

# *Kalman-and-Bayesian-Filters-in-Python*



## Preface



群名称: AibotBeginer  
群 号: 710288823



Mar 12th, 2022 <https://github.com/duyongquan/LTSLAM>



AibotBeginer 视觉SLAM  
quandy2020@126.com

# 2

## Outline

---

- Kalman and Bayesian Filters
- Motivation for this Book
- Reading Online
- PDF Version
- Downloading and Running the Book
- Jupyter
- SciPy, NumPy, and Matplotlib

3

# Kalman and Bayesian Filters

## 卡尔曼和贝叶斯滤波

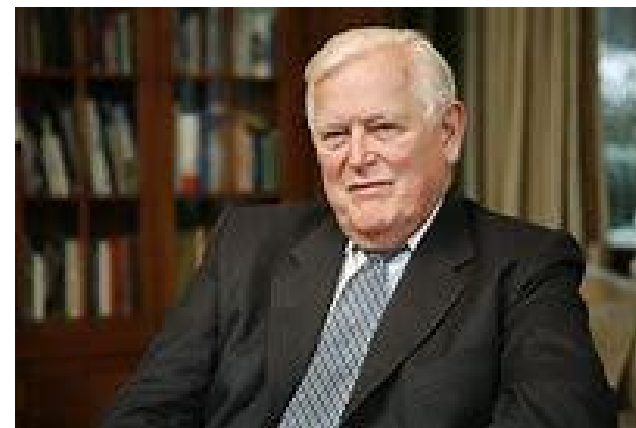
# 4

## Kalman and Bayesian Filters



### Sensors are noisy

- GPS in my car reports altitude, Each time pass the same point, it reports a slightly different altitude
- Kitchen scale gives different readings weigh the same object twice.
- Sensor is very noisy, environment makes data collection difficult. How can track?



Rudolf Emil Kálmán  
鲁道夫·卡尔曼 匈牙利裔美国数学家



### Introduce

- Apollo missions to the moon
- Aircraft, submarines, cruise missiles
- Medical imaging
- Robots, IoT (Internet of Things) , laboratory instruments

5

# Motivation for this Book

## 书的目标

# 6

## Motivation for this Book

◎ The theory is beautiful, but quite difficult to learn

- signal processing
- control theory
- probability and statistics

◎ Other books quite difficult to learn

- notation is introduced without explanation
- books are almost devoid of examples or worked problems
- no idea as to what real world phenomena these words and math were attempting to describe

7

# Reading Online

## 在线阅读

# 8

## Reading Online



GitHub



<https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python>



ninder



binder serves interactive notebooks online



<http://mybinder.org/repo/rlabbe/Kalman-and-Bayesian-Filters-in-Python>



nbviewe



<https://nbviewer.org/github/rlabbe/Kalman-and-Bayesian-Filters-in-Python/tree/master/>



9

# PDF Version

## PDF 版本



PDF book

[https://drive.google.com/file/d/0By\\_SW19c1BfhSVFzNHc0SjduNzg/view?resourcekey=0-41oIC9ht9xE3wQe2zHZ45A](https://drive.google.com/file/d/0By_SW19c1BfhSVFzNHc0SjduNzg/view?resourcekey=0-41oIC9ht9xE3wQe2zHZ45A)

Preface	5
0.1 Kalman and Bayesian Filters	5
0.2 Motivation for this Book	6
0.3 Reading Online	7
0.4 PDF Version	7
0.5 Downloading and Running the Book	7
0.6 Jupyter	8
0.7 SciPy, NumPy, and Matplotlib	9
0.7.1 Exercise - Create arrays	13
0.7.2 Solution	13
0.8 Companion Software	14
0.9 Thoughts on Python and Coding Math	14
0.10 License	15
0.11 Contact	15
0.12 Resources	15
<b>1 The g-h Filter</b>	<b>17</b>
1.1 Building Intuition via Thought Experiments	17
1.2 The g-h Filter	29
1.3 Notation	32
1.4 Exercise: Write Generic Algorithm	32
1.4.1 Solution and Discussion	33
1.5 Choice of $g$ and $h$	34
1.6 Exercise: create measurement function	34
1.6.1 Solution	34
1.6.2 Discussion	35
1.7 Exercise: Bad Initial Conditions	35
1.7.1 Solution and Discussion	36
1.8 Exercise: Extreme Noise	36
1.8.1 Solution and Discussion	36
1.9 Exercise: The Effect of Acceleration	37
1.9.1 Solution and Discussion	37
1.10 Exercise: Varying $g$	38
1.10.1 Solution and Discussion	38
1.11 Varying $h$	40
1.12 Interactive Example	41
1.13 Don't Lie to the Filter	42
1.14 Tracking a Train	43
1.15 g-h Filters with FilterPy	47
1.16 Summary	49

