East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.161016.300517a

Vol. 7, No. 4, pp. 810-826
November 2018

Linear Regression to Minimize the Total Error of the Numerical Differentiation

Jengnan Tzeng*

Department of Mathematical Science, National Chengchi University, Taipei, No. 64, Sec. 2, ZhiNan Rd., Wenshan District, Taipei City 11605, Taiwan (R.O.C).

Received 16 October 2016; Accepted (in revised version) 30 May 2017.

Abstract. It is well known that numerical derivative contains two types of errors. One is truncation error and the other is rounding error. By evaluating variables with rounding error, together with step size and the unknown coefficient of the truncation error, the total error can be determined. We also know that the step size affects the truncation error very much, especially when the step size is large. On the other hand, rounding error will dominate numerical error when the step size is too small. Thus, to choose a suitable step size is an important task in computing the numerical differentiation. If we want to reach an accuracy result of the numerical difference, we had better estimate the best step size. We can use Taylor Expression to analyze the order of truncation error, which is usually expressed by the big O notation, that is, $E(h) = Ch^k$. Since the leading coefficient C contains the factor $f^{(k)}(\xi)$ for high order k and unknown ξ, the truncation error is often estimated by a roughly upper bound. If we try to estimate the high order difference $f^{(k)}(\xi)$, this term usually contains larger error. Hence, the uncertainty of ξ and the rounding errors hinder a possible accurate numerical derivative.
We will introduce the statistical process into the traditional numerical difference. The new method estimates truncation error and rounding error at the same time for a given step size. When we estimate these two types of error successfully, we can reach much better modified results. We also propose a genetic approach to reach a confident numerical derivative.

AMS subject classifications: 65M10, 78A48

Key words: Truncation error, leading coefficient, asymptotic constant, rounding error.

1. Introduction

In numerical computation, the total numerical error comes from two types of errors. The first one is rounding error due to the limitation of hardware so as not to represent the real number. The second one is truncation error as a result of the approximation ability of specific numerical method. It is important to control errors of numerical computation for many applications. To get the best numerical derivative, we must select the best step size

*Corresponding author. *Email address:* jengnan@math.nccu.edu.tw (J. Tzeng)

to balance rounding error and truncation error. However, there are some unknown factors in truncation error analysis, so that we could not decide the best step size.

We first give some definitions that will be used in this paper.

Definition 1.1. Let $f \in C^{\infty}(R)$, the k-th order numerical derivative of function f at x_0 that is computed by the step size h is denoted by $D^{(k)}(f, x_0, h)$.

For example, $D^{(1)}(f,x,h) = (f(x+h)-f(x))/h$ is the forward difference to approximate $f'(x)$ and $D^{(2)}(f,x,h) = (f(x+h)-2f(x)+f(x-h))/h^2$ is the central difference to approximate $f''(x)$. Because there are three types of approximation methods (forward, backward and central) and every approximation method has its specific truncation error, we use the notation $D^{(k)}_{F,n}(f,x,h)$ to indicate the numerical derivative is forward method with n-th order truncation error. Similarly, $D^{(k)}_{C,n}(f,x,h)$ indicates the central method with n-th order truncation error and $D^{(k)}_{B,n}(f,x,h)$ indicates the backward method with n-th order truncation error.

To know the error of approximation $(f(x+h)-f(x))/h \approx f'(x)$, we use Taylor expression

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)}{2}h^2, \tag{1.1}$$

where $\xi \in (x, x+h)$. Hence, we have $D^{(1)}_{F,1}(f,x,h) - f'(x) = f''(\xi)h/2 = O(h)$. The error term $f''(\xi)h/2$ contains two unknowns. One is the function $f''(x)$ and the other is ξ. We call this term $f''(\xi)h/2$ is the truncation error. To analyse the truncation error, we have $|D^{(1)}_{F,1}(f,x,h) - f'(x)| \le Kh$, where $K = \max_{\xi\in(x,x+h)} |f''(\xi)/2|$. In practice, we will assume that h is small and $f''(x)$ is a continuous function, then the value of $f''(x)$ is closed to $f''(\xi)$. To see the error of central difference to approximate $f''(x)$, we express $f(x+h)$ and $f(x-h)$ by

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f^{(3)}(x)}{6}h^3 + \frac{f^{(4)}(\xi_1)}{24}h^4 \tag{1.2}$$

and

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f^{(3)}(x)}{6}h^3 + \frac{f^{(4)}(\xi_2)}{24}h^4, \tag{1.3}$$

where $\xi_1 \in (x, x+h)$ and $\xi_2 \in (x-h, x)$. We have

$$\frac{f(x+h)-2f(x)+f(x-h)}{h^2} = f''(x) + \frac{f^{(4)}(\xi)}{12}h^2, \tag{1.4}$$

for some $\xi \in (x-h, x+h)$. Thus, we use $D^{(2)}_{C,2}(f,x,h)$ to denote $(f(x+h)-2f(x)+f(x-h))/h^2$.

The total error of $D^{(1)}_{F,1}(f,x,h)$ is also related to the rounding error. We assume that the machine epsilon is ϵ. Evaluate $f(x+h)$ and $f(x)$ will include the rounding errors, so-called e_1 and e_2, and the rounding error will be proportional to the value of f. That is

$e_1 = \pm f(x+h)\epsilon$ and $e_2 = \pm f(x)\epsilon$. Let y_1, y_2 be the numerical value of $f(x+h)$ and $f(x)$ respectively. We have

$$\begin{aligned} D_{F,1}^{(1)}(f,x,h) = \frac{y_1 - y_2}{h} &= \frac{(f(x+h)+e_1)-(f(x)+e_2)}{h} \\ &= f'(x) + \frac{f''(\xi_1)}{2}h + \frac{\pm f(x+h)\epsilon \pm f(x)\epsilon}{h} \\ &= f'(x) + \frac{f''(\xi_1)}{2}h + \frac{\pm(f(x)+f'(\xi_2)h)\epsilon \pm f(x)\epsilon}{h} \\ &= f'(x) + \frac{f''(\xi_1)}{2}h + f(x)\frac{\pm\epsilon \pm \epsilon}{h} \pm f'(\xi_2)\epsilon, \end{aligned}$$

where $\xi_1, \xi_2 \in (x, x+h)$. Thus

$$\begin{aligned} \left|D_{F,1}^{(1)}(f,x,h) - f'(x)\right| &= \left|\frac{f''(\xi)}{2}h + f(x)\frac{\pm\epsilon \pm \epsilon}{h} \pm f'(\xi_2)\epsilon\right| \\ &\le \left|\frac{f''(\xi)}{2}h\right| + \left|f(x)\frac{2}{h}\right|\epsilon + \left|f'(\xi_2)\right|\epsilon. \end{aligned} \tag{1.5}$$

Assume that function f is smooth and the machine epsilon is pretty small, the last term can be omitted from Eq. (1.5).

