National Cheng Kung University

Mechanical Engineering

Stochastic Dynamic Data - Analysis and Processing

Fundamental Estimators

Author:
Zhao Kai-Wen

Supervisor:
Chang Ren-Jung

December 7, 2012

Contents

1 Data Preprocessing

The assignment discusses the bias error estimator caused, and investigate the algorithm of density estimator.

1.1 Random Process

As we applied before, random process could be treated as a linear system. The low-pass filter is a useful form in processing. We take

$$H(s) = \frac{0.25}{s^2 + 0.7071s + 0.25}$$

as our random procedure.

Order	Cutoff frequency	Settling Time
n = 2	0.4994 rad/sec	16 sec

Table 1: Specs of the process

1.2 Data information

Source	Statistic Property	Data Length	JB Test($\alpha = 0.05$)
Matlab func normrnd()	$\mu = 0, \sigma = 1$	10^4	Accept

Table 2: The Specs of Raw Data

Sample Rate	Reject Data Length	Stationarity Test($\alpha = 0.05$)
1 Hz (TimeStep = 1 sec)	first 40 data	Accept

Table 3: The Specs of Processed Data

2 Autocorrelation Function Estimator

We implement a function to find autocorrelation function and compare with Matlab function.

2.1 Self-Define algorithm

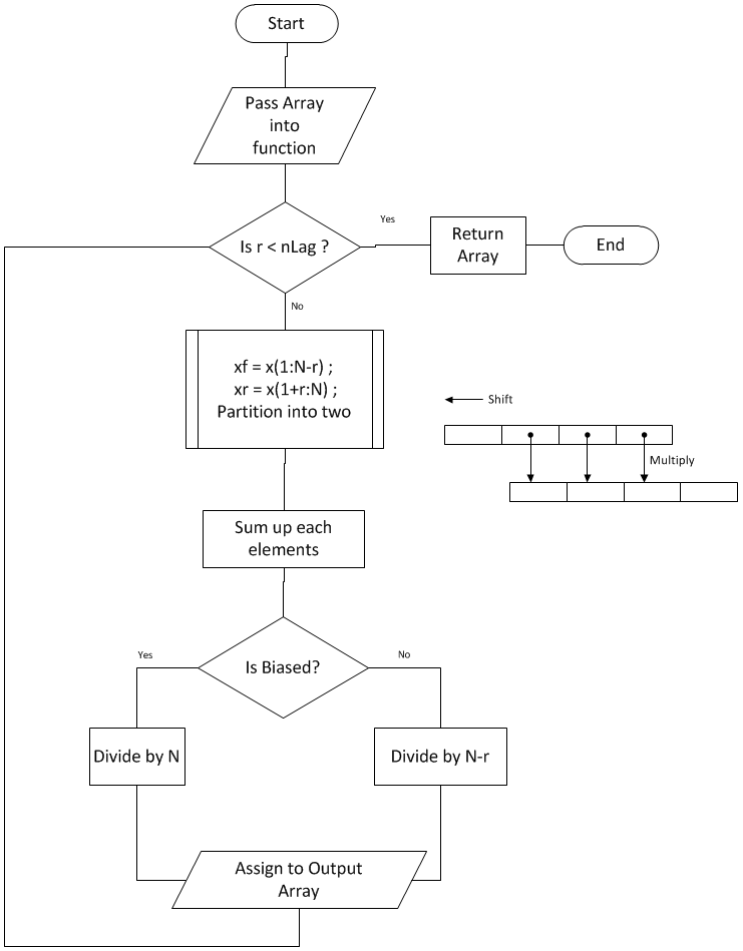


Figure 1: The flow chart of Autocorrelation function

2.2 Statistic Bias Error

We compare three kinds of function at the same plot, and they do not show significant difference because data length is much larger than time lag. So it is hard to find difference between biased form $\frac{1}{N}\sum^{N-r} x_i x_{i+r}$ and unbiased form $\frac{1}{N-r}\sum^{N-r} x_i x_{i+r}$.

