Gaussian smoothing using sliding discrete Fourier transform

Yukihiko Yamashita
School of Environment and Society Science and Engineering , Tokyo Institute of Technology
yamasita@ide.titech.ac.jp

July 24, 2018

1 Introduction

This document is to explain the program for Gaussian smoothing, differentials with Gaussian smoothing, and Laplacian of Gaussian of an image using sliding discrete Fourier transform.

First, we explain its mathematical bases. And we explain how to use the mex program for Matlab.

2 Gaussian smoothing using window discrete Fourier transform

Although an image is a 2D signal, it is enough to explain the algorithm for a 1D signal. Let $f[n]$ be a real-valued input signal. We define a discrete Gaussian function by

$$G[n] = e^{-\alpha n^2}. \tag{1}$$

Its standard deviation is given by $\frac{1}{\sqrt{2\alpha}}$. Its derivatives $G_{\mathrm{D}}(n)$ and the second order derivatives $G_{\mathrm{DD}}(n)$are given by

$$\begin{aligned} G_{\mathrm{D}}[n] &= -2\alpha n e^{-\alpha n^2}, \\ G_{\mathrm{DD}}[n] &= 4\alpha^2 n^2 e^{-\alpha n^2} - 2\alpha e^{-\alpha n^2}. \end{aligned}$$

Then, what we want to obtain is given by

$$g[n] = \sum_{k=-\infty}^{\infty} G[k] f[n-k], \tag{2}$$

$$g_{\mathrm{D}}[n] = \sum_{k=-\infty}^{\infty} G_{\mathrm{D}}[k] f[n-k], \tag{3}$$

$$g_{\mathrm{DD}}[n] = \sum_{k=-\infty}^{\infty} G_{\mathrm{DD}}[k] f[n-k]. \tag{4}$$

Since Gaussian function converges to zero rapidly, the sum can be truncated from $-K$ to K. Then, we have

$$g[n] \simeq \sum_{k=-K}^{K} G[k] f[n-k], \tag{5}$$

$$g_{\mathrm{D}}[n] \simeq \sum_{k=-K}^{K} G_{\mathrm{D}}[k] f[n-k], \tag{6}$$

$$g_{\mathrm{DD}}[n] \simeq \sum_{k=-K}^{K} G_{\mathrm{DD}}[k] f[n-k]. \tag{7}$$

From here, we consider to calculate the right hand side expressions using sliding discrete Fourier transform. Let $\beta = \pi/K$ and we consider the following approximations.

$$G[k] \simeq \sum_{p=0}^{P} a_p \cos(\beta p k), \tag{8}$$

$$G_{\mathrm{D}}[k] \simeq \sum_{p=1}^{P} b_p \sin(\beta p k), \tag{9}$$

$$G_{\mathrm{DD}}[k] \simeq \sum_{p=0}^{P} d_p \cos(\beta p k), \tag{10}$$

where P is the order to truncate the sliding discrete Fourier transform. We assume P is 2 or 4 or 6 for the calculation complexity.

The sliding discrete Fourier transform of input signal is defined by

$$c_p[n] = \sum_{k=-K}^{K} f[n-k]\cos(\beta p k), \tag{11}$$

$$s_p[n] = \sum_{k=-K}^{K} f[n-k]\sin(\beta p k). \tag{12}$$

Then, eqs.(5), (6), and (7) are approximated by

$$g[n] \simeq \sum_{p=0}^{P} a_p c_p[n]. \tag{13}$$

$$g_{\mathrm{D}}[n] \simeq \sum_{p=1}^{P} b_p s_p[n], \tag{14}$$

$$g_{\mathrm{DD}}[n] \simeq \sum_{p=0}^{P} d_p c_p[n]. \tag{15}$$

Here, we discuss the algorithm to calculate eqs.(11) and (12).