We define the error function $E(h) = |f''(\xi)h/2| + |2\epsilon f(x)|/h$. To minimize $E(h)$ by h, we have $E'(h) = |f''(\xi)/2| - |2\epsilon f(x)|/h^2 = 0$. Thus, $h^* = 2\sqrt{\epsilon|f(x)/f''(\xi(h))|}$ will be reach a better-expected numerical result. Since we don't know the exact ξ, we can roughly estimate $f''(\xi)$ by the central difference $D_{C,2}^{(2)}(f,x,h)$. That's why we denote ξ by a function of h, i.e., $\xi(h)$. Here, we call the solved step size h^* by the best modified step size. Notice that the unknown ξ is determined by the given h and the best modified step size h^* is determined by some ξ near h. However, these equations are only of theoretical value and cannot be used practically to determine h^* because we usually don't have any information about the high-order derivatives. The step size h^* does not minimize the real error, but only its upper bound [7]. We will look for a better way to explore the best estimation of $f''(\xi)$ for error control.

Another approach to estimate $f''(\xi)$ is adjust the step size h to $2h$ [2]. With the new step size, we have

$$D_{F,1}^{(1)}(f,x,2h) - f'(x) = f''(\xi_2)h, \tag{1.6}$$

where $\xi_2 \in (x, x+2h)$. If we assume that h is sufficiently small, $\xi \approx \xi_2$ and $f''(x)$ is continuous, then we have

$$D_{F,1}^{(1)}(f,x,2h) - D_{F,1}^{(1)}(f,x,h) \approx \frac{f''(\xi)}{2}h.$$

Thus, $f''(\xi)$ is estimated by $2(D_{F,1}^{(1)}(f,x,2h) - D_{F,1}^{(1)}(f,x,h))/h$.

After we estimate $f''(\xi)$ successfully, we can update the truncation error to Eq. (1.1). Then we have

$$D_{F,1}^{(1)}(f,x,h) - f'(x) = \frac{f''(\xi)}{2}h = D_{F,1}^{(1)}(f,x,2h) - D_{F,1}^{(1)}(f,x,h).$$

Thus, $2D^{(1)}_{F,1}(f,x,h)-D^{(1)}_{F,1}(f,x,2h)$ becomes a new approximation of $f'(x)$. Actually, it is a second order approximation of $f'(x)$. This implement is called Richardson's extrapolation method. The general form of Richardson's extrapolation method is

$$D^{(1)}_{F,n+1}(f,x,h)=\frac{2^nD^{(1)}_{F,n}(f,x,h)-D^{(1)}_{F,n}(f,x,2h)}{2^n-1}, \tag{1.7}$$

for $n\geq 1$. One can modify the step size and use the corresponding linear combination to obtain a high order approximation of numerical derivative.

Notice that the Richardson's extrapolation method is derived by adjusting the step size from h to $2h$. Actually, we can estimate $f''(\xi)$ by any th for $t\neq 0$ or 1. That is

$$D^{(1)}_{F,1}(f,x,th)-f'(x)=t\frac{f''(\xi_t)}{2}h, \tag{1.8}$$

where $\xi_t\in(x,x+th)$ and then $f''(\xi)$ is estimated by $2(D^{(1)}_{F,1}(f,x,2h)-D^{(1)}_{F,1}(f,x,h))/((t-1)h)$. Since t is chosen randomly to be near 1 but not equal to 1, the corresponding ξ_t should be variant randomly and the leading coefficient $f''(\xi_t)/2$ can be considered as a random variable. Therefore, we can try to use the statistical method to estimate the value of $f''(\xi)/2$.

2. Linear Regression Method

Let $D^{(1)}_{F,1}(f,x,h)=(f(x+h)-f(x))/h$ be the forward method of the first order numerical differential. From Section 1, we have

$$D^{(1)}_{F,1}(f,x,h)-f'(x)=\frac{f''(\xi)}{2}h+f(x)\frac{c\epsilon}{h}+f'(\xi_2)\epsilon$$

for some $\xi,\xi_2\in(x,x+h)$ and some constant c. For the sake of convenience, we merge the last two terms as E by assuming $|f'(\xi_2)|\epsilon$ is a pretty small number. We rewrite this equation by

$$D^{(1)}_{F,1}(f,x,h)-f'(x)\approx\frac{f''(\xi)}{2}h+f(x)\frac{E}{h}. \tag{2.1}$$

For $t\neq 0$ or 1, we also have

$$D^{(1)}_{F,1}(f,x,th)-f'(x)\approx t\frac{f''(\xi_t)}{2}h+f(x)\frac{E_t}{th} \tag{2.2}$$

for some $\xi_t\in(x,x+th)$ and other rounding error term E_t. When ξ and ξ_t are near, and f'' is continuous, we can assume $f''(\xi_t)\approx f''(\xi)$. We minus Eq. (2.1) by Eq. (2.2), then we have

$$D^{(1)}_{F,1}(f,x,th)-D^{(1)}_{F,1}(f,x,h)\approx\frac{tf''(\xi_t)-f''(\xi)}{2}h+f(x)\frac{tE-E_t}{th}.$$

Although the rounding error term has no continuity property, we also merge $tE-E_t$ to $(t-1)E$ plus an error term. Thus, we have

$$D^{(1)}_{F,1}(f,x,th)-D^{(1)}_{F,1}(f,x,h)=\frac{(t-1)f''(\xi)}{2}h+f(x)\frac{(1-t)E}{th}+\epsilon_t,$$

where ϵ_t collects all error terms and is considered a random variable. If we use different t_i to replace t, we have

$$y_i = D_{F,1}^{(1)}(f,x,t_i h) - D_{F,1}^{(1)}(f,x,h) = \frac{f''(\xi)}{2} h v_i + f(x)\frac{E}{h} w_i + \epsilon_i, \tag{2.3}$$

where $v_i = (t_i - 1)$, $w_i = (1-t_i)/t_i$ for $i = 1, \cdots, n$.

Unlike the methods to optimize the best step size [8], we will introduce a statistical approach that use the generated data to estimate the expectation values of truncation error and rounding error by the given step size h. If we have a certain number (usually greater than 2) of Eq. (2.3) for different t_i, we can regard (v_i, w_i, y_i) as data. We would like to use linear regression method to find the linear solution $y = \alpha v + \beta w$ that fits these data. Then the best solution of α and β is the expectation value of $f''(\xi)h/2$ and $f(x)E/h$ respectively. It is probable that the numerical derivative will become better after updating this truncation error $f''(\xi)h/2$ and the rounding error $f(x)E/h$.