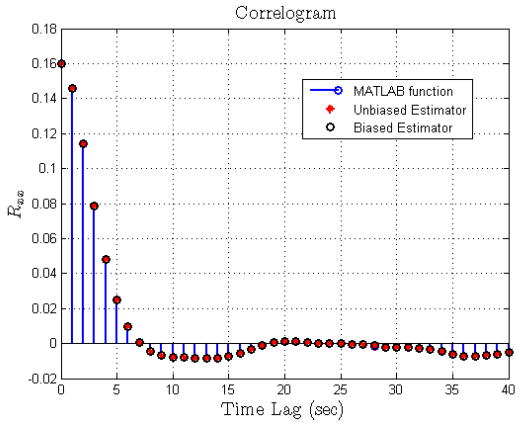


Figure 2: Comparision of Autocorrelation function

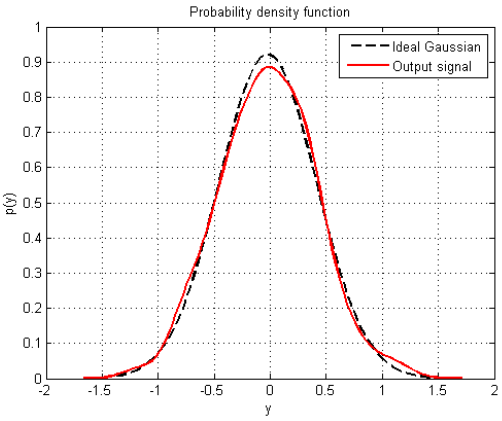


Figure 3: Invariant property of Gaussian Process

Observe the results, we could conclude the facts once again that Gaussian process is ergodic and invariant when it is linear process.
The graph on left-hand side indicates $2^n d$ moment envelop decay and right hand-side one tells us the output distribution is still be normal.

If we zoom in the graph, we not only could discovery the difference but could verify that MATLAB implement the function $xcorr()$ with unbiased formulation.