2.1 Evaluation of error by truncation

Kober [3] proposed to calculate them by using the kernel integral. We describe it by using complex numbers for brevity. We define $u[n]$ by

$$u[n] = \sum_{k}^{n} f[k] e^{i\beta p k}. \tag{16}$$

By using (16), the sliding discrete Fourier transform is given by

$$c_p[n] + i s_p[n] = e^{-i\beta p n}(u[n+K] - u[n-K-1]) \tag{17}$$

Furthermore, $u[n]$ can be easily calculated by the following recurrence relation

$$u[n] = u[n-1] + f[n] e^{i\beta p n} \tag{18}$$

We modify the method by using the following IIR filter.

$$v[n] = e^{-i\beta p} v[n-1] + f[n] \tag{19}$$

Then, the sliding discrete Fourier transform is given by

$$c_p[n] + i s_p[n] = (-1)^p (v[n+K] - v[n-K] + x[n-K]).$$

Sugimoto et al. [1, 2, 3] proposed to use the second order IIR filter. Let's substitute $n+1$ into n in $v[n] = e^{-i\beta p} v[n-1] + f[n]$, then we have

$$v[n+1] = e^{-i\beta p} v[n] + f[n+1]. \tag{20}$$

Multiply $e^{i\beta p}$ to eq.(19) and subtract it from eq.(20), and we have

$$v[n+1] = 2\cos(\beta p) v[n] + v[n-1] + f[n+1] - e^{i\beta p} f[n]. \tag{21}$$

By using eq.(21), we can calculate the terms for cos and sin separately. However, because it is something like the second order differential equation, the calculation accuracy will be degraded.

Not integrals from $-\infty$ but integrals in an interval can be calculated directly. With the recurrence relation

$$u'[n] = u'[n-1] + f[n]e^{i\beta pn} - f[k-2K-1]e^{i\beta p(n-2K-1)}, \tag{22}$$

we have

$$c_p[n] + is_p[n] = e^{-i\beta pn}u'[n+K]. \tag{23}$$

With the first order IIR filter

$$\tilde{v}'[n] = e^{-i\beta p}\tilde{v}'[n-1] + f[n] - f[n-2K]. \tag{24}$$

we have

$$c_p[n] + is_p[n] = (-1)^p(\tilde{v}'[n+K] + x[n-K]). \tag{25}$$

Similarly, from

$$v'[n+1] = 2\cos(\beta p)v'[n] - v'[n-1] + f[n+1] - e^{i\beta p}f[n] - e^{-i\beta p}f[n-2K] + f[n-2K-1], \tag{26}$$

we have

$$c_p[n] + is_p[n] = (-1)^p v'[n+K]. \tag{27}$$

The calculation errors of the above methods are different especially when a single precision floating point operations are used. We show it by experiments.

3 Calculation error

3.1 Truncation errors because of interval and order are limited

Let σ be a standard deviation of a Gaussian function when we assume the interval $[-K, K]$ as $[-\pi, \pi]$. The truncation errors because of interval and order are limited significantly depend on σ but not much on K. Fig. 1 shows Gauss function approximated by sliding discrete Fourier transform for different values of σ.

Next, we investigate truncation errors by changing σ.

Figs. 2, 3, and 4 show the truncation errors for $G[n]$, $G_{\mathrm{D}}[n]$, and $G_{\mathrm{DD}}[n]$, respectively. For each case, by setting σ to 1.1, 0.8, and 0.7 for $P = 2$, $P = 4$, and $P = 6$, respectively, the sum of truncation errors reach near to the minimum values.

3.2 Error by single precision floating point calculation

Because economical processors do not have double precision floating point unit, calculation by single precision floating point unit is very important. Fig. 5 shows the calculation error of $c_1[n]$ by using single precision floating point calculation. The error is measured by

$$\frac{\text{The maximum error in the interval of length 10,000}}{\text{The root mean square of the signal}} \times 100$$

The original signal is generated by concatenate the standard image data Lenna and Barbara by assuming them as a 1D data.

From Fig. 5, we can know the 2nd order IIR filter may not have enough precision for single precision floating point calculation.