Using the linear regression method to yield to the best α and β is very easy. From Eq. (2.3), we define $A \in M_{K,2}(R)$, where $A_{i,1} = v_i = t_i - 1, A_{i,2} = w_i = (1-t_i)/t_i$ for $i = 1, \cdots, K$ and $Y = (y_1, \cdots, y_K)$. The least square solution of Eq. (2.3) is $(\alpha, \beta)^T = (A^T A)^{-1} A^T Y$. After obtaining the solution (α, β), we can compute the residue r which is defined by

$$r = \sqrt{(Y^T - (\alpha,\beta)A^T)(Y - A(\alpha,\beta)^T)}. \tag{2.4}$$

This residue r shows the fitness of the linear regression. Then the new numerical derivative $D_*^{(1)}(f,x,h)$ modified by estimating $f''(\xi)h/2$ and $f(x)E/h$ is defined by

$$D_*^{(1)}(f,x,h) := D_{F,1}^{(1)}(f,x,h) - \alpha - \beta.$$

We can extend this method to higher-order derivatives. The general form of the total numerical error is

$$D_{\cdot,n}^{(k)}(f,x,h) - f^{(k)}(x) = g(\xi)h^n + \frac{E}{h^k}, \tag{2.5}$$

where k is the order of derivative and n is the asymptotic rate. The dot symbol "$\cdot$" in the subscript of $D_{\cdot,n}^{(k)}(f,x,h)$ means that it could be "F"(forward method), "C" (central method), or "B" (backward method). The first term of the right hand side of Eq. (2.5) is truncation error and $g = f^{(k+p)}$ for some $p \geq 1$. The second term is rounding error and it usually contains the factor h^{-n}. If we adjust h to $t_i h$, we have

$$D_{\cdot,n}^{(k)}(f,x,t_i h) - f^{(k)}(x) = g(\xi_{t_i}) t_i^n h^n + \frac{E_i}{t_i^k h^k}. \tag{2.6}$$

Eq. (2.6) minus equation (2.5) and collect the error term to ϵ_i, we have

$$D_{\cdot,n}^{(k)}(f,x,t_i h) - D_{\cdot,n}^{(k)}(f,x,h) = g(\xi)h^n(t_i^n - 1) + \frac{E}{h^k}\frac{1-t_i^k}{t_i^k} + \epsilon_i. \tag{2.7}$$

We set

$$y_i = D^{(k)}_{\cdot,n}(f,x,t_ih) - D^{(k)}_{\cdot,n}(f,x,h) = g(\xi)h^n v_i + \frac{E}{h^k}w_i + \epsilon_i, \tag{2.8}$$

where $v_i = t_i^n - 1$, $w_i = (1-t_i^k)/t_i^k$ and $i = 1,\cdots,N$ for some $N > 2$. Then y_i, v_i and w_i form a linear system with error term ϵ_i. Therefore, we can easily solve the expectation value of $g(\xi)h^n$ and E/h^k by linear regression method.

Similarly, the matrix form of this linear system can be expressed by $A \in M_{N,2}(R)$, where $A_{i,1} = v_i = t_i^n - 1$, $A_{i,2} = w_i = (1-t_i^k)/t_i^k$ for $i = 1,\cdots,N$ and $Y = (y_1,\cdots,y_N)$. The minimal square solution of Eq. (2.8) is $(\alpha,\beta)^T = (A^TA)^{-1}A^TY$. The solution α and β is the expectation value of $g(\xi)h^n$ and E/h^k respectively. We can also compute the residue r by $r = \sqrt{(Y^T-(\alpha,\beta)A^T)(Y-A(\alpha,\beta)^T)}$. Then we can update $g(\xi)h^n$ and E/h^k to the numerical high-oder derivative that is defined by

$$D^{(p)}_*(f,x,h) := D^{(p)}_{\cdot,n}(f,x,h) - \alpha - \beta, \tag{2.9}$$

when the residue r is acceptable.

3. Distribution of the Scaling Factor t

In Section 2, we show that we can modify the Richardson's extrapolation with different scaling factors t_i to generate a set of data (v_i, w_i, y_i) as Eq. (2.8). Using this data, we can estimate truncation error and rounding error by linear regression method. Then we can modify the numerical derivative by the regression coefficient. Notice that we are not discussing how to find the optimal step size t_ih or the optimal scaling factor t_i. Some t_ih make $D^{(p)}_{\cdot,n}(f,x,t_ih)$ good and some t_ih make $D^{(p)}_{\cdot,n}(f,x,t_ih)$ bad. However, we have no idea whether $D^{(p)}_{\cdot,n}(f,x,t_ih)$ is good or not, because we need a prior true solution. Up to now, we only know t_i is considered a random number. Therefore, what is the best distribution of t_i is a task.

In numerical analysis, errors can be described by two characteristics. One is accuracy and the other is precision [3]. Accuracy refers to how closely a computed result agrees with the true solution. Precision refers to how closely individual computed results agree with each other. When we have no idea about the true solution, we never know the accuracy of our computed results. However, we can compute the numerical derivative by different step size t_ih. Then we have resource to compute the precision. This is the key point for us to look for the appropriate distribution of the scaling factor t_i.

Let us focus on what is the better form of the distribution of t_i now. First, t_i should not have the same sign only. It is well known that the central method $D^{(1)}_{c,2}(f,x,h)$ is usually better than the forward method $D^{(1)}_{f,1}(f,x,h)$ and the backward method $D^{(1)}_{b,1}(f,x,h)$. If t_i contains two signs, then we have chance to obtain the result as central method. Therefore, it is natural for us to design the distribution of t_i as a Gaussian mixture distribution $t_i \sim 0.5N(-\mu,\sigma)+0.5N(\mu,\sigma)$, where $N(\mu,\sigma)$ is normal distribution with mean μ and standard deviation σ. Around one half samples t_i come from $N(-\mu,\sigma)$, and the rest comes from $N(\mu,\sigma)$.

The further problem is how we can find the better μ and σ, so that the distribution of t_i reaches a better performance. To solve this problem, we can design a genetic approach to seek a better distribution.

For each distribution $G_j(\mu_j,\sigma_j) = 0.5N(-\mu_j,\sigma_j) + 0.5N(\mu_j,\sigma_j)$, we can compute the coefficient of the linear regression for Eq. (2.8). Then we can compute the corresponding residue r_j and the modified numerical derivative $D_*^{(k)}(f,x,h)_j$. Instead of precision, our concern is the fitness of regression. That is, the smaller r_j should lead to the better $D_*^{(p)}(f,x,h)_j$. Thus, we can design the genetic approach to seek the better distribution $G(\mu_j,\sigma_j)$ by the following.

Given a step size h, we start from an initial μ_0 and σ_0. We sample t_i from the mixture distribution $G_0(\mu_0,\sigma_0)$, then we compute the related data (v_i,w_i,y_i). We solve Eq. (2.8). Then we update the expectation value of truncation error and rounding error to obtain $D_*^{(k)}(f,x,h)_0$ and compute the residue r_0.

Assume we have (μ_j,σ_j), r_j and $D_*^{(k)}(f,x,h)_j$ for some $j \geq 0$. We perturb (μ_j,σ_j) twice and obtain two parameters $(\mu_1^{(j)},\sigma_1^{(j)})$ and $(\mu_2^{(j)},\sigma_2^{(j)})$. For each parameter $(\mu_\ell^{(j)},\sigma_\ell^{(j)})$, $\ell = 1,2$, we have the mixture distribution $G_j(\mu_\ell^{(j)},\sigma_\ell^{(j)})$. We randomly choose t_i^ℓ from $G_j(\mu_\ell^{(j)},\sigma_\ell^{(j)})$, generate data $(v_i^\ell,w_i^\ell,y_i^\ell)$, compute the regression coefficient, and compute the related residue $\tilde{r}_\ell$ and the modified numerical derivative $D_{*,\ell}$ by our method. If $\tilde{r}_1 < \tilde{r}_2$, then we set $r_{j+1} = \tilde{r}_1$ and the modified numerical derivative $D_*^{(k)}(f,x,h)_{j+1}$ is $D_{*,1}$; else we set $r_{j+1} = \tilde{r}_2$ and the modified numerical derivative $D_*^{(k)}(f,x,h)_{j+1}$ is $D_{*,2}$.