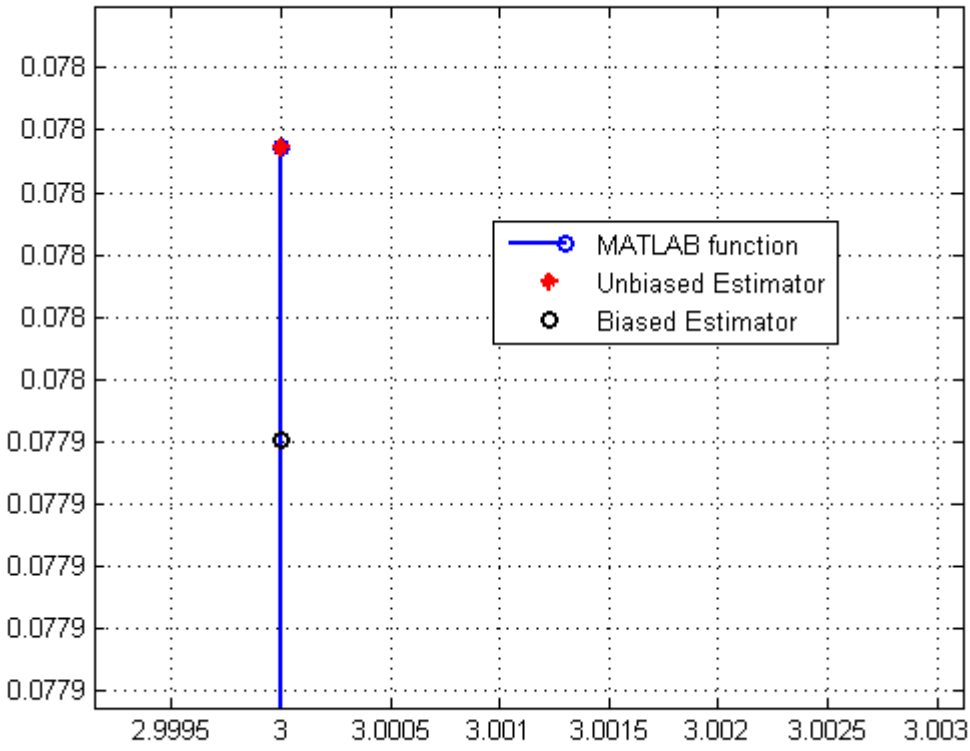


Figure 4: Zoom in the Figure 2

As we open the source code of matlab function $xcorr()$, I find that thay implemented the function in frequency domain. It transfers data by fft first, after processing it takes inverse fft back to time domain. My function is implemented by definition and all calculation handled in time domain.

3 Probability Density Estimator

We use probability density function often and its estimator worth deeply studying.

3.1 Algorithm of MATLAB function

We call the Matlab function named *ksdensity*(), witch means kernel smooth density estimator.

The idea of kernel smoother is that compute weighted average for X_0 by data within a close distance λ. And its weights are determined by kernel function, for example, kenerl of Gaussain distribution is $e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}}$.

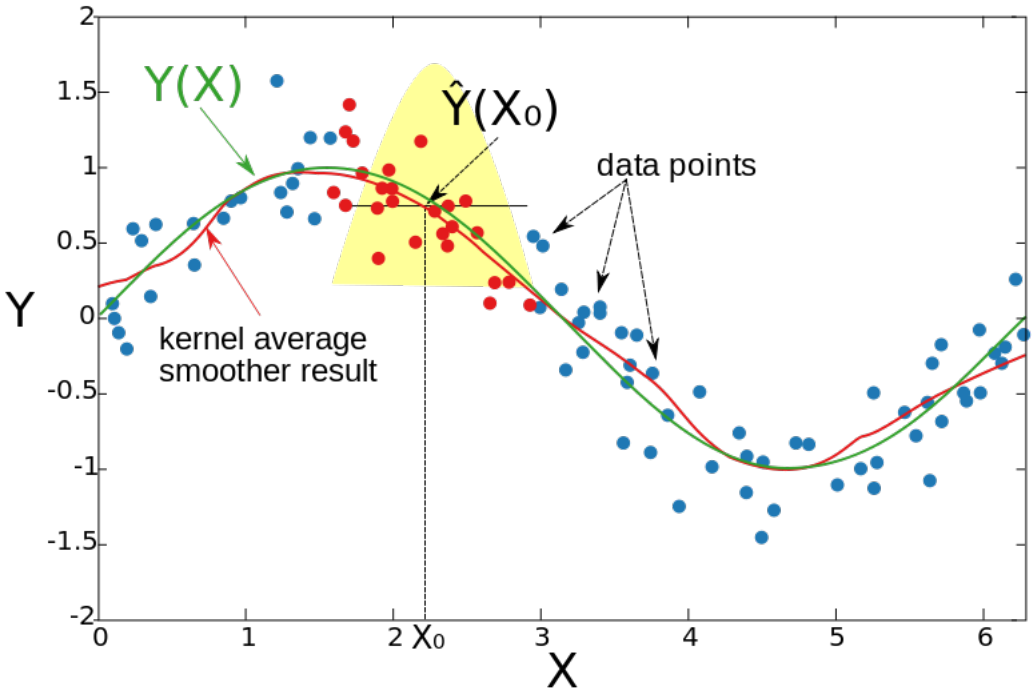


Figure 5: Schematic of Kenek Smoother algorithm

3.2 Self-Defined function

I define the density estimator by definition of probabilty density.

