# Findings gained through computational physics[a)]

Daisuke Nishihama[b)]
*Department of Physics, Saitama University, 338-8570, Japan*

(Dated: 29 June 2022)

Computers have become indispensable in physics these days. Students learn how to integrate numerically and how to solve ordinary differential equations, and the errors are discussed in terms of their principles. Numerical calculations also have their limits, and no matter how small the increments are made to increase the number of calculations in order to improve accuracy, the error cannot be reduced to zero. There is a danger of creating large errors, like a kind of butterfly effect. It has also been found that the Runge-Kutta method sometimes exhibits a behavior where the error rapidly decreases. This is a topic for future discussion.

Keywords: numeric calculation, physics, error, trapezoidal formula, Simpson formal, Euler, Runge-Kutta

## I. ALGORITHM PRINCIPLE

### A. Numerical integration method

In the world of physics and mathematics, there are many integrals that cannot be solved analytically, and the algorithms for numerically calculating them are explained below. Here, the following integral that can be derived analytically is adopted as the true value, and the error from it is examined.

$$I = \int_0^\pi \sin\theta \, \mathrm{d}\theta = 2 \tag{1}$$

#### 1. Algorithm by Trapezoidal rule

The integration interval $[0, \pi]$ is divided into $n$ equal parts, and $f(x)$ is approximated to be a straight line in each interval. Then, each parts becomes a trapezoid:

$$I_i \sim \frac{h}{2}\{f(x_{i-1}) + f(x_i)\} \tag{2}$$

$$I_{i+1} \sim \frac{h}{2}\{f(x_i) + f(x_{i+1})\} \tag{3}$$

Where $h = (b-a)/n$, $x_0 = a$, $x_i = a + ih$, $x_n = b$.

This can be introduced as:

$$I \sim \frac{h}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots 2f(x_{n-1}) + f(x_n)\right) \tag{4}$$

$$\sim h\left(\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(a+ih)\right) \tag{5}$$

---

[a)] Experimental Physics IIa
[b)] Student ID: 20RP021

## 2. Algorithm by Simpson's rule

The Simpson's rule divides the integration interval into $2n$ equal parts. This is called the Simpson 1/3 formula. As a result, $h = (b-a)/2n$, so the interval with a width of $2h$ is approximated by a quadratic function at three points.

If we donote $f(x+h) - f(x)$ by $\Delta f(x)$, then $\Delta^2 f(x)$ reads as follows:

$$\Delta(\Delta f(x)) = (f(x+2h) - f(x+h)) - (f(x+h) - f(x)) \tag{6}$$

and,

$$f(x) = \sum \frac{h}{3}(f(x_i) + 4f(x_{i+1}) + f(x_{x+2})) \tag{7}$$

$$I \sim \frac{h}{3}\left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f(a+2ih) + 4\sum_{i=1}^{n} f(a+(2i-1)h)\right) \tag{8}$$

Furthermore, the interval $[a, b]$ is divided into $3n$ equal parts and complemented by a cubic equation. This equation is called Simpson 3/8 formula, and it becomes as follows from $h = (b-a)/3n$.

$$I \sim \frac{3h}{8}(f(x_i) + 3f(x_{i+1}) + 3f(x_{i+2}) + f(x_{i+3})) \tag{9}$$

$$I \sim \frac{3h}{8}\left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f(a+3ih) + 3\sum_{i=1}^{n}(f(a+(3i-1)h) + f(a+(3i-2)h))\right) \tag{10}$$

The Simpson 3/8 formula can be numerically calculated with higher accuracy than the Simpson 1/3 formula.

## 3. Error evaluation method

Define as $f(x_i) = f_i$, $x_{i+1} = x_i + h$. Taylor expand $f_i$ near $x_i$:

$$f_{i+1} = f_i + hf_i' + \frac{h^2}{2!}f_i'' + \frac{h^3}{3!}f_i''' + \cdots \tag{11}$$

Therfore

$$f_i' = \frac{f_{i+1} - f_i}{h} - \frac{h}{2}f_i'' - \frac{h^2}{6}f_i''' - \cdots \tag{12}$$

Where integrating from $x$ to $x_{i+1}$

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\ \mathrm{d}x = hf_i + \frac{h^2}{2} + \frac{h^3}{6}f_i'' + \frac{h^4}{24}f_i'''' + \cdots \tag{13}$$

Substituting equation (12) into equation (13):

$$I_i = h\frac{f_{i+1} + f_i}{2} - \frac{h^3}{12}f_i''' \cdots \tag{14}$$

The first term of Eq(14) corresponds to the trapezoidal rule, and the error per division interval of the trapezoidal rule is proportional to $h3$. On the other hand, since the number of intervals is proportional to $1/h$, the error is proportional to $h^3 \times 1/h = h^2$, that is, proportional to $h^2$.