We can generate a sequence $D_*^{(k)}(f,x,h)_j$ for $j = 0,1,2,\cdots$. We can compute the relative error between $D_*^{(k)}(f,x,h)_{j+1}$ and $D_*^{(k)}(f,x,h)_j$ for $j \geq 1$. If the relative error $RE_j = |D_*^{(k)}(f,x,h)_{j+1} - D_*^{(k)}(f,x,h)_j|/|D_*^{(k)}(f,x,h)_j|$ is smaller than a tolerance τ, then we stop the iteration process. We can also set a maximal iteration number M to stop the iteration process. The final $D_*^{(k)}(f,x,h)_j$ is output. With this genetic approach, we can find a better result than the traditional method. The experimental result will be shown in the next section.

4. Experimental Result

Two examples are to be shown that our implements are better than traditional methods. One example is the first derivative $D_{F,1}^{(1)}(f,x,h)$ and the other is the second derivative $D_{C,2}^{(2)}(f,x,h)$. We set $f(x) = 3e^x + 10\sin(x)$ and valuate the first derivative and the second derivative at $x_0 = \pi/4$. Both of them are computed under the condition that we have the true solution of function derivative.

In Table 1, the first column is the step size h that starts from 10^{-3} to 10^{-11}. These step sizes are chosen subjectively. The second column is the step size h_b, which is derived by solving the optimal step size to minimize the total error in Eq. (2.5) by the given h. The third column is the absolute error w.r.t the step size h, the fourth column is the absolute error w.r.t the step size h_b, and the last column is the absolute error computed by our method. In Table 1, we use 8 samples of $t_i h_b$ to form the linear system and 4 scaling factors t_i

Table 1: Example of $D_{F,1}^{(1)}(f,x,h)$ for $f(x)=3e^x+10\sin(x)$ at $x=\pi/4$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-3}	1.57104988608e-07	0.000245695130346	4.00592519156e-08	4.57891502492e-1 1
10^{-4}	1.5710483124e-07	2.45621961596e-05	3.62021062017e-08	9.31006383098e-11
10^{-5}	1.57109518407e-07	2.45605701998e-06	4.57071038795e-08	3.00861557889e-11
10^{-6}	1.57257450482e-07	2.4431511747e-07	3.6117695501e-08	9.89590631661e-11
10^{-7}	1.84735675792e-07	2.22705125452e-08	4.43045458098e-08	1.52837742462e-11
10^{-8}	2.61255698159e-08	1.73128739789e-07	5.27110071147e-08	4.06359390581e-11
10^{-9}	1.84735675792e-09	1.77184989525e-06	4.07631825183e-07	3.10553360805e-09
10^{-10}	1.84735675792e-10	2.13117751287e-05	3.29233208518e-06	8.78334240895e-08
10^{-11}	1.84735675792e-11	0.000216711027463	7.06020047971e-05	5.87577741129e-07

Table 2: Example of $D_{F,1}^{(1)}(f,x,h)$ for $f(x)=\ln(x)$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-3}	1.17817305704e-09	1.20983583057	1.7024447132e-06	1.02329522633e-09
10^{-4}	1.17890304578e-09	0.12458488961	1.48504782516e-06	1.7430608068e-09
10^{-5}	1.17891034013e-09	0.0124958348835	1.58702189168e-06	2.79953837889e-10
10^{-6}	1.17891036526e-09	0.00124995847557	1.52275192988e-06	2.33555397244e-11
10^{-7}	1.17891036526e-09	0.000124999682782	1.52275192988e-06	8.72248051564e-10
10^{-8}	1.17781246406e-09	1.25163268194e-05	1.64273784975e-06	5.4116355841e-10
10—9	1.14193155157e-09	1.19205196825e-06	1.45973995558e-06	1.4272103499e-09

from the normal distribution $N(40,0.1)$ and the others from $N(-40,0.1)$. The parameter $(\mu,\sigma)=(\pm 40,0.1)$ is tuned by the prior true solution.

We can see that the best subjective step size is $h=10^{-7}$. It makes the minimal error in the level of 10^{-8}. The best modified step size is $h^*=1.57257450482\times 10^{-7}$ and its corresponding error is $3.6117695501\times 10^{-8}$. The best result of our method makes the error to be $1.52837742462\times 10^{-11}$. Compare the third column, the fourth column and the last column of Table 1. We can see that our proposed method makes 3 significant figures improvement.

To avoid only show the example of smooth function, we will show some examples of derivative near the steep region.

Tables 2, 3 and 4 show the first derivative of $\ln(x)$, $\sqrt{x}$ and $\arctan(x)$ at $x=0.02$, respectively. Some data are not shown in the table, for example the $h=10^{-10}$ and 10^{-11} in Table 2, because we meet the case of overflow. We can see that our method is pretty good near the steep region. In the example of $f(x)=\ln(x)$, our method have 5 figures improvement compared with the best modified step size h^*; in the case of $f(x)=\sqrt{x}$, we have 4 figures improvement and in the case of $f(x)=\arctan(x)$, we have also 4 figures improvement.

In Table 5, we use the same function and the same point x_0 in the Table 1 to demonstrate the numerical second derivative. The subjective step size comes from 10^{-1} to 10^{-8}. The modified step size h^* is solved by Eq. (2.5) for $D_{C,2}^{(2)}(f,x,h)$ method. We still use 8 samples to construct the linear system and 4 t_i come from the normal distribution $N(20,0.8)$ and the

Table 3: Example of $D_{F,1}^{(1)}(f,x,h)$ for $f(x)=\sqrt{x}$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-3}	1.19162689545e-09	0.0431226813479	5.65993558688e-08	5.46602763052e-11
10^{-4}	1.19208823886e-09	0.00440840324538	6.58976992973e-08	9.82023351526e-11
10^{-5}	1.19209284989e-09	0.000441831285876	5.0935365703e-08	7.83861864306e-11
10^{-6}	1.19209305203e-09	4.41930524246e-05	4.50962009957e-08	7.01447788742e-11
10^{-7}	1.19207901455e-09	4.41953461205e-06	4.25713433394e-08	9.18776166259e-12
10^{-8}	1.19294089849e-09	4.39940180286e-07	5.80553560781e-08	7.52775619617e-12
10^{-9}	1.22820782709e-09	4.02598914206e-08	5.21358507477e-08	2.56195065163e-11
10^{-10}	2.12731835878e-10	1.81784713504e-07	1.33002697744e-08	3.67555763603e-10
10^{-11}	2.12731835878e-11	1.84711925044e-06	1.43772399319e-07	2.97564994867e-09