$$\hat{p}[x, W] = Prob[(x - W/2) \leq x(t) \leq (x + W/2)] = \frac{T_x}{T}$$

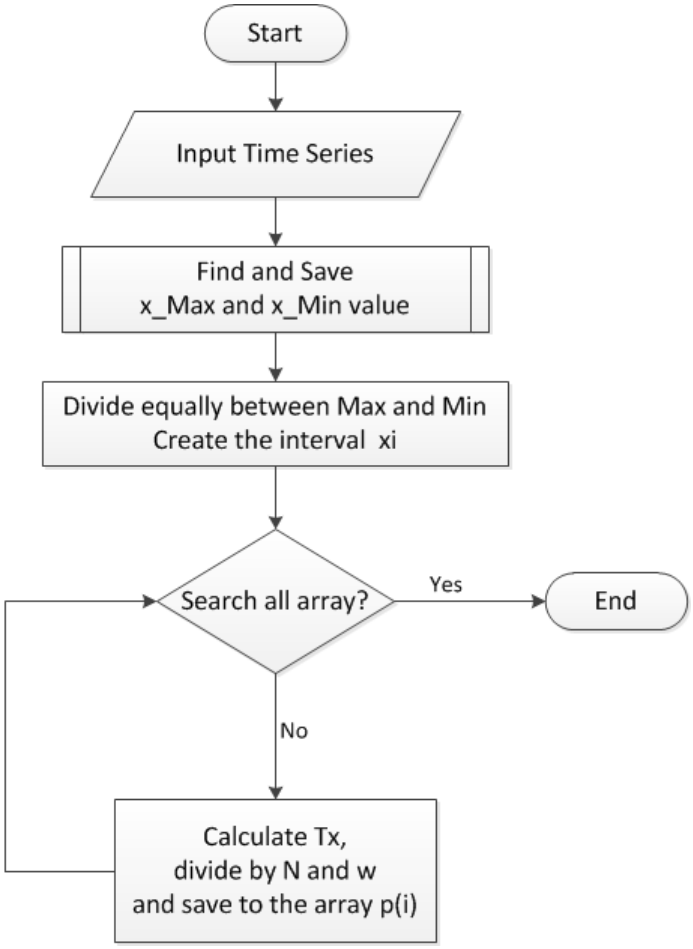


Figure 6: Flow chart of self-defined density estimator

3.3 Estimation

3.3.1 Single Variable Gaussian Distribution

We list the data and settings to different function.

Type	Methods	Data length	Width
Matlab	$ksdensity()$	10^4	0.5
Ideal normal	Analytical form	10^4	–
Self-Defined	$EstDensity()$	10^4	0.5

Table 4: Specs of the Data in different function

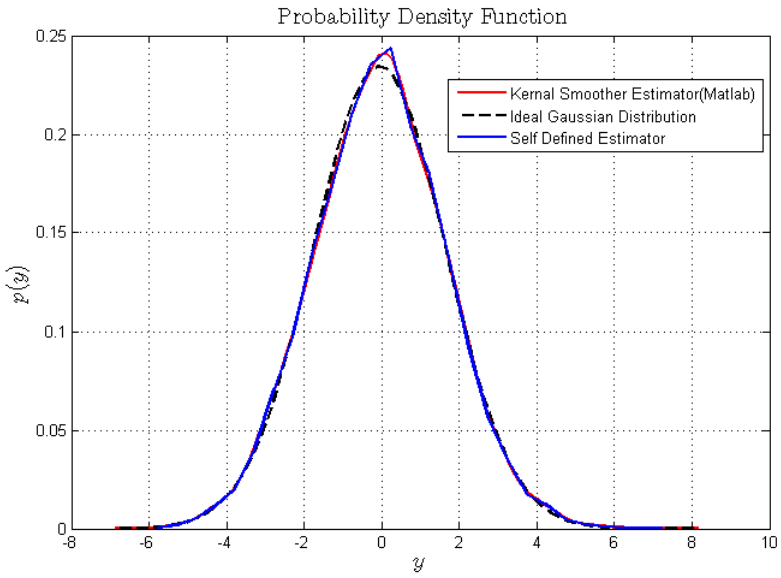


Figure 7: Estimation of Gaussian Distribution Density function

The result shows that three kinds of dist are almost indentical. But if we try different width (tougher condition), it gives significantly different.