<b>2 Discrete Bayes Filter</b>	<b>51</b>
2.1 Tracking a Dog	51
2.2 Extracting Information from Sensor Readings	53
2.3 Noisy Sensors	54
2.4 Incorporating Movement	57
2.5 Terminology	59
2.6 Adding Uncertainty to the Prediction	60
2.7 Generalizing with Convolution	63
2.8 Integrating Measurements and Movement Updates	65
2.9 The Discrete Bayes Algorithm	68
2.10 The Effect of Bad Sensor Data	71
2.11 Drawbacks and Limitations	73
2.12 Tracking and Control	74
2.12.1 Simulating the Train Behavior	74
2.13 Bayes Theorem and the Total Probability Theorem	78
2.14 Summary	79
2.15 References	79
<b>3 Probabilities, Gaussians, and Bayes' Theorem</b>	<b>81</b>
3.1 Introduction	81
3.2 Mean, Variance, and Standard Deviations	81
3.2.1 Random Variables	81
3.3 Probability Distribution	82
3.3.1 The Mean, Median, and Mode of a Random Variable	83
3.4 Expected Value of a Random Variable	84
3.4.1 Exercise	85
3.4.2 Solution	85
3.4.3 Exercise	86
3.4.4 Solution	86
3.4.5 Variance of a Random Variable	86
3.4.6 Why the Square of the Differences	90
3.5 Gaussians	91
3.6 Nomenclature	92
3.7 Gaussian Distributions	93
3.8 The Variance and Belief	96
3.9 The 68-95-99.7 Rule	97
3.10 Interactive Gaussians	98
3.11 Computational Properties of Gaussians	99
3.12 Putting it all Together	101
3.12.1 Bayes Theorem	102
3.12.2 Total Probability Theorem	104
3.13 Computing Probabilities with scipy.stats	105
3.14 Limitations of Using Gaussians to Model the World	106
3.15 Product of Gaussians (Optional)	109
3.16 Sum of Gaussians (Optional)	110
3.17 Summary and Key Points	111
3.18 References	111
3.19 Useful Wikipedia Links	112

# Downloading and Running the Book

下载和运行

# Downloading and Running the Book



## Installation



<http://nbviewer.ipython.org/github/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/Appendix-A-Installation.ipynb>



## Introduce

This book is intended to be interactive and I recommend using it.

Its a little more effort to set up.

You can perform experiments, see how filters react to different data, see how different filters react to the same data, and so on