Table 4: Example of $D_{F,1}^{(1)}(f,x,h)$ for $f(x)=\arctan x$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-3}	2.10804589852e-08	2.03165235707e-05	5.08672881594e-10	1.25155441566e-12
10^{-4}	2.10804485548e-08	2.00172629128e-06	4.81062634172e-10	3.9179770539e-13
10^{-5}	2.10804367519e-08	1.9987370059e-07	5.50319900761e-10	8.85291839836e-13
10^{-6}	2.10795310111e-08	1.99863454675e-08	6.08984640493e-10	8.71525074331e-14
10^{-7}	2.1007628945e-08	2.01114080944e-09	4.24490997908e-10	1.59650070941e-13
10^{-8}	1.59989335537e-08	3.45806272506e-10	4.95699037373e-10	5.57109913757e-13

Table 5: Example of $D_{C,2}^{(2)}(f,x,h)$ for $f(x)=3e^x+10\sin(x)$ at $x=\pi/4$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-1}	0.000321311849431	0.0113756208619	1.39763453255e-07	3.82777476382e-10
10^{-2}	0.000321307189585	0.000113757590071	1.38210512257e-07	6.61025123527e-11
10^{-3}	0.000321398951575	1.14063431056e-06	1.21283529497e-07	1.19108611862e-11
10^{-4}	0.000252970977166	1.56532621531e-07	7.2418397612e-08	3.65634245103e-10
10^{-5}	1.6917188842e-05	2.94664204717e-05	2.23234017693e-06	6.79091642919e-08
10^{-6}	2.01180413368e-06	0.00095317197696	0.000545098216634	1.78540298951e-06
10^{-7}	1.92216375885e-07	0.135956291771	0.0585226207873	0.000174089324108
10^{-8}	2.01180413368e-08	18.2547960537	4.88015933617	0.0369473845101

other 4 t_i come from $N(20,0.8)$. The parameter is also tuned by the prior true solution. We can see that the best subjective step size is $h=10^{-4}$ and the best modified step size is $h^*=0.000252970977166$. Compare the third column, the fourth column and the last column of Table 5, we can see that our method makes at least 2 significant figures improvement.

As we have demonstrated before, we compare our method by the second order difference in the $\ln(x)$, $\sqrt{x}$ and $\arctan(x)$. The results are shown from the Table 6 to Table 8. We can see the ability of our method is good in second order derivative as in first order derivative. Even though the given step size h is far away the best modified step size h^*, the modification ability of our method is very stable, i.e., errors of our method are almost in the same level.

From Tables 1 and 8, the good results of our method are fine tuned by the mean and standard deviation of the distribution of t_i. This can be done only when we know the true

Table 6: Example of $D^{(2)}_{C,2}(f,x,h)$ for $f(x)=\ln(x)$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-2}	5.77448023003e-06	376.820724518	0.000111786614525	1.09695520223e-07
10^{-3}	5.77448023003e-06	3.13021811843	0.000111786614525	1.82426902029e-07
10^{-4}	5.77448023003e-06	0.0312505518705	0.000111786614525	3.26532699546e-07
10^{-5}	5.77448023003e-06	0.000313432337407	0.000111786614525	1.18005118566e-07
10^{-6}	5.77448023003e-06	0.000206850927498	0.000111786614525	4.10050233768e-08
10^{-7}	5.77448023003e-06	0.000206850927952	0.000111786614525	6.89792614139e-08
10^{-8}	5.77448023003e-06	4.66314355435	0.000111786614525	2.51523488259e-07
10^{-9}	5.77448023003e-06	164.5352591	0.000111786614525	2.0624293029e-07
10^{-10}	5.77448023003e-06	2500.0	0.000111786614525	2.47273874265e-07

Table 7: Example of $D^{(2)}_{C,2}(f,x,h)$ for $f(x)=\sqrt{x}$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-2}	6.53066704079e-06	7.98796952899	3.05024778413e-06	2.73003735174e-08
10^{-3}	6.53066704079e-06	0.0691441740839	3.05024778413e-06	1.15812923696e-08
10^{-4}	6.53066704079e-06	0.000690543098088	3.05024778413e-06	1.790100157e-08
10^{-5}	6.53066704079e-06	6.76506388686e-06	3.05024778413e-06	7.45875183839e-09
10^{-6}	6.53066704079e-06	2.32109577638e-05	3.05024778413e-06	5.94624793848e-10
10^{-7}	6.53066704079e-06	0.00205845721342	3.05024778413e-06	7.98226551524e-09
10^{-8}	6.53066704079e-06	0.125617190618	3.05024778413e-06	2.58509658124e-09
10^{-9}	6.53066704079e-06	5.12162080143	3.05024778413e-06	5.66281244119e-09
10^{-10}	6.53066704079e-06	2863.94590921	3.05024778413e-06	5.96089932969e-09

Table 8: Example of $D^{(2)}_{C,2}(f,x,h)$ for $f(x)=\arctan x$ at $x=0.02$.

step size h	modified step size h^*	Error by h	Error by h^*	Error of our method
10^{-2}	0.00014523463664	3.99161044438e-06	7.17035285314e-10	3.34539340674e-12
10^{-3}	0.00014523463664	3.99163509052e-08	7.17035285314e-10	2.1729285038e-12
10^{-4}	0.00014523463664	3.64655652896e-10	7.17035285314e-10	4.76804706828e-12
10^{-5}	0.00014523463664	4.43912100204e-08	7.17035285314e-10	1.96585803192e-13
10^{-6}	0.00014523463664	3.47914369246e-06	7.17035285314e-10	8.01418653662e-12
10^{-7}	0.00014523463664	0.000277565452897	7.17035285314e-10	1.54441737177e-12
10^{-8}	0.00014523463664	0.0294209198493	7.17035285314e-10	9.32413868338e-12
10^{-9}	0.00014523463664	0.0399680191898	7.17035285314e-10	4.2170017478e-12
10^{-10}	0.00014523463664	346.904727176	7.17035285314e-10	3.37530697836e-12

solution. Practically, we don't know the true solution and we don't know the direction to tune the best distribution of t_i. In the second part of this section, we will use the genetic approach in Section 3 to seek the suitable distribution of the scaling factor t automatically.

Table 9 is the performance of first order derivative by the genetic approach of our method. The first column is the given step size h. They are 10^{-p}, where $p=3,\cdots,11$. We all start from the initial parameter $(\mu_0,\sigma_0)=(3,0.1)$. We fix the $\sigma_k=\sigma_0$ for all $k\geq 1$ and only perturb the μ_k in this example. The tolerance to stop the process is set to be 10^{-11}. And we set 600 as the maximal number of iterations. The second column is the number of iterations that stop the processing, the third column is the error of our method, the fourth

Table 9: Example of $D_{F,1}^{(1)}(f,x,h)$ for $f(x)=3e^x+10\sin(x)$ at $x=\pi/4$ for automatically adjust t_i.

