficient approaches for solving (9). Here for brevity and simplicity, we use the solution that is not the most efficient one to be computed. This solution is given by the following equation

$$\hat{\mathbf{x}} = (C^T C)^{-1} C^T \mathbf{y} \tag{10}$$

There are at least two issues with the formulation and solution of the least squares problem:

1. The least-squares problem formulation and its solutions assume that all measurements are available at a certain time instant. However, this often might not be the case in the practice. Also, we would like to update the solution as new measurements arrive in order to have an adaptive approach for updating the solution.
2. When the number of measurements s is extremely large, the solutions of the least squares problem are difficult to compute. That is, the computational and memory complexities of the solution are very high.

These facts motivate the development of the recursive least squares that is presented in the sequel.

Recursive Least Squares Method – Complete Derivation From Scratch

Let us assume that at the discrete-time instant k, the set of measurements grouped in the vector $\mathbf{y}_k \in \mathbb{R}^l$ becomes available. Let us assume that these measurements are related to the vector $\mathbf{x}$ that we want to estimate by the following equation

$$\mathbf{y}_k = C_k \mathbf{x} + \mathbf{v}_k \tag{11}$$

where $C_k \in \mathbb{R}^{l \times n}$, and $\mathbf{v}_k \in \mathbb{R}^l$ is the measurement noise vector. We assume that the covariance of the measurement noise is given by

$$E[\mathbf{v}_k \mathbf{v}_k^T] = R_k \tag{12}$$

and we assume that the mean of $\mathbf{v}_k$ is zero:

$$E[\mathbf{v}_k] = 0 \tag{13}$$

The recursive least-squares method that we derive in this section, has the following form:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + K_k(\mathbf{y}_k - C_k \hat{\mathbf{x}}_{k-1}) \tag{14}$$

where $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_{k-1}$ are the estimates of the vector $\mathbf{x}$ at the discrete-time instants k and $k-1$, and $K_k \in \mathbb{R}^{n \times l}$ is the gain matrix that we need to determine.

The equation (14) updates the estimate of $\mathbf{x}$ at the time instant k on the basis of the estimate $\hat{\mathbf{x}}_{k-1}$ at the previous time instant $k-1$ and on the basis of the measurement $\mathbf{y}_k$ obtained at the time instant k, as well as on the basis of the gain matrix K_k computed at the time instant k.

Our goal is to derive the formulas for computing the gain matrix K_k.

First, we need to introduce new notation. We assume that the vector $\hat{\mathbf{x}}_k$ is partitioned as follows

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{x}_{1,k} \\ \hat{x}_{2,k} \\ \vdots \\ \hat{x}_{n,k} \end{bmatrix} \tag{15}$$

This partitioning directly corresponds to the partitioning of the vector $\mathbf{x}$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{16}$$

Next, we introduce **the estimation error vector** ε_k and the following partitioning:

$$\boldsymbol{\varepsilon}_k = \begin{bmatrix} \varepsilon_{1,k} \\ \varepsilon_{2,k} \\ \vdots \\ \varepsilon_{n,k} \end{bmatrix} = \mathbf{x} - \hat{\mathbf{x}}_k = \begin{bmatrix} x_1 - \hat{x}_{1,k} \\ x_2 - \hat{x}_{2,k} \\ \vdots \\ x_n - \hat{x}_{n,k} \end{bmatrix} \tag{17}$$

That is,

$$\varepsilon_{i,k} = x_i - \hat{x}_{i,k}, \quad i = 1, 2, \ldots, n \tag{18}$$

The gain K_k is computed by minimizing the sum of variances of the estimation errors

$$W_k = E[\varepsilon_{1,k}^2] + E[\varepsilon_{2,k}^2] + \ldots + E[\varepsilon_{n,k}^2] \tag{19}$$

Next, we show that this cost function, can be represented as follows:

$$W_k = \mathrm{tr}(P_k) \tag{20}$$

where $\mathrm{tr}(\cdot)$ is the notation for the matrix trace, and P_k **is the estimation error covariance matrix that is defined by**

$$P_k = E[\boldsymbol{\varepsilon}_k \boldsymbol{\varepsilon}_k^T] \tag{21}$$

That is, **the gain matrix K_k is found by minimizing the trace of the estimation error covariance matrix.**