Unsurprisingly, function defined by ourselves generate a worse result due to its crude and naive algorithm. Kernel smoother, on the othe hand, find a beautiful curve fitting the ideal distribution with just little bias.

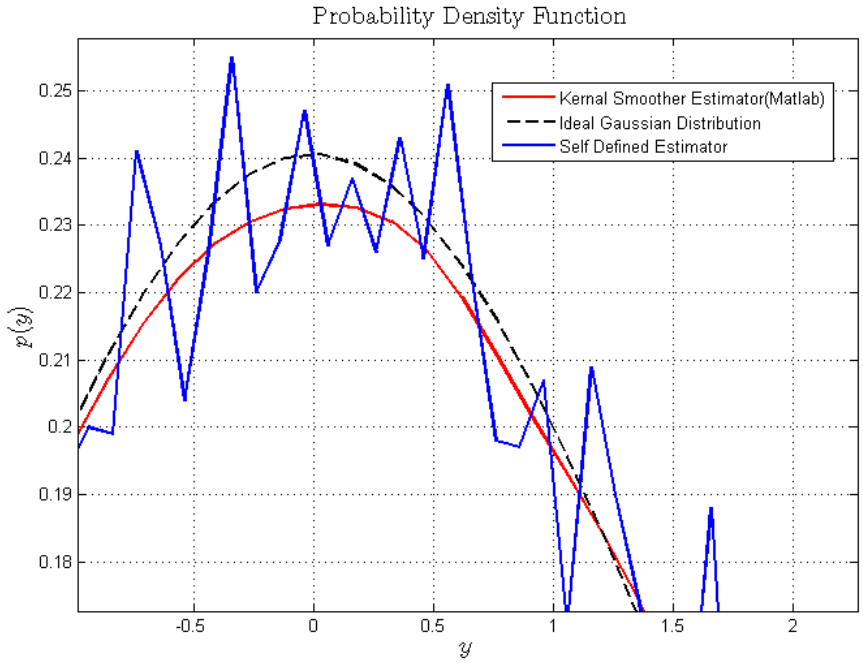
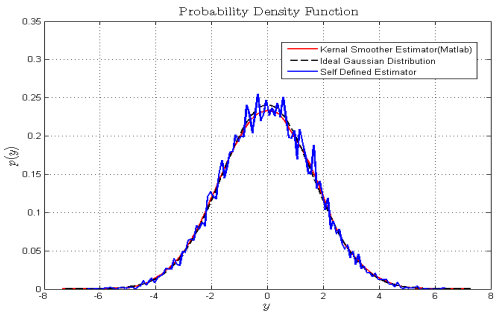
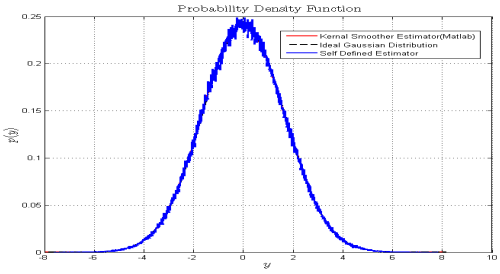


Figure 8: Zoom in Density function with width 0.01

If we want to eliminate these jitters, no doubt, increasing data length is always an option. The right-hand side one contains 100 times data of the left.





3.3.2 Joint Gaussian Distribution

In two dimensional distribution, it tends to generate fluctuation and jitters when using my function. It is more likely introduce error when cutting interval. We estimate the joint probability between input and output signal of the process.