4 Summery

In the program we use the following parameters. The standard deviation σ_{NP} in $[-\pi, \pi]$ that minimizes error for each P is given by

- $\sigma_{\mathrm{NP}} = 1.1$ when $P = 2$.
- $\sigma_{\mathrm{NP}} = 0.8$ when $P = 4$.
- $\sigma_{\mathrm{NP}} = 0.7$ when $P = 6$.

We prepare a_k, b_k, d_k for $P = 2, 4, 6$ and $0.9\sigma_{\mathrm{NP}}$, 0.95, σ_{NP}, $1.05\sigma_{\mathrm{NP}}$, $1.1\sigma_{\mathrm{NP}}$.

Let σ_{I} be the target standard deviation for smoothing in a image. Then, K is given by

$$K = \mathrm{round}\left(\frac{\pi\sigma_{\mathrm{I}}}{\sigma_{\mathrm{NP}}}\right)$$

Then, the target standard deviation σ_{N} in $[-\pi, \pi]$ is given by

$$\sigma_{\mathrm{N}} = \frac{\pi\sigma_{\mathrm{I}}}{K}.$$

Because σ_{N} is slightly different from σ_{NP} so that a_k, b_k, d_k are calculated by the liner interpolation.

For calculation of sliding discrete Fourier transform in the program, we use eqs.(24) and (25).

5 How to use

1. Download gaussSmooth.c
2. Compile it.
   ```
   mex gaussSmooth.c
   ```
3. You can use a function gaussSmooth() in matlab.
4. Input;
   - `inImg` : Input image (2D matrix).
   - `type` : 0: Gaussian smoothing, 1: x and y-directional differential with Gaussian smoothing, 3: LOG.
   - `sigma` : Standard deviation.
   - `P` : Order of Fourier transform (2 or 4 or 6).
   - `extType` : type of edge extension, 0: zero extension, 1: the value of edge is extended to its outside region.
5. Output: 1 or 2 images (2D matrix or 2 2D matrices)
6. Usage:
   - Gaussian smoothing (type = 1).
     ```
     smoothedImg = gaussSmooth(inImg, type, sigma, P, extType)
     ```
   - x and y-directional differential with Gaussian smoothing (type = 2).
     ```
     [diffX diffY] = gaussSmooth(inImg, type, sigma, P, extType)
     ```
   - LOG (type = 3).
     ```
     logImg = gaussSmooth(inImg, type, sigma, P, extType)
     ```

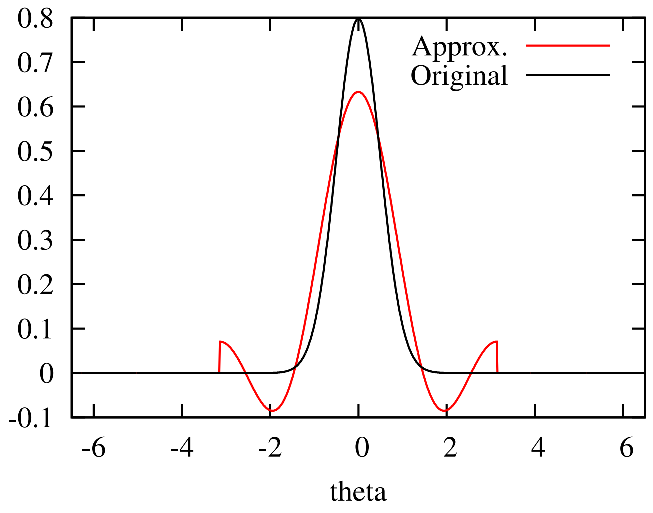
References

[1] K. Sugimoto, S. Kyochi, and S. Kamata, “Comprehensive Performance Analysis on Constant-time Gaussian Filter based on Discrete Cosine Transform, “ Technical Report of IEICE, vol.117, no.48,pp.19–24, May 2017. (in Japanese)