To show this, let us first expand the following expression

$$\begin{aligned} \boldsymbol{\varepsilon}_k \boldsymbol{\varepsilon}_k^T &= \begin{bmatrix} \varepsilon_{1,k} \\ \varepsilon_{2,k} \\ \vdots \\ \varepsilon_{n,k} \end{bmatrix} \begin{bmatrix} \varepsilon_{1,k} & \varepsilon_{2,k} & \ldots & \varepsilon_{n,k} \end{bmatrix} \\ &= \begin{bmatrix} \varepsilon_{1,k}^2 & \varepsilon_{1,k}\varepsilon_{2,k} & \varepsilon_{1,k}\varepsilon_{3,k} & \ldots & \varepsilon_{1,k}\varepsilon_{n,k} \\ \varepsilon_{1,k}\varepsilon_{2,k} & \varepsilon_{2,k}^2 & \varepsilon_{2,k}\varepsilon_{3,k} & \ldots & \varepsilon_{2,k}\varepsilon_{n,k} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \varepsilon_{1,k}\varepsilon_{n,k} & \varepsilon_{2,k}\varepsilon_{n,k} & \varepsilon_{3,k}\varepsilon_{n,k} & \ldots & \varepsilon_{n,k}^2 \end{bmatrix} \end{aligned} \tag{22}$$

From the last expression, we have

$$\begin{aligned} P_k &= E[\boldsymbol{\varepsilon}_k \boldsymbol{\varepsilon}_k^T] \\ &= \begin{bmatrix} E[\varepsilon_{1,k}^2] & E[\varepsilon_{1,k}\varepsilon_{2,k}] & E[\varepsilon_{1,k}\varepsilon_{3,k}] & \ldots & E[\varepsilon_{1,k}\varepsilon_{n,k}] \\ E[\varepsilon_{1,k}\varepsilon_{2,k}] & E[\varepsilon_{2,k}^2] & E[\varepsilon_{2,k}\varepsilon_{3,k}] & \ldots & E[\varepsilon_{2,k}\varepsilon_{n,k}] \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ E[\varepsilon_{1,k}\varepsilon_{n,k}] & E[\varepsilon_{2,k}\varepsilon_{n,k}] & E[\varepsilon_{3,k}\varepsilon_{n,k}] & \ldots & E[\varepsilon_{n,k}^2] \end{bmatrix} \end{aligned} \tag{23}$$

Next, by taking the matrix trace of the last expression, we obtain

$$\mathrm{tr}(P_k) = E[\varepsilon_{1,k}^2] + E[\varepsilon_{2,k}^2] + \ldots + E[\varepsilon_{n,k}^2] \tag{24}$$

and this is exactly the expression for the cost function W_k given by (19).

Next, we derive the expression for the time propagation of the estimation covariance matrix. By substituting the estimate propagation equation (14) and the measurement equation (11) in the expression for the estimation error, we obtain

$$\begin{aligned}\boldsymbol{\varepsilon}_k &= \mathbf{x} - \hat{\mathbf{x}}_k \\ &= \mathbf{x} - \hat{\mathbf{x}}_{k-1} - K_k(\mathbf{y}_k - C_k\hat{\mathbf{x}}_{k-1}) \\ &= \mathbf{x} - \hat{\mathbf{x}}_{k-1} - K_k(C_k\mathbf{x} + \mathbf{v}_k - C_k\hat{\mathbf{x}}_{k-1}) \\ &= (I - K_kC_k)(\mathbf{x} - \hat{\mathbf{x}}_{k-1}) - K_k\mathbf{v}_k \\ &= (I - K_kC_k)\boldsymbol{\varepsilon}_{k-1} - K_k\mathbf{v}_k\end{aligned} \tag{25}$$