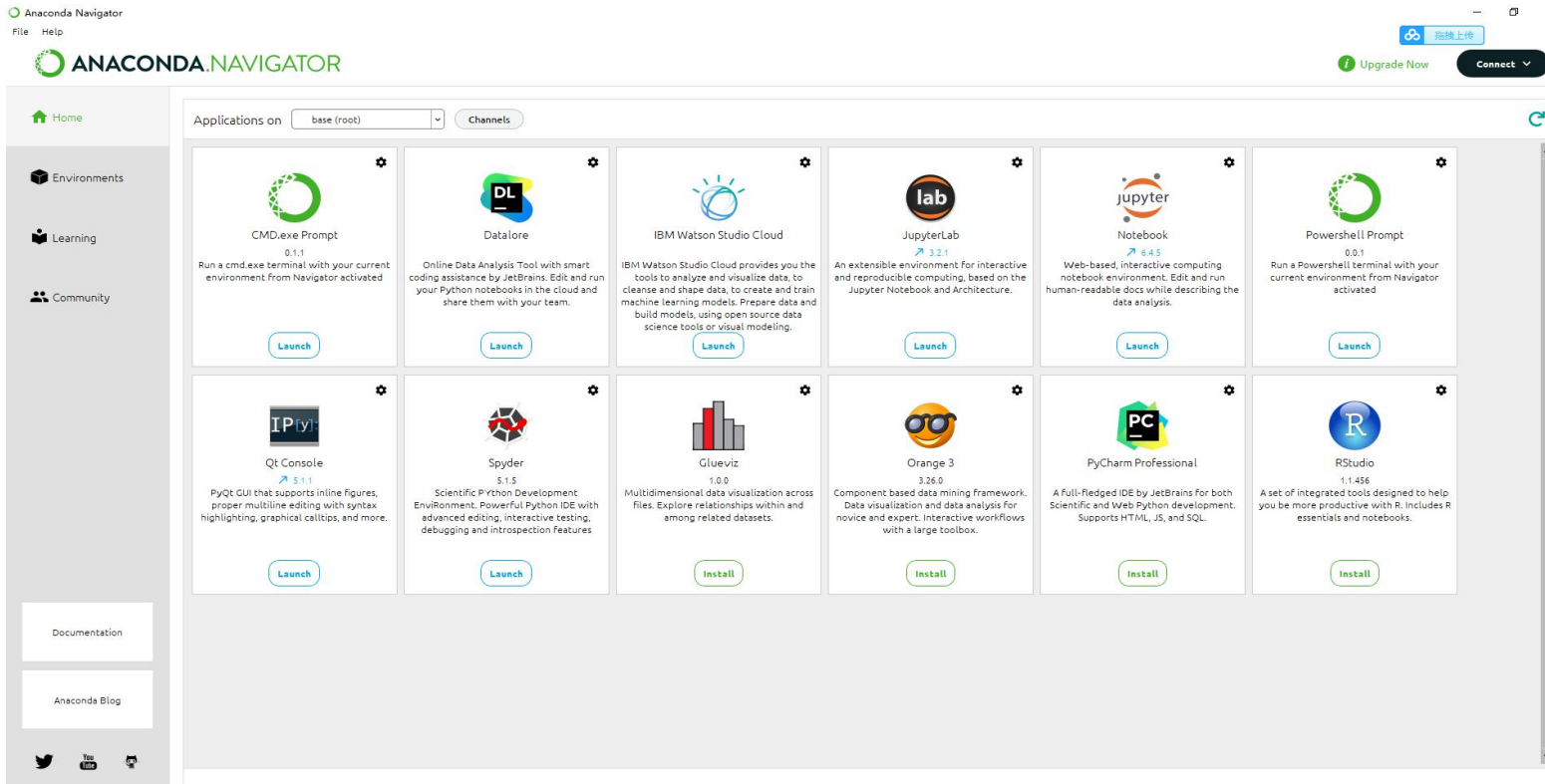
13

# Jupyter 开发环境



## jupyter notebook

安装Anaconda



15

# SciPy, NumPy, and Matplotlib

## Python 库

# SciPy, NumPy, and Matplotlib

## SciPy is a mathematic

array objects

linear algebra

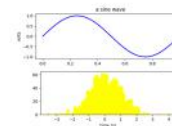
random numbers

<https://scipy.org/>

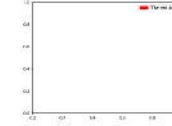
## NumPy

## Matplotlib

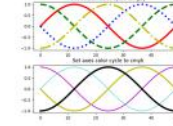
<https://matplotlib.org/stable/tutorials/index>



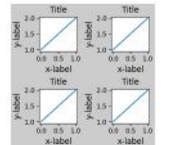
Artist tutorial



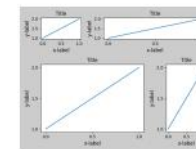
Legend guide



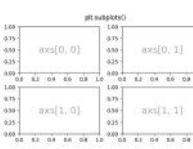
Styling withycler



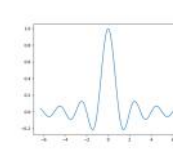
Constrained Layout Guide



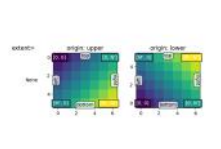
Tight Layout guide



Arranging multiple Axes in a Figure



Autoscaling



origin and extent in imshow



# SciPy, NumPy, and Matplotlib

- numpy.array implements a one or more dimensional array

```
In [3]: import numpy as np
        x = np.array([1, 2, 3])
        print(type(x))
        x
```

```
<class 'numpy.ndarray'>
```

```
Out[3]: array([1, 2, 3])
```

# SciPy, NumPy, and Matplotlib

- numpy.array implements a one or more dimensional array

```
In [4]: x = np.array((4,5,6))  
x
```


```
Out[4]: array([4, 5, 6])
```

Create multidimensional arrays with nested brackets:

```
In [5]: x = np.array([[1, 2, 3],  
                      [4, 5, 6]])  
print(x)
```

```
[[1 2 3]  
 [4 5 6]]
```

# SciPy, NumPy, and Matplotlib

-  `numpy.array` implements a one or more dimensional array

```
In [6]: x = np.array([1, 2, 3], dtype=float)
        print(x)
```

[1. 2. 3.]