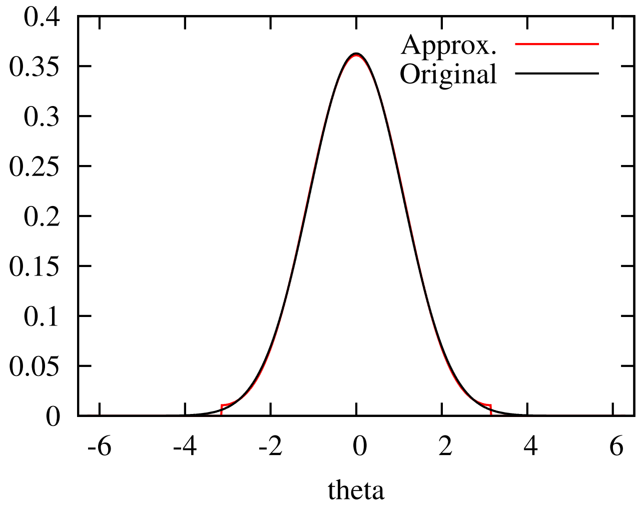
[2] K.=Sugimoto and S. Kamata, “Compressive bilateral filtering,” IEEE Transactions on Image Processing, vol.24, no.11, pp.3357–3369, Nov. 2015.

[3] E. Elboher and M. Werman, “Cosine integral images for fast spatial and range filtering,”2011 18th IEEE International Conference on Image Processing (ICIP), pp.89–92, Brussels, Sept. 2011.

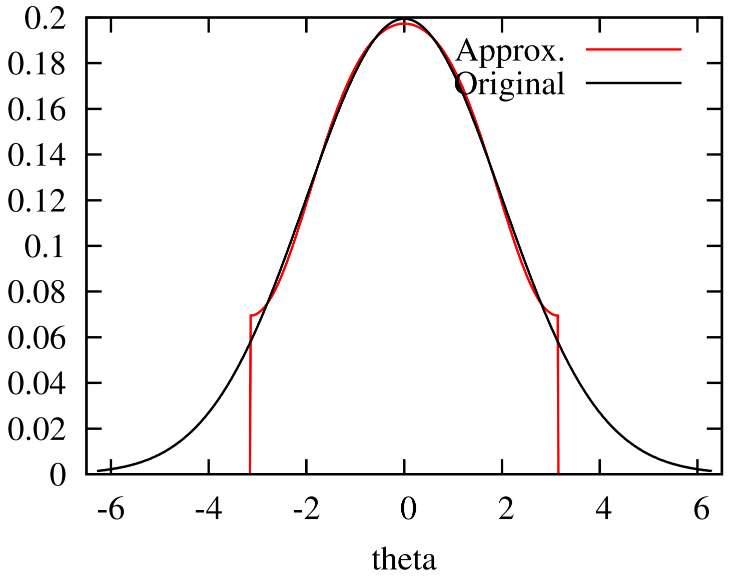
[4] V. Kober, “Fast algorithms for the computation of sliding discrete sinusoidal transforms,” IEEE Transactions on Signal Processing, vol.52, no.6, pp.1704–1710, June 2004.