Next, we have

$$\begin{aligned}\boldsymbol{\varepsilon}_k\boldsymbol{\varepsilon}_k^T &= ((I - K_kC_k)\boldsymbol{\varepsilon}_{k-1} - K_k\mathbf{v}_k)((I - K_kC_k)\boldsymbol{\varepsilon}_{k-1} - K_k\mathbf{v}_k)^T \\ &= ((I - K_kC_k)\boldsymbol{\varepsilon}_{k-1} - K_k\mathbf{v}_k)(\boldsymbol{\varepsilon}_{k-1}^T(I - K_kC_k)^T - \mathbf{v}_k^TK_k^T) \\ &= (I - K_kC_k)\boldsymbol{\varepsilon}_{k-1}\boldsymbol{\varepsilon}_{k-1}^T(I - K_kC_k)^T - (I - K_kC_k)\boldsymbol{\varepsilon}_{k-1}^T\mathbf{v}_k^TK_k^T \\ &- K_k\mathbf{v}_k\boldsymbol{\varepsilon}_{k-1}^T(I - K_kC_k)^T + K_k\mathbf{v}_k\mathbf{v}_k^TK_k^T\end{aligned} \tag{26}$$

Applying the expectation operator to (26), we obtain

$$\begin{aligned}P_k &= E[\boldsymbol{\varepsilon}_k\boldsymbol{\varepsilon}_k^T] \\ &= (I - K_kC_k)E[\boldsymbol{\varepsilon}_{k-1}\boldsymbol{\varepsilon}_{k-1}^T](I - K_kC_k)^T - (I - K_kC_k)E[\boldsymbol{\varepsilon}_{k-1}\mathbf{v}_k^T]K_k^T \\ &- K_kE[\mathbf{v}_k\boldsymbol{\varepsilon}_{k-1}^T](I - K_kC_k)^T + K_kE[\mathbf{v}_k\mathbf{v}_k^T]K_k^T\end{aligned} \tag{27}$$

We have

$$P_{k-1} = E[\boldsymbol{\varepsilon}_{k-1}\boldsymbol{\varepsilon}_{k-1}^T] \tag{28}$$

Then, since $\mathbf{v}_k$ and $\boldsymbol{\varepsilon}_{k-1}$ are independent (measurement noise at the time instant k is independent from the estimation error at the time instant $k-1$), and due to the fact that $\mathbf{v}_k$ is zero-mean:

$$E[\boldsymbol{\varepsilon}_{k-1}\mathbf{v}_k^T] = E[\boldsymbol{\varepsilon}_{k-1}]E[\mathbf{v}_k^T] = 0 \tag{29}$$

$$E[\mathbf{v}_k\boldsymbol{\varepsilon}_{k-1}^T] = E[\mathbf{v}_k]E[\boldsymbol{\varepsilon}_{k-1}^T] = 0 \tag{30}$$

$$E[\mathbf{v}_k\mathbf{v}_k^T] = R_k \tag{31}$$

By substituting (28), (29), (30), and (31) in (27), we obtain **an important expression for the propagation of the estimation error covariance matrix**

$$P_k = (I - K_kC_k)P_{k-1}(I - K_kC_k)^T + K_kR_kK_k^T \tag{32}$$

Let us expand the last expression

$$P_k = P_{k-1} - P_{k-1}C_k^T K_k^T - K_k C_k P_{k-1} + K_k C_k P_{k-1} C_k^T K_k^T + K_k R_k K_k^T \tag{33}$$

By substituting (49) in (20), we obtain

$$\begin{aligned} W_k = &\operatorname{tr}(P_{k-1}) - \operatorname{tr}(P_{k-1}C_k^T K_k^T) \\ &- \operatorname{tr}(K_k C_k P_{k-1}) + \operatorname{tr}(K_k C_k P_{k-1} C_k^T K_k^T) + \operatorname{tr}(K_k R_k K_k^T) \end{aligned} \tag{34}$$