step size h	iterations	Error of our method	final μ	residue
10^{-3}	600	1.89549453466e-07	-1.53532444528	1.45417999735e-08
10^{-4}	27	1.9543318075e-09	-1.50321747158	2.26930663538e-10
10^{-5}	4	2.3944224381e-10	5.53938472154	6.75506067507e-12
10^{-6}	11	9.98845450795e-12	8.55463039103	6.07647290642e-11
10^{-7}	43	3.42723183167e-10	13.4489517888	8.56695877758e-10
10^{-8}	268	1.04719788396e-09	-20.9218048074	2.4304803334e-09
10^{-9}	600	3.30193739018e-08	13.5533853948	3.66736226511e-08
10^{-10}	600	1.03510055638e-07	41.7581081724	4.49121450386e-07
10^{-11}	600	1.08822671052e-06	29.7910215213	3.16557370495e-06

Table 10: Example of $D_{C,2}^{(2)}(f,x,h)$ for $f(x)=3e^x+10\sin(x)$ at $x=\pi/4$ for automatically adjust t_i.

step size h	iterations	Error of our method	final μ	residue
10^{-1}	600	4.07661652468e-06	-2.26213894994	3.22595339162e-08
10^{-2}	60	7.21258608394e-11	-0.424114283318	2.11850032521e-10
10^{-3}	12	1.97531158097e-10	11.5473654127	7.54039386063e-12
10^{-4}	200	1.18793704318e-08	19.1288807055	2.26839134415e-10
10^{-5}	600	2.05409143861e-07	56.6157962034	1.12364129007e-09
10^{-6}	600	7.2792357963e-06	59.4761465819	5.18360008963e-07
10^{-7}	600	0.00737921301028	33.0407040331	5.23379975084e-05
10^{-8}	600	1.20287020036	48.3163185335	0.00586809520932

column is the μ of the final mixture distributions, and the last column is the final residue.

When the initial step size is near the best step size of the traditional method, we can see that the processing stops soon and the result is good as we demonstrate in Table 1. When the initial step size is far from the best step size of the traditional method, the processing can not stop automatically and the result is worse than the experiments in Table 1. There is one significant figure difference or so. We also see that this genetic processing has the ability to self-adjust the parameter μ. If we set the initial step size h as the best step size and see the processing stop automatically in a few iterations, then we can trust the output result.

We are interested in whether the residue is a signal to control the accuracy. In Table 9, we can compute the correlation coefficient between the absolute error (column three) and the residue (column five). The correlation coefficient is 0.98. That is, the residue is actually a signal to control the accuracy.

Table 10 is the example of the self-adjust processing for the second derivative. We use the setting $h=10^{-p}$ for $p=1,\cdots,8$. The tolerance is set to be 10^{-10}. The other settings are the same as in the example of Table 9. Obviously, when the step size h falls on 10^{-2} to 10^{-4}, the processing stops before it reaches the maximal iterations. Therefore, the optimal solution will appear in these three outputs. We can select the output that has the smallest residue. That is the case of $h=10^{-3}$. This solution is one significant feature difference than we have shown in Table 1. This characteristic is the same as the example in the first derivative. From Tables 9 and 10 we can see that the self-adjust processing

works pretty well without any prior knowledge. In Table 10, we are also interested in the relationship between the accuracy and residue. The correlation coefficient between column 3 and column 5 is 0.85. That is, they are highly relative to each other. This gives us confidence to select the output with small residue.

The last part is to show our method works well to general functions. To show the general performance of our method, we define the smooth function $f_{a,b}(x)$ as $ae^x + b\sin(x)$, where a and b are random variables from normal distribution $N(3, 0.5)$ and $N(10, 1)$ respectively. We set the domain of $x \in [0.5\pi, 1.5\pi]$ and choose randomly 64 functions of $f_{a,b}$ to obtain the average results for the data presentation.

As we have shown before, the best initial h is chosen by solving the minimization of the total error in Eq. (2.6). For example, the initial h is chosen by $2\sqrt{|\frac{f(x)}{D^{(2)}_{C,2}(f,x,h)}|\epsilon}$ in the case of $D^{(1)}_{F,1}(f,x,h)$, h is chosen by $\sqrt[3]{|\frac{3f(x)}{D^{(3)}_{F,1}(f,x,h)}|\epsilon}$ in the case of $D^{(1)}_{C,2}(f,x,h)$, and $h = \sqrt[4]{|\frac{48f(x)}{D^{(4)}_{C,2}(f,x,h)}|\epsilon}$ in the case of $D^{(2)}_{C,2}(f,x,h)$.

We use the same approach and the same setting of all the parameter in Tables 9 and 10 to compute the first, second and fourth derivative by our method. Then we compare these results with those of the traditional methods. Since one half the scaling factor t_i in our method is sampled from the distribution $N(\mu,\sigma)$ and the other half from $N(-\mu,\sigma)$, we can consider the new method uses information of high order numerical derivative, like the central difference. Therefore, we also compare our method to the relative high order numerical derivative that is derived by Richardson' s extrapolation.

We first compare the first derivative between forward method, central method, the second order forward method derived by Richardson's extrapolation and our method. The result is shown in Fig. 1. From top to bottom, the black line is the error of the traditional forward method $D^{(1)}_{F,1}(f,x,h)$ for first derivative. All values are the average of the numerical derivative of 64 random functions $f_{a,b}$. We also take the average by log_{10} for showing the result clearly. The blue line is the error of $D^{(1)}_{F,2}(f,x,h)$ that is derived from $D^{(1)}_{F,1}(f,x,h)$ by Richardson's extrapolation. The third green line is the error of the central difference $D^{(1)}_{C,2}(f,x,h)$. The last red line is the error of our method. We can see that our method is the best one. We improve about three significant figures to the forward method $D^{(1)}_{F,1}(f,x,h)$ and about two significant figures to the central difference $D^{(1)}_{C,2}(f,x,h)$.

The next example shown in Fig. 2 is the comparison of the second derivative for the central method $D^{(2)}_{C,2}(f,x,h)$, the high order method derived by Richardson's extrapolation $D^{(2)}_{C,4}(f,x,h)$ and our method. All parameters are the same as we have shown in Table 10. From top to bottom, the black line is the error of $D^{(2)}_{C,2}(f,x,h)$, the blue line is the error of $D^{(2)}_{C,4}(f,x,h)$ and the red line is for our method. We can see that our method is still the best one. We improve about four significant figures than the traditional method $D^{(2)}_{C,2}(f,x,h)$ and improve about two significant figures than the high order method $D^{(2)}_{C,4}(f,x,h)$.

The next example shown in Fig. 3 is the comparison of the fourth derivative for the

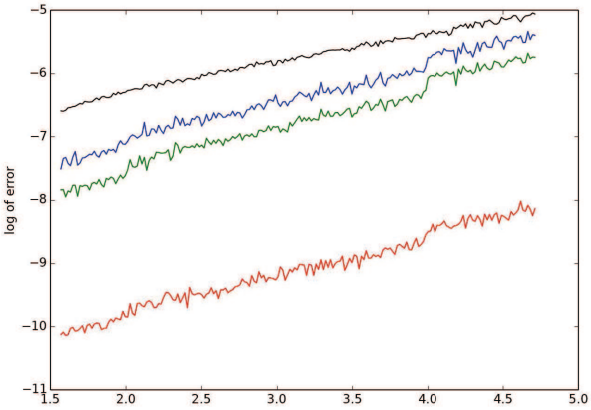

Figure 1: Comparison between the numerical first derivative and our method, where $f(x) \approx 3e^x + 20\sin(x)$. The black line is the error of $D^{(1)}_{F,1}(f,x,h)$, the blue line is $D^{(1)}_{F,2}(f,x,h)$, the green line is $D^{(1)}_{C,2}(f,x,h)$, and the red line is our method.