$\sigma = 0.5$

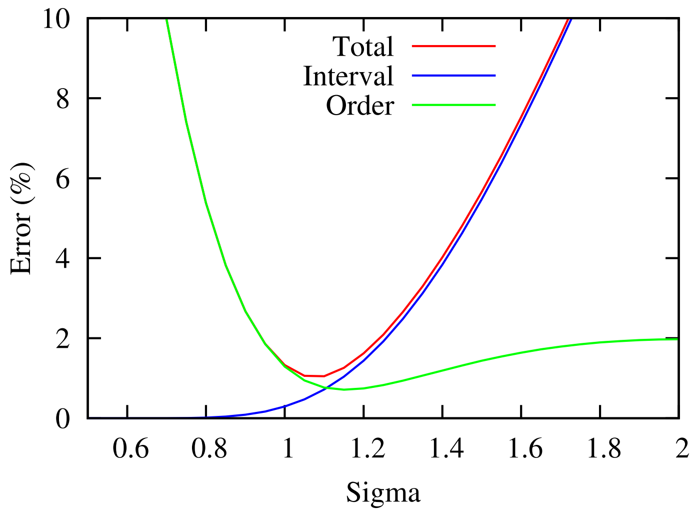


$\sigma = 1.1$

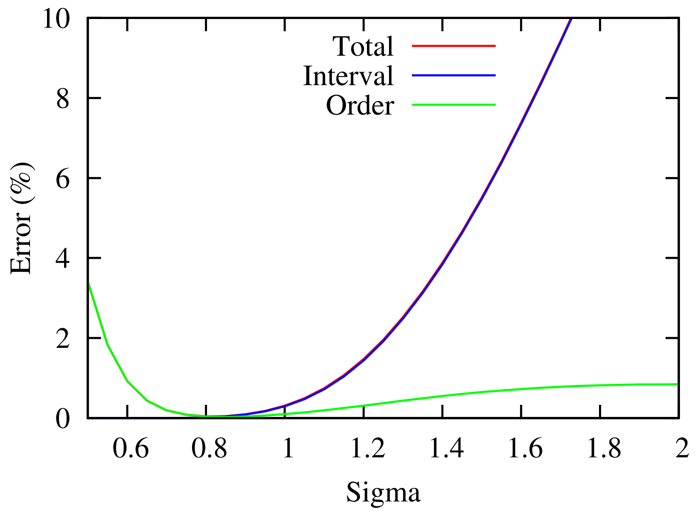


$\sigma = 2.0$

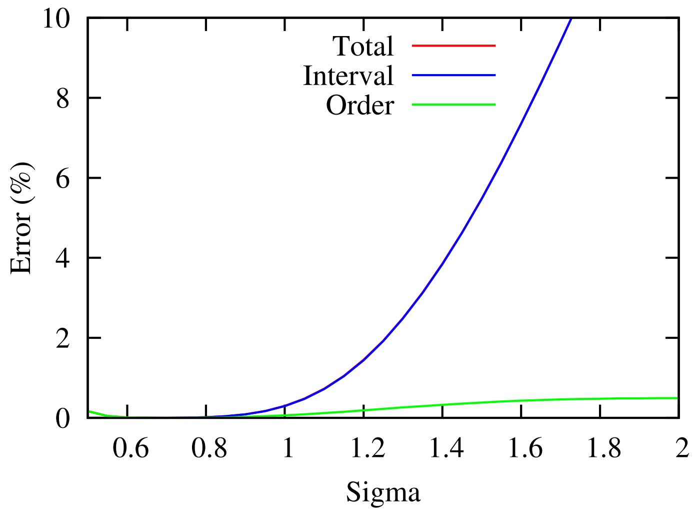
Figure 1: Gauss function approximated by sliding discrete Fourier transform