Similarly, the error of the Simpson 1/3 formula is

$$I_i = \frac{h}{3}(f_i + 4f_{i+1} + f_{i+2}) - \frac{h^5}{90}f'''_{i+1}\cdots \tag{15}$$

when calculated by Taylor expansion for $f_i$ and $f_{i+2}$. The error per interval is proportional to $h^5$, and the error over all intervals is proportional to $h^4$. However, the error is zero when the formula is cubic or less.

### B. Debye Model

The Einstein model of lattice specific heat explains the transition of the specific heat of real solids from high to low temperatures quite well.

However, experimentally, it is well known that the behavior of the specific heat of solids at very low temperatures follows Debye's cubic law $C(T) \propto T^3$, which is at odds with the Einstein model. Debye assumed that the forces acting on actual atoms are coupled vibrations as a whole, and he considered a model in which atoms are connected by springs like a chain, and gave them energy for lattice vibrations.

The quantized lattice vibration is called a phonon. Denoting the density of states $D(E)$ of the real phonon, if we let $\omega D$ be the upper limit of the frequency of the lattice vibration, and furthermore let the total number of states coincide with $3N$ degrees of freedom, we obtain the following Debye approximation.

$$D(E) = \begin{cases} \dfrac{9N}{\hbar\omega_D^3}E^2, & E \le \hbar\omega_D \\ 0, & E > \hbar\omega_D \end{cases} \tag{16}$$

Substituting this into the formula for specific heat:

$$C_v = \frac{9Nk_B}{(\beta\hbar\omega_D)^3}\int_0^{\beta\hbar\omega_D}\frac{x^4}{(\exp(x)-1)(1-\exp(-x))}\mathrm{d}x \tag{17}$$

Where $beta = 1/(k_BT)$, $C_D$ is Debye's specific heat function, and the Debye temperature is $T_D = \hbar\omega_D/k_B$, using $\beta\hbar\omega_D = T_D/T = 1/t$,

$$C_D = 3t^3\int_0^{1/t}\frac{x^4}{(\exp(x)-1)(1-\exp(-x))}\mathrm{d}x \sim \begin{cases} 1, & t \to 0 \\ \frac{4\pi^4}{5}t^3, & t \to +\infty \end{cases} \tag{18}$$

Finally, Debye's specific heat equation can be rearranged to:

$$\frac{C_v}{Nk_B} = 3 \times 3t^3\int_0^{1/t}\frac{x^4}{(\exp(x)-1)(1-\exp(-x))}\mathrm{d}x \tag{19}$$

Integrate Equation (19) and consider $t$ in the interval [0,2].

### C. Numerical solution of ordinary differential equations

Many of the laws of physics are described by differential equations, and by solving these equations, the motion of objects can be accurately determined.

By solving these equations, we can accurately determine the motion of an object. Here, when the initial conditions $x = x_0$, $y = y_0$:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y) \tag{20}$$

Let us consider solving numerically the ordinary differential equation expressed by where $x_0$, $x_1$, $x_2$, $\cdots$ $(x_1 - x_0 = x_2 - x_1 = h)$ and $y$ corresponding to $y_0$, $y_1$, $y_2$, $\cdots$ respectively, then

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y)\ \mathrm{d}x \tag{21}$$

Let us solve the following ordinary differential equations using the Euler and Runge-Kutta methods.

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 1 - y \tag{22}$$

### 1. Euler method

The Taylor expansion of equation (20) yields equation (23).

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2!}y''(x_0) + \frac{h^3}{3!}y'''(x_0) + \cdots \tag{23}$$

where $y(x_0) = y_0$ and $y'(x_0) = f(x_0, y_0)$. Therefore, if $h$ is sufficiently small, from equation (23), $y(x_1)$ can be approximated by the following equation.

$$y_1 = y_0 + hf(x_0, y_0) \tag{24}$$

Since this can be repeated, it can be calculated as follows.

$$y_{i+1} = y_i + hf(x_i, y_i) \tag{25}$$

This $h$ is called the step size, and in the Euler method, it is approximated to the term of $h^1$.