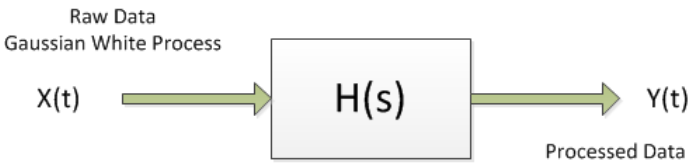


Figure 9: Block diagram of process

$$\hat{P}[x, W_x, y, W_y] = \frac{T_{xy}}{T}$$

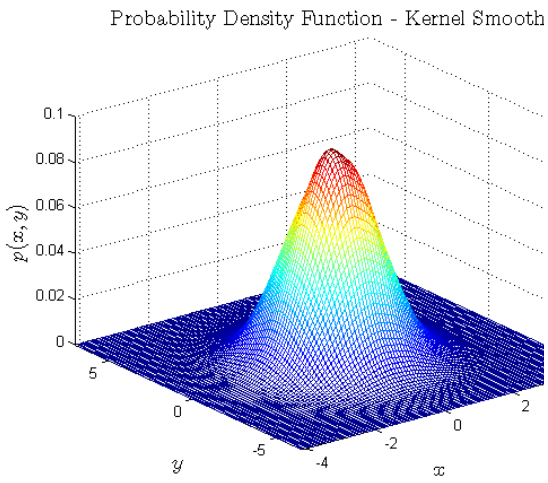


Figure 10: Kernel Smoother

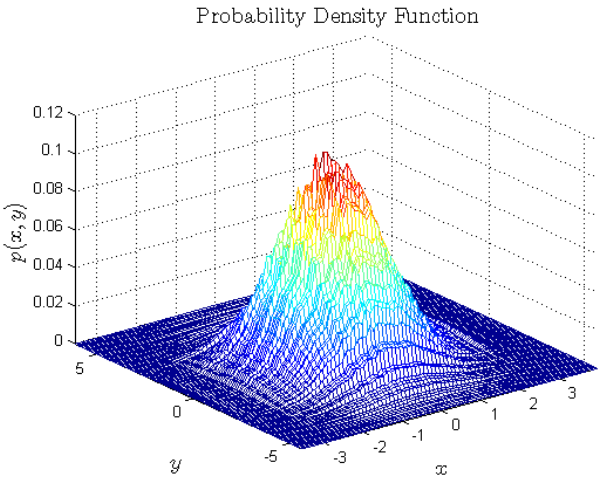


Figure 11: Mathematical Definition

3.3.3 Uniform Distribution

However, estimating uniform distribution is another story. Kernel smoother could generate a worse result than our function. We find that the red line has larger bias error. It seems come out from infeasible kernel function.