$G[n]$, $K = 256$, $P = 2$

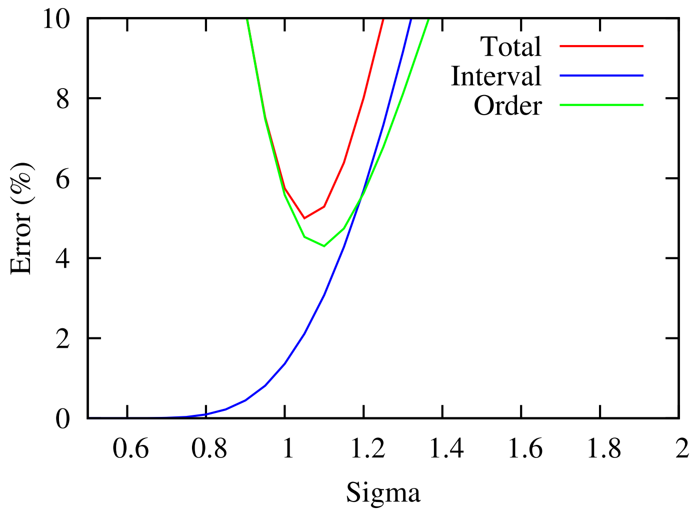


$G[n]$, $K = 256$, $P = 4$

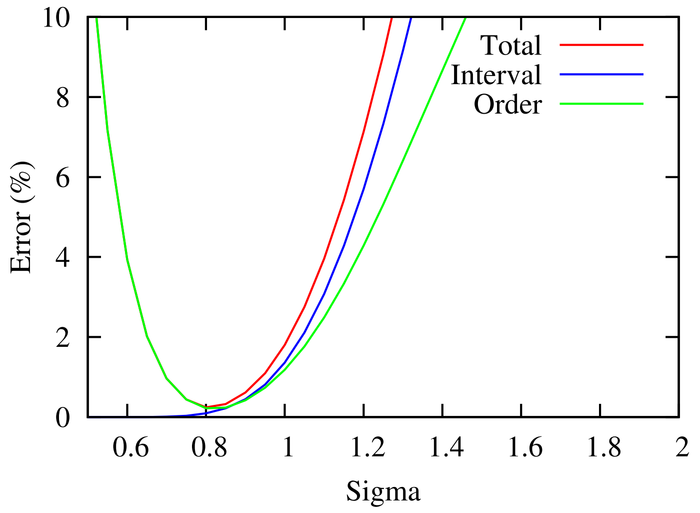


$G[n]$, $K = 256$, $P = 6$

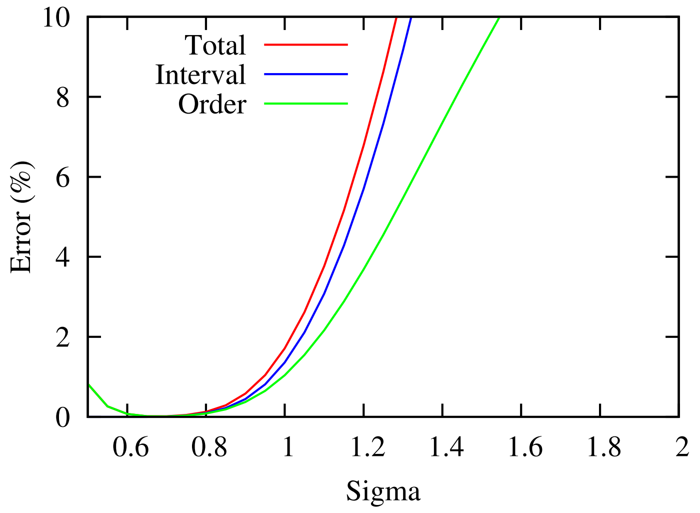
Figure 2: Truncation errors of Gaussian function with interval and order