### 2. Runge-Kutta method

The most used 4th-order Runge-Kutta method is as follows:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{26}$$

Where, $k_1$, $k_2$, $k_3$, and $k_4$ are defined as follows:

$$k_1 = f(x, y) \tag{27}$$

$$k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \tag{28}$$

$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \tag{29}$$

$$k_4 = f(x_i + h, y_i + hk_3) \tag{30}$$

Taylor expansion of $k_1$, $k_2$, $k_3$, and $k_4$, respectively, and coefficients 1/6 in Equation (26) to match up to the term in $h^4$. The coefficient of 2/6, 2/6, and 1/6 were obtained.

### D. Rutherford scattering

As an example of ordinary differential equations, consider Rutherford scattering: we calculate the scattering by the $^{197}$Au (Au: atomic number 79) nucleus of the alpha ray (energy 8.954 MeV) emitted from $^{212}$Po, which Geiger and Marsden actually tested. The coordinates of the Au nucleus are the origin $(x, y) = (0, 0)$, and the Coulomb force $F$ acting on the alpha particle is expressed in polar coordinates $(r, \theta)$ as:

$$F = \frac{1}{4\pi\varepsilon_0}\frac{Z_\alpha A_{\mathrm{Au}} e^2}{r^2} \tag{31}$$

The Cartesian coordinate component of $F$ is $r = \sqrt{x^2 + y^2}$ as:

$$F_x = F\cos\theta = F\frac{x}{r} \tag{32}$$

$$F_y = F\sin\theta = F\frac{y}{r} \tag{33}$$

Newton's equations describing the motion of the alpha particle are rewritten from Equation (34) to the simultaneous differential equations in Equation (37).

$$\frac{\mathrm{d}v_x}{\mathrm{d}t} = \frac{Z_\alpha Z_{\mathrm{Au}} e^2}{4\pi\varepsilon_0 m_\alpha}\frac{x}{r^3} \tag{34}$$

$$\frac{\mathrm{d}v_y}{\mathrm{d}t} = \frac{Z_\alpha Z_{\mathrm{Au}} e^2}{4\pi\varepsilon_0 m_\alpha}\frac{y}{r^3} \tag{35}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v_x \tag{36}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = v_y \tag{37}$$

Where, the initial condition is:

$$v_x(t=0) = \sqrt{\frac{2E_\alpha}{m_\alpha}}, \qquad v_y(t=0) = 0 \tag{38}$$

In addition to that, the following constant values are used in the calculation. $Z_\alpha = 2$, $Z_{\mathrm{Au}} = 79$, $m_\alpha c^2 = 375.7$ MeV, $\frac{e^2}{4\pi\varepsilon_0\hbar c} = \frac{1}{137.036}$ (Fine Structure Constant), $\hbar c = 197.327$ (Mev·fm)

### E. Machine Epsilon

Floating point is data of finite precision for the approximate representation of real numbers in a computer. Computers use the binary system, where one bit can be either a 0 or a 1. Floating point is typically 32-bit or 64-bit; the 32-bit case is called a float. The first bit is called the sign, the next 8 bits are called the exponent part, and the remaining 23 bits are called the mantissa part. It means that:

$$\text{(sign)1.(mantissa portion)} \times 2^{\text{(exponent portion) - 127}} \tag{39}$$

In the case of 64 bits (the type is called double in this case), the exponential part is 11 bits, and the mantissa part is the remaining 52 digits.

In the process of arithmetic with floating-point numbers, errors occur in order to fit them into their significant digits. Thus, a number that sums to a power of 2, such as $2^{-1} + 2^{-2} + 2^{-3} = 0.875$. But other numbers (e.g. 0.6) will contain rounding errors. We now seek a machine epsilon.

Figure 1 shows a flowchart of the calculation algorithm.

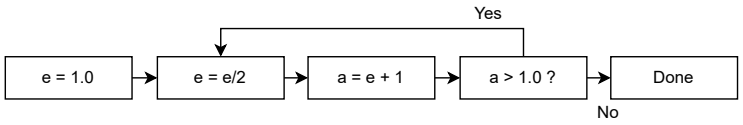