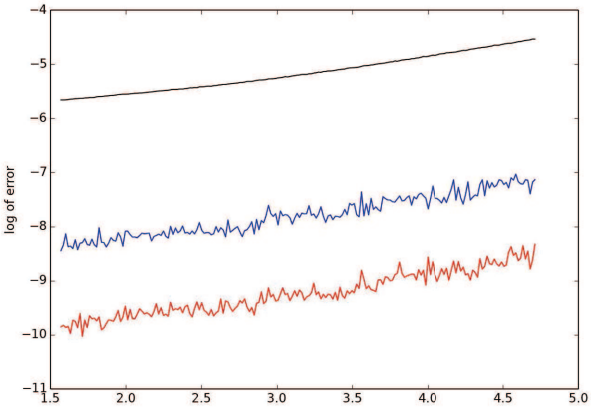

Figure 2: Comparison between the numerical second derivative and our method, where $f(x) \approx 3e^x + 20\sin(x)$. The black line is the traditional method $D^{(2)}_{C,2}(f,x,h)$, the blue line is $D^{(2)}_{C,4}(f,x,h)$, and the red line is our method.

central method $D^{(4)}_{C,2}(f,x,h)$, the high order method derived by Richardson's extrapolation $D^{(4)}_{C,4}(f,x,h)$ and our method. All parameters are the same as we have shown in Table 10. From top to bottom, the black line is the error of $D^{(4)}_{C,2}(f,x,h)$, the blue line is the error of $D^{(4)}_{C,4}(f,x,h)$ and the red line is for our method. We can see that our method is still the best one. We improve about three significant figures than the traditional method $D^{(4)}_{C,2}(f,x,h)$ and improve about one significant figures than the high order method $D^{(4)}_{C,4}(f,x,h)$.

The previous example using the function form $f(x) = ae^x + b\sin(x)$, where $a \sim$

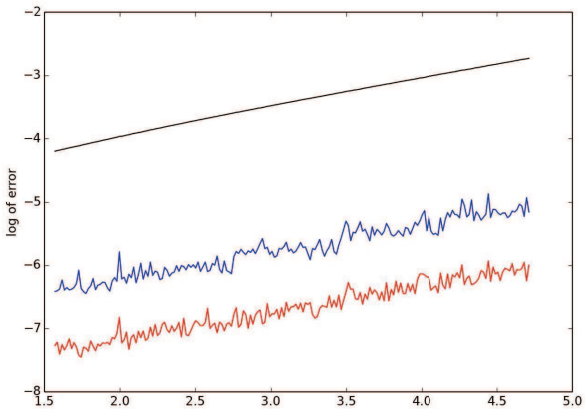

Figure 3: Comparison between the numerical fourth derivative and our method, where $f(x) \approx 3e^x + 20\sin(x)$. The black line is the traditional method $D^{(4)}_{C,2}(f,x,h)$, the blue line is $D^{(4)}_{C,4}(f,x,h)$, and the red line is our method.

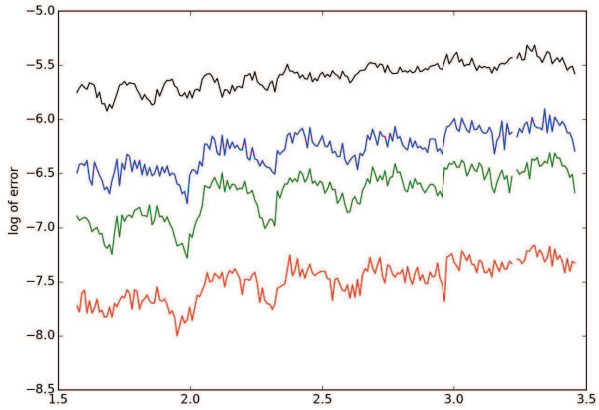

Figure 4: First order derivative comparison, where $f(x) \approx 3e^x + \sin(20x)$.The black line is the error of $D^{(1)}_{F,1}(f,x,h)$, the blue line is $D^{(1)}_{F,2}(f,x,h)$,the green line is $D^{(1)}_{C,2}(f,x,h)$, and the red line is our method.

$N(3,0.5)$ and $b \sim N(10,1)$, respectively. This function form is a smooth function. We are interested in how performance is when the function is oscillating. We use the function form $f(x) = ae^x + \sin(bx)$, where $a \sim N(3,0.5)$ and $b \sim N(20,1)$ to demonstrate results of first order, second order and fourth order derivative. Fig. 4 is the result of first order derivative. The step size h is 10^{-8}. From top to bottom, they are the error of $D^{(1)}_{F,1}(f,x,h)$, $D^{(1)}_{F,2}(f,x,h)$, $D^{(1)}_{C,2}(f,x,h)$ and our method $D^{(1)}_{*}(f,x,h)$, respectively. We can see that the order is the same as Fig. 1. Our method improve the first order forward method very much. We have about 3 figures significant improvement.

Fig. 5 is result of the second derivative. The step size h is 10^{-6}. The black line is $D^{(2)}_{C,2}(f,x,h)$; the blue line is $D^{(2)}_{C,4}(f,x,h)$ and the red line is our method $D^{(2)}_{*}(f,x,h)$.

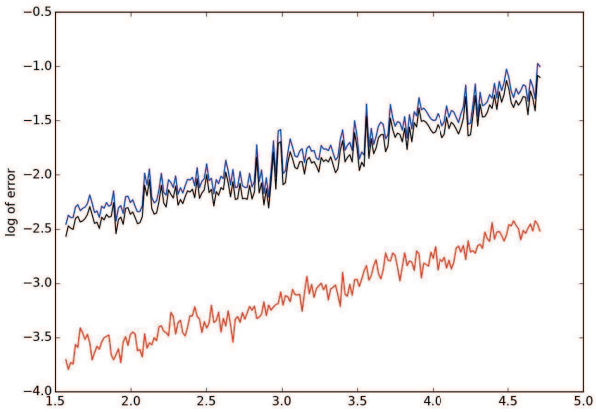

Figure 5: Second order derivative comparison, where $f(x) \approx 3e^x + \sin(20x)$. The black line is the traditional method $D^{(4)}_{C,2}(f,x,h)$, the blue line is $D^{(4)}_{C,4}(f,x,h)$, and the red line is our method.