$G_D[n]$, $K = 256$, $P = 2$

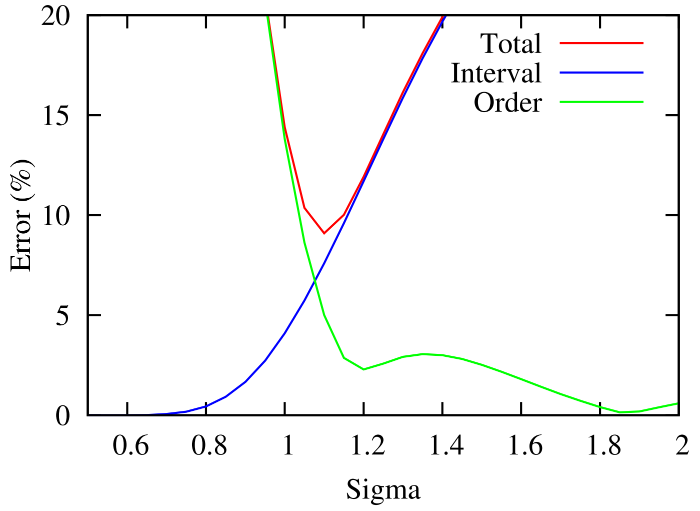


$G_D[n]$, $K = 256$, $P = 4$

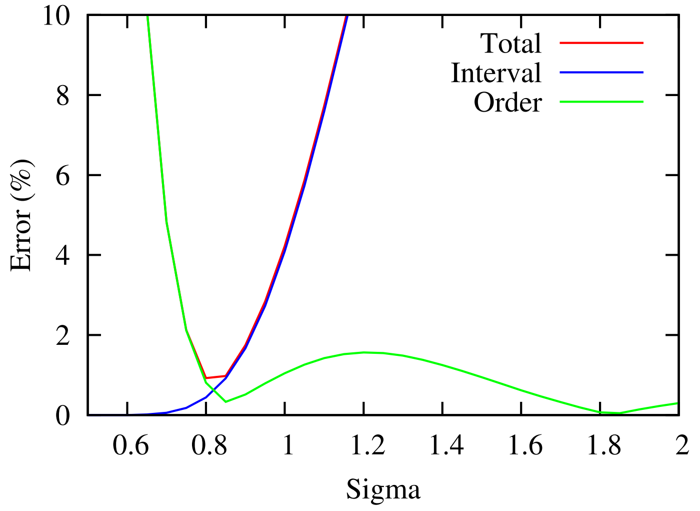


$G_D[n]$, $K = 256$, $P = 6$

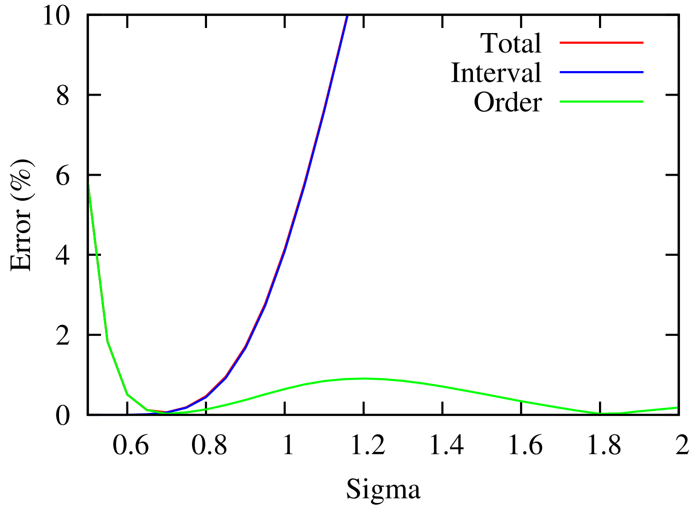
Figure 3: Truncation errors of derivatives of Gauss function with interval and order