We find the gain matrix K_k by optimizing the cost function (34). That is, we find the value of K_k by solving the following equation

$$\frac{\partial W_k}{\partial K_k} = 0 \tag{35}$$

To compute the derivatives of matrix traces, we use the formulas given in The Matrix Cookbook (Section 2.5). We use the following formulas

$$\frac{\partial \operatorname{tr}(AX^T B)}{\partial X} = BA \tag{36}$$

$$\frac{\partial \operatorname{tr}(AXB)}{\partial X} = A^T B^T \tag{37}$$

$$\frac{\partial \operatorname{tr}(XBX^T)}{\partial X} = XB^T + XB \tag{38}$$

We find derivatives of all terms in (34). Since P_{k-1} is constant, we have

$$\frac{\partial \operatorname{tr}(P_{k-1})}{\partial K_k} = 0 \tag{39}$$

Next, by using (36), we have

$$\frac{\partial \operatorname{tr}(P_{k-1}C_k^T K_k^T)}{\partial K_k} = P_{k-1}C_k^T \tag{40}$$

By using (37), we have

$$\frac{\partial \operatorname{tr}(K_k C_k P_{k-1})}{\partial K_k} = P_{k-1}C_k^T \tag{41}$$

By using (38), we have

$$\frac{\partial \operatorname{tr}(K_k C_k P_{k-1} C_k^T K_k^T)}{\partial K_k} = 2K_k C_k P_{k-1} C_k^T \tag{42}$$

$$\frac{\partial \operatorname{tr}(K_k R_k K_k^T)}{\partial K_k} = 2K_k R_k \tag{43}$$

By substituting (39), (40), (41), (42), and (43):

$$-2P_{k-1}C_k^T + 2K_kC_kP_{k-1}C_k^T + 2K_kR_k = 0 \tag{44}$$

From the last equation, we have

$$K_k(R_k + C_kP_{k-1}C_k^T) = P_{k-1}C_k^T \tag{45}$$

By solving the last equation we obtain the final expression for the gain matrix K_k

$$K_k = P_{k-1}C_k^T(R_k + C_kP_{k-1}C_k^T)^{-1} \tag{46}$$

Before we summarize the recursive least-squares method, we derive an alternative form of the estimation error covariance matrix propagation equation. First, we can write the gain matrix (46), as follows

$$K_k = P_{k-1}C_k^TL_k^{-1}, \quad L_k = R_k + C_kP_{k-1}C_k^T \tag{47}$$

Here, we should observe that the matrix L_k and its inverse L_k^{-1} are symmetric, that is

$$L_k = L_k^T, \;\; (L_k^{-1})^T = L_k^{-1} \tag{48}$$

We substitute (47) in (49), and as the result, we obtain

$$\begin{aligned} P_k = {} & P_{k-1} - P_{k-1}C_k^TL_k^{-1}C_kP_{k-1} - P_{k-1}C_k^TL_k^{-1}C_kP_{k-1} \\ & + P_{k-1}C_k^TL_k^{-1}C_kP_{k-1}C_k^TL_k^{-1}C_kP_{k-1} + P_{k-1}C_k^TL_k^{-1}R_kL_k^{-1}C_kP_{k-1} \end{aligned} \tag{49}$$

By adding the second and third terms and by grouping the last two terms, we obtain

$$\begin{aligned} P_k &= P_{k-1} - 2P_{k-1}C_k^TL_k^{-1}C_kP_{k-1} \\ &\quad + P_{k-1}C_k^TL_k^{-1}\underbrace{(C_kP_{k-1}C_k^T + R_k)}_{L_k}L_k^{-1}C_kP_{k-1} \\ &= P_{k-1} - 2P_{k-1}C_k^TL_k^{-1}C_kP_{k-1} + P_{k-1}C_k^TL_k^{-1}C_kP_{k-1} \\ &= P_{k-1} - \underbrace{P_{k-1}C_k^TL_k^{-1}}_{K_k}C_kP_{k-1} \\ &= P_{k-1} - K_kC_kP_{k-1} \\ &= (I - K_kC_k)P_{k-1} \end{aligned} \tag{50}$$