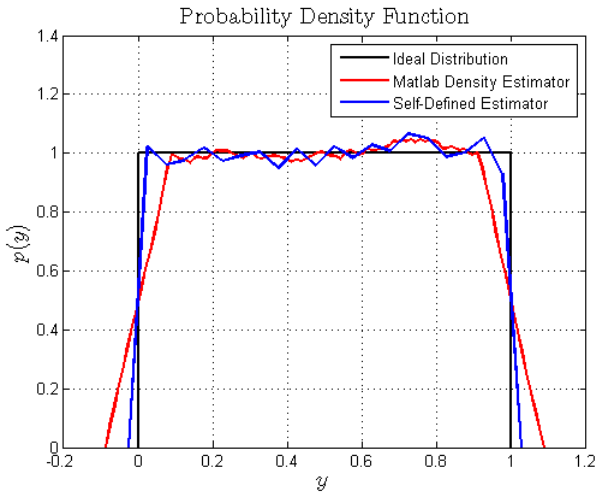


Figure 12: Estimation of Uniform distribution with width 0.05

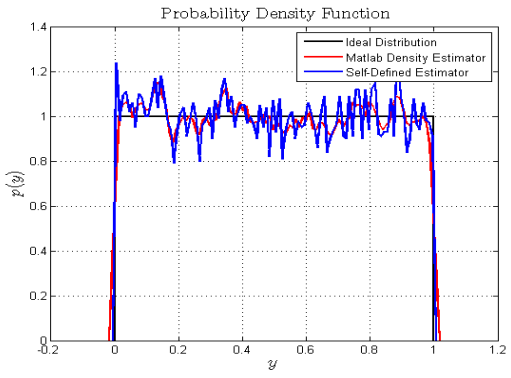


Figure 13: T = 10,000

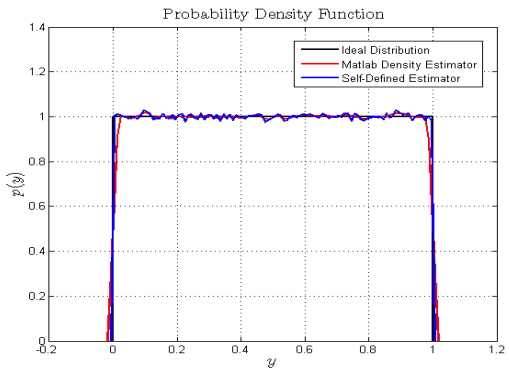


Figure 14: T = 1,000,000

As we tried before, more data could make fitting more accurate.

3.4 Optimal Window Width

Due to curiosity, I wonder how to select the best width when we only get finite (constrained) data length. The idea is getting the best estimation from the same data, in other word, is considering efficient quality.

3.4.1 Define the Optimal point

However, the tricky problem is what the best? Here, I think that we could tolerate small bias error and trade off with variance error. Both error exist but performance is not so bad.
By the way, there is another consideration. Variance error would interfere visualization and correctness significantly. We should confine it first then adjust bias error. The benefit of this way is when we acquire more information about data, we might fix bias error relatively easy.

3.4.2 Define Goal fucntion

All in all, I consider both bias and variance error together in the section. List the normalized mean square error first.

$$\epsilon^2[\hat{p}(x)] \sim \frac{c^2}{2BTWp(x)} + \frac{W^4}{576}[\frac{p(x)''}{p(x)}]^2$$

Window width is the variable and we treat other parameters are constant. So, the problem is greatly simplify to one varialbe optimization problem.

$$G(W) = c_1\frac{1}{W} + c_2W^4$$

where

$$c_1 = \frac{c^2}{2BTp(x)},\ c_2 = \frac{1}{576}[\frac{p(x)''}{p(x)}]^2$$

Then I try to find these coefficients.

We know the bandwidth B times length T is $N/2$ and $N = 2BT$. Then selecting $p(x)$ is an issue.
I determine probability function with standard normal distribution, which is $\mu = 0, \sigma = 1$ and the form could be written as

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{(-x^2/2)}$$

The analytical form of derivatives are

$$p'(x) = \frac{1}{\sqrt{2\pi}}[-x]e^{(-x^2/2)} = -xp(x)$$

$$p''(x) = \frac{1}{\sqrt{2\pi}}[x^2 - 1]e^{(-x^2/2)} = (x^2 - 1)p(x)$$

First order derivative means how steep and fluctuated it is.
Second order derivative shows how fast it fluctuates.
I expect the overall performance could be good, so hope error approaches zeio nearby the mean value.

$$c_1 = \frac{c^2}{Np(0)}$$

$$c_2 = \lim_{x \to 0} \frac{(x^2 - 1)^2}{576}$$

where $p(0) = 0.3989$, $N = 10^4$ and I take $c = 0.65$.
So, $c_1 = 1.06 \times 10^{-4}$ and c_2 is small but non-zero value, we pick $c_2 = 1 \times 10^{-4}$. In fact, I pick these numbers on purpose.

3.4.3 Optimization and Results Comparison

The specific, intentionally choosed coefficients could give us more clear picture but most cases are not so good as this one.