$G_{\mathrm{DD}}[n]$, $K = 256$, $P = 2$

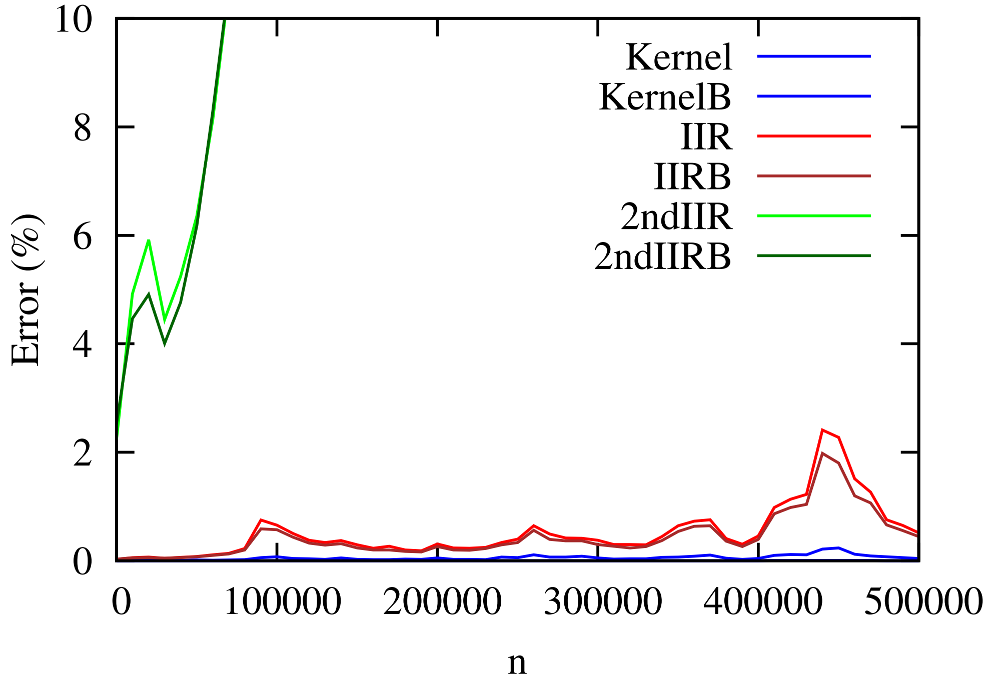


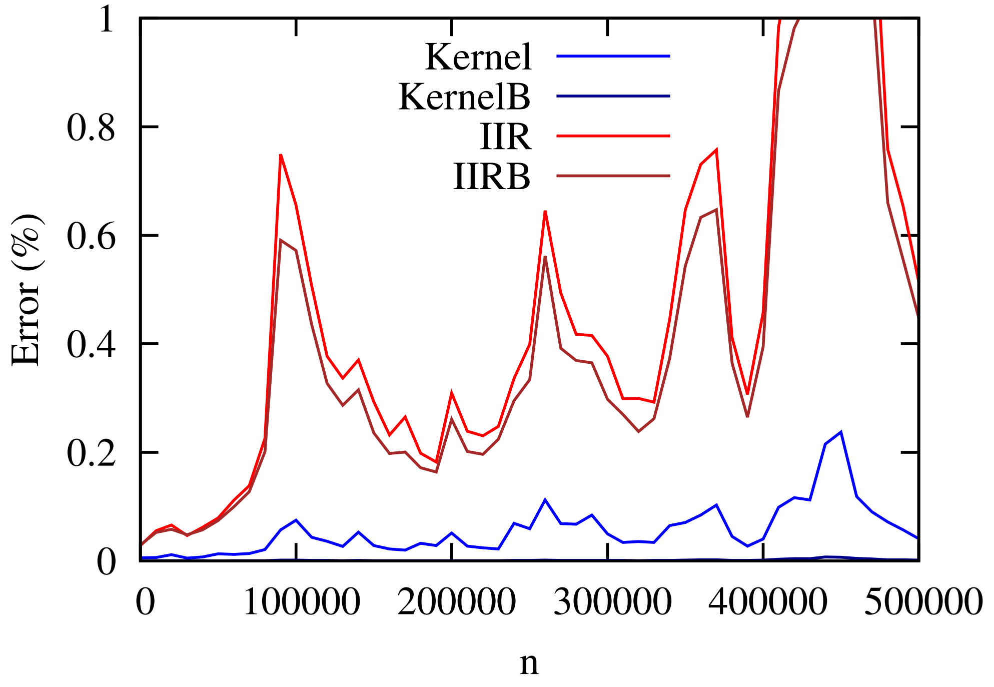
$G_{\mathrm{DD}}[n]$, $K = 256$, $P = 4$



$G_{\mathrm{DD}}[n]$, $K = 256$, $P = 6$

Figure 4: Truncation errors of 2nd order Derivatives of Gauss function with interval and order





Magnified version of the top figure

Figure 5: Error by single precision floating point calculation (Kernel: eq.(16), KernelB: eq.(22), IIR: eq.(19), IIRB: eq.(24), 2ndIIR: eq.(21), 2ndIIRB: eq.(26))