That is, an alternative expression for the propagation of the estimation error covariance matrix is given by

$$P_k = (I - K_kC_k)P_{k-1} \tag{51}$$

Consequently, the recursive least squares method consists of the following three equations

1. Gain matrix update

$$K_k = P_{k-1}C_k^T(R_k + C_kP_{k-1}C_k^T)^{-1} \tag{52}$$

2. Estimate update

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + K_k(\mathbf{y}_k - C_k\hat{\mathbf{x}}_{k-1}) \tag{53}$$

3. Propagation of the estimation error covariance matrix by using this equation

$$P_k = (I - K_kC_k)P_{k-1}(I - K_kC_k)^T + K_kR_kK_k^T \tag{54}$$

or this equation

$$P_k = (I - K_kC_k)P_{k-1} \tag{55}$$

If no apriori knowledge of the estimate is known, then this method is initialized with the estimation error covariance matrix that is equal to $P_0 = cI$, where c is a large number, and with a random value of $\mathbf{x}_0$.

The Python implementation of the derived least-squares method is given here.

PREVIOUS POST

Solve Constrained Optimization Problems in Python by Using SciPy Library, minimize() Function and Trust-Region Constrained Method

NEXT POST

Introduction to Kalman Filter: Disciplined Python Implementation of Recursive Least Squares Method

admin

View all posts by admin →

YOU MIGHT ALSO LIKE

Copyright Notice and License for Using Codes

October 1, 2023

Introduction to MATLAB Control System Toolbox – Defining Models and Computing Responses

November 14, 2021

Arduino-ROS Interface with Python – Send Messages From Python/ROS to Arduino and Receive Messages Back – Tutorial

August 26, 2023

This website contains more than *250* free tutorials! Every tutorial is accompanied by a YouTube video. All the tutorials are completely free. **The website has more than *50,000* visitors per month from all over the world!**

Search …

SEARCH

RECENT POSTS

How to See if ROS2 Package is Installed and How to Search for ROS2 Package

How to Display a List of All Packages Installed in ROS2

Interface Encoder with STM32 Microcontroller and Read Encoder Angle Measurements

STM32 Microcontroller Tutorial 7: External Interrupts

Control of Drone and Quadrotor DC Motors by Using Teensy/Arduino and PWM Signals

RECENT COMMENTS

admin on Deep Q Networks (DQN) in Python From Scratch by Using OpenAI Gym and TensorFlow- Reinforcement Learning Tutorial

Mike on Deep Q Networks (DQN) in Python From Scratch by Using OpenAI Gym and TensorFlow- Reinforcement Learning Tutorial

admin on Easy Introduction to Observability and Open-loop Observers with MATLAB Implementation

Cakan on Easy Introduction to Observability and Open-loop Observers with MATLAB Implementation

admin on Simple and Easy-to-Understand Introduction to Recurrent Neural Networks for Time-Series Prediction in Keras and TensorFlow

ARCHIVES

June 2021

April 2021

March 2021

February 2021

January 2021

December 2020

November 2020

October 2020

September 2020

July 2020

June 2020

May 2020

March 2020

January 2020

December 2019

November 2019

September 2019

August 2019

July 2019

June 2019

May 2019

CATEGORIES

AP Physics

Arduino

C/C++

Calculus

Computer Science

Computer Vision

Control Systems Lectures

Control Theory

Control/Estimation

Digital Signal Processing

drones

Dynamics

Graph Theory

Kalman Filter

Kinematics

Latex

lidar

Linux

localization

Machine Learning

Machine Learning/Data Science

Math/Optimization

Mathematics

MATLAB

matplotlib

Mechatronics/Robotics

Mobile Robots

Model Predictive Control

Nonlinear Systems

OpenCV

Optimal Control

Optimization

Particle Filters

Physics

Pygame

Python

quadcopter

Raspberry Pi

ROS

ROS/ROS2

ROS2

Scientific Computing

Signal Processing

SLAM

Statistics

STM32 microcontroller

SymPy

System Identification

Teensy

Ubuntu

Uncategorized

META

Log in

Entries feed

Comments feed

WordPress.org