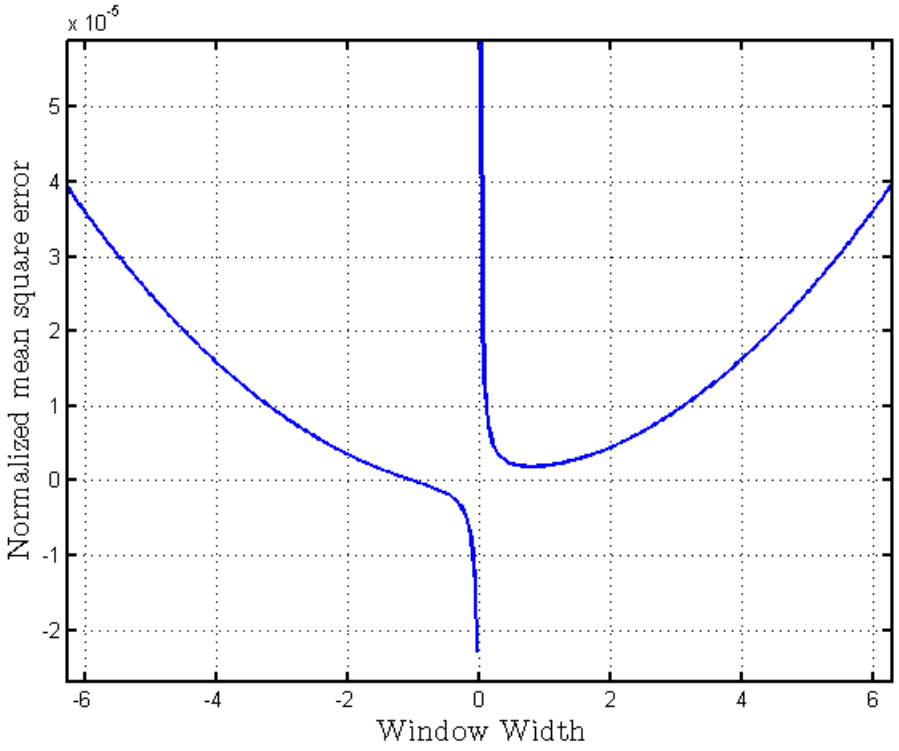


Figure 15: Graph of Polynomial

We find that too small or too large window width could generate great error. I take derivative of the goal function $G(x)$ and solve the equation can get

$$x = 0.3017$$

According to $RandomData - AnalysisandMeasurementProcedures$, we often take $W \leq 0.2\sigma_x$. In this case we have

$$0.2\sigma_x = 0.3364$$

They are similar and check the results.

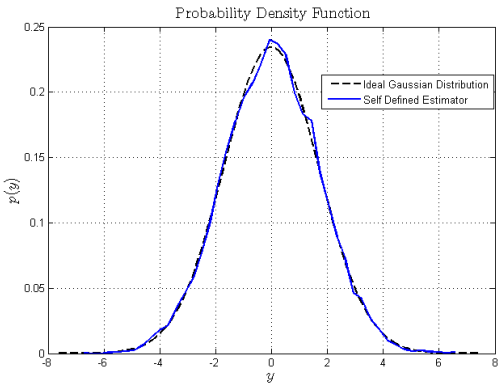


Figure 16: Optimization Window Width

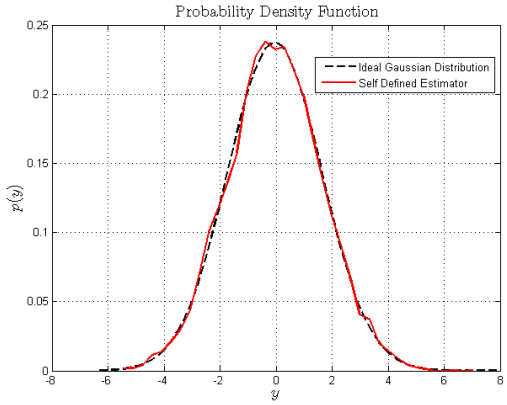


Figure 17: Window Width as $0.2\sigma_x$

They look okay but not really well as kernel smoother does. Reasons abund. Optimization I used, actually, is naive and too simple to find a good solution. I spent lots of time to try different coefficients and merely find a proper one. And pick width as $0.2\sigma_x$ is rough and not rigorous. However, consider the time consumption and efficiency, taking $0.2\sigma_x$ is incredible well. I think the latter is much better than my optimization.

References

[1] Wikipedia : Kernel density estimation

[2] Random Data, Allan Piersol and Julius Bendat