You can access the array elements using subscript location:

```
In [7]: x = np.array([[1, 2, 3],
                        [4, 5, 6]])

print(x[1,2])
```

```
print(x[1,2])
```

# SciPy, NumPy, and Matplotlib

- numpy.array implements a one or more dimensional array

You can access a column or row by using slices. A colon (:) used as a subscript is shorthand for all data in that row or column. So `x[:,0]` returns an array of all data in the first column (the 0 specifies the first column):

```
In [8]: x[:, 0]
```

```
Out[8]: array([1, 4])
```

We can get the second row with:

We can get the second row with:

```
In [9]: x[1, :]
```

```
Out[9]: array([4, 5, 6])
```

■ Get the last two elements of the second row with:

```
In [10]: x[1, 1:]
```

```
Out[10]: array([5, 6])
```

■ As with Python `lists`, you can use negative indexes to refer to the end of the array. `-1` refers to the last index. So another way to get the last two elements of the second (last) row would be:

```
In [11]: x[-1, -2:]
```

```
Out[11]: array([5, 6])
```

# SciPy, NumPy, and Matplotlib

You can perform matrix addition with the `+` operator, but matrix multiplication requires the `dot` method or function. The `*` operator performs element-wise multiplication, which is **not** what you want for linear algebra.

```
In [12]: x = np.array([[1., 2.],  
                       [3., 4.]])  
         print('addition:\n', x + x)  
         print('\nelement-wise multiplication\n', x * x)  
         print('\nmultiplication\n', np.dot(x, x))  
         print('\ndot is also a member of np.array\n', x.dot(x))
```

addition:

```
[[2. 4.]  
 [6. 8.]]
```

multiplication

```
[[ 7. 10.]  
 [15. 22.]]
```

element-wise multiplication

```
[[ 1.  4.]  
 [ 9. 16.]]
```

dot is also a member of np.array

```
[[ 7. 10.]  
 [15. 22.]]
```

# SciPy, NumPy, and Matplotlib

Python 3.5 introduced the @ operator for matrix multiplication.

```
In [13]: x @ x
```

```
Out[13]: array([[ 7., 10.],  
               [15., 22.]])
```

transpose and the inverse

```
In [14]: import scipy.linalg as linalg  
         print('transpose\n', x.T)  
         print('\nNumPy ninverse\n', np.linalg.inv(x))  
         print('\nSciPy inverse\n', linalg.inv(x))
```

transpose	NumPy ninverse	SciPy inverse
[[1. 3.]	[[ -2.  1. ]	[[ -2.  1. ]
[2. 4.]]	[ 1.5 -0.5]]	[ 1.5 -0.5]]



# SciPy, NumPy, and Matplotlib

zeros matrix , ones matrix, eye identity matrix

```
In [15]: print('zeros\n', np.zeros(7))  
          print('\nzeros(3x2)\n', np.zeros((3, 2)))  
          print('\neye\n', np.eye(3))
```

zeros

```
[0. 0. 0. 0. 0. 0. 0.]
```

zeros(3x2)

```
[[0. 0.]
```

```
[0. 0.]
```

```
[0. 0.]]
```

eye

```
[[1. 0. 0.]
```

```
[0. 1. 0.]
```

```
[0. 0. 1.]]
```



# SciPy, NumPy, and Matplotlib

```
In [16]: np.arange(0, 2, 0.1)
```

```
Out[16]: array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1,  
               1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9])
```

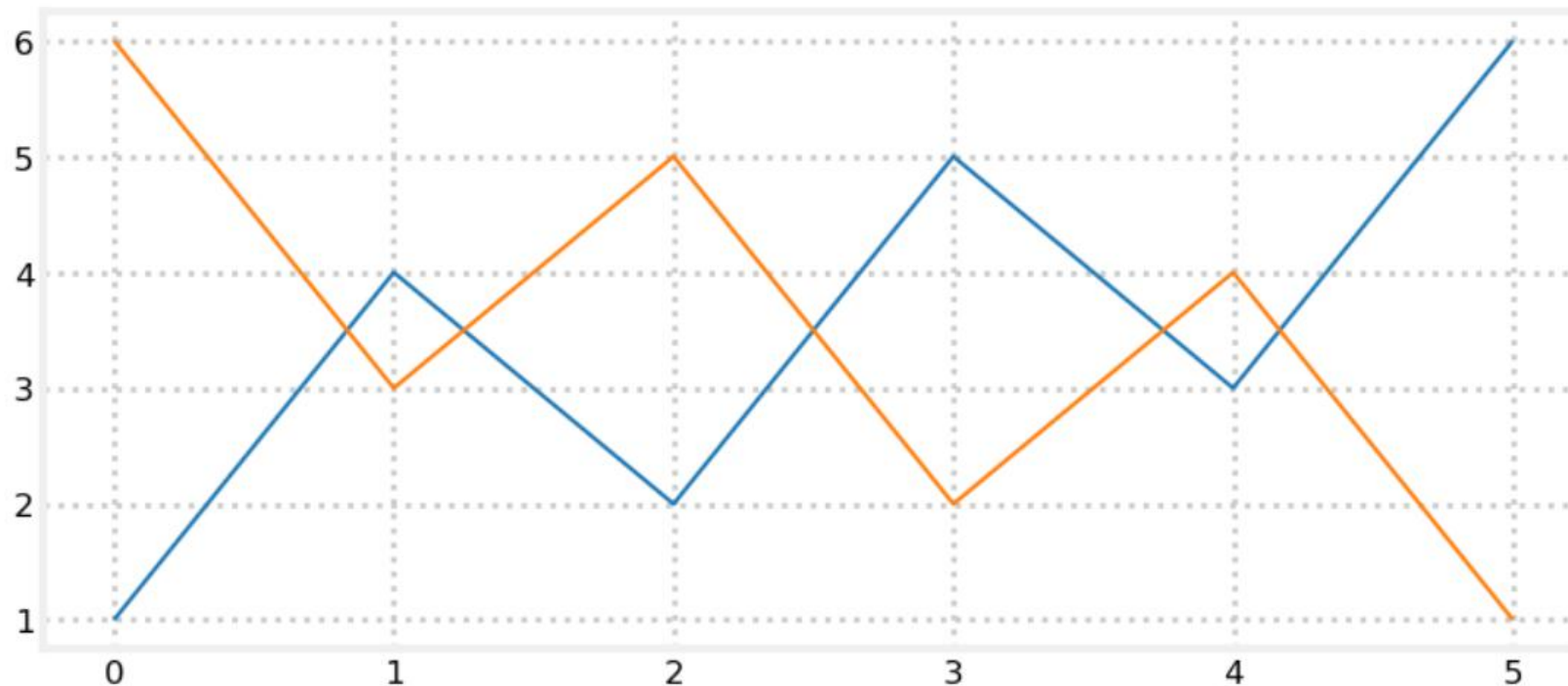
```
In [17]: np.linspace(0, 2, 20)
```

```
Out[17]: array([0.    , 0.105, 0.211, 0.316, 0.421, 0.526, 0.632, 0.737, 0.842,  
               0.947, 1.053, 1.158, 1.263, 1.368, 1.474, 1.579, 1.684, 1.789,  
               1.895, 2.    ])
```

# SciPy, NumPy, and Matplotlib

```
In [18]: import matplotlib.pyplot as plt  
a = np.array([6, 3, 5, 2, 4, 1])  
plt.plot([1, 4, 2, 5, 3, 6])  
plt.plot(a)
```

```
Out[18]: [<matplotlib.lines.Line2D at 0x298bcc7b4c8>]
```



A close-up shot of a hand pulling a dark-colored drawer from a desk. The word 'SUNSPRING' is printed in large, white, bold, sans-serif capital letters on the front of the drawer. On the desk surface above the drawer, there is a small yellow box, a silver thermos, and a blue and white box. The background is a warm, out-of-focus indoor setting.

**SUNSPRING**