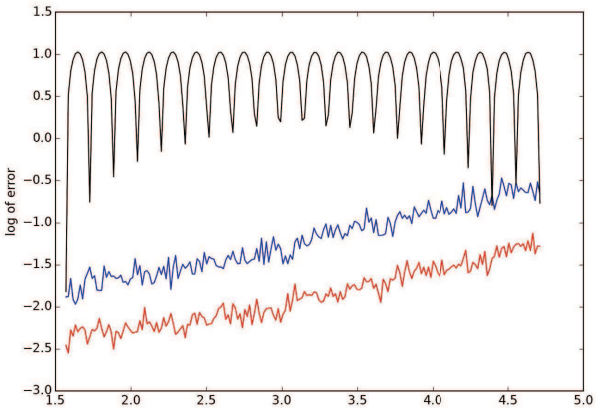

Figure 6: Fourth derivative comparison, where $f(x) \approx 3e^x + \sin(20x)$. The black line is the traditional method $D^{(4)}_{C,2}(f,x,h)$, the blue line is $D^{(4)}_{C,4}(f,x,h)$, and the red line is our method.

We can see that our method still improve the accuracy very much. There is about 2 figures significant. We also see that the high order method $D^{(2)}_{C,4}(f,x,h)$ becomes worse than $D^{(2)}_{C,2}(f,x,h)$ under the step size setting.

Fig. 6 is result of the fourth derivative. The step size h is 10^{-3}. The black line is $D^{(4)}_{C,2}(f,x,h)$; the blue line is $D^{(4)}_{C,4}(f,x,h)$ and the red line is our method $D^{(4)}_{*}(f,x,h)$. We can see that our method still works fine. There is about 3 figures significant improvement then $D^{(4)}_{C,2}(f,x,h)$ and half figure significant improvement then $D^{(4)}_{C,4}(f,x,h)$.

It is clearly that our method takes more computational complexity to obtain the estimations of truncation error and rounding error. One might ask how is the accurate performance under the equal computational complexity? If we only use two sampling of t_i

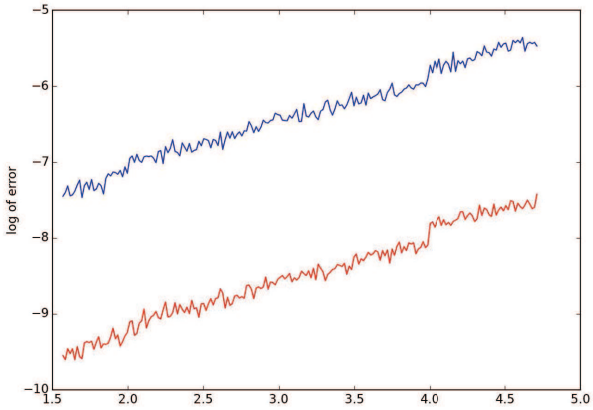

Figure 7: Comparison the numerical derivative between $D_{F,3}^{(1)}(f,x,h)$ and our method $D_{*}^{(1)}(f,x,h)$ under the same computational complexity. The blue line is $D_{F,3}^{(1)}(f,x,h)$ and the red line is $D_{*}^{(1)}(f,x,h)$.

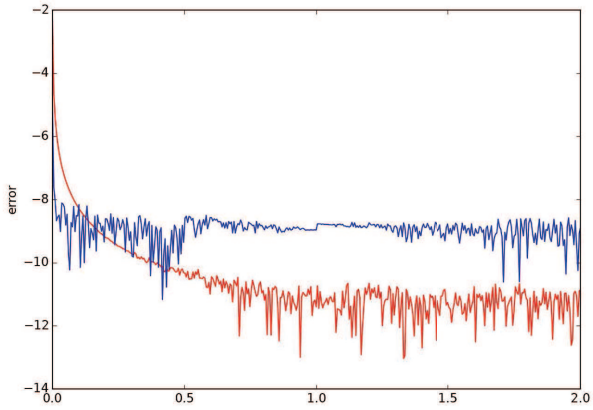

Figure 8: Using $\ln(x)$ to compare the numerical derivative between $D_{F,3}^{(1)}(f,x,h)$ and our method $D_{*}^{(1)}(f,x,h)$ under the same computational complexity. The blue line is $D_{F,3}^{(1)}(f,x,h)$ and the red line is $D_{*}^{(1)}(f,x,h)$.

to compute $D_{F,1}^{(1)}(f,x,h)$, i.e., $n=2$, the computational complexity is equivalent to the third order forward method $D_{F,3}^{(1)}(f,x,h)$ that is applied Richardson's extrapolation twice. Fig. 7 is result that compared between $D_{F,3}^{(1)}(f,x,h)$ by Richardson's extrapolation twice and $D_{*}^{(1)}(f,x,h)$ by our method. All the setting of function is the same as in Fig. 1, but change the sampling of our method from $n=8$ to $n=2$. We can see that the performance of $D_{F,3}^{(1)}(f,x,h)$ is almost the same as $D_{F,2}^{(1)}(f,x,h)$ by Richardson's extrapolation. The performance of $D_{*}^{(1)}(f,x,h)$ by our method decayed one figure significant. However, it is still better than the performance of $D_{F,3}^{(1)}(f,x,h)$.

Fig. 8 shows the result that compared between $D^{(1)}_{F,3}(f,x,h)$ by Richardson's extrapolation twice and $D^{(1)}_{*}(f,x,h)$ by our method, but change the testing function to $\ln(x)$. We are interested in the performance between the steep region and the smooth region. Thus, we focus the numerical derivative on the interval $[0.001, 2]$ and set the step size by $h = 10^{-7}$. We can see that when x is near 0, the performance of $D^{(1)}_{F,3}(f,x,h)$ is better than our method. On the other hand, when x is away from 0, our performance is better. That is our method works well in the smooth region under the fixed computational complexity. The reason is we used the fixed step size h. If the step size become small in the steep region, our method can obtain the better result.

5. Conclusion

Our linear regression method is superior to the traditional numerical derivative. We can successfully estimate truncation error and rounding error at the same time. Therefore, we can use these estimation to modified the numerical derivative to obtain a better result. The proposed genetic approach gives us a way to find the better solution of numerical derivative without the prior true solution. In the lower order numerical derivative, our performance is pretty well. The performance will decay as the order of the derivative increases. However, our method improves at least one significant figure.

Acknowledgments

This paper is supported by the Ministry of Science and Technology, R.O.C of the project MOST 103-2410-H-004-182-MY2.

References

[1] J. L. Barlow, *Numerical aspects of solving linear least squares problems*. In Rao, C.R. Computational Statistics. Handbook of Statistics. 9. North-Holland. ISBN 0-444-88096-8, 1993.
[2] R. Butt, *Introduction to Numerical Analysis Using MATLAB*, Jones & Bartlett Learning, pp. 11-18, 2009.
[3] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, McGraw-Hill, 7-th Edition, 2015.
[4] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag New York, Inc, 1999.
[5] L. F. Richardson and J. A. Gaunt, *The diferred approach to the limit*, Philosophical Transactions of the Royal Society A **226**, 299-349, 1927.
[6] C. W. Ueberhuber, *Numerical Computation 1: Methods, Software, and Analysis*, Springer, pp. 139-146, 1997.
[7] W. Y. Yang, W. Cao, T. S. Chung, and J. Morris, *Applied Numerical Methods Using MATLAB*, John Wiley & Sons, 2005.
[8] A. Curtis and J. Reid *The choice of step lengths when using differences to approximate Jacobian matrices*, J. Inst. Math. Appl., v. 13, 121-126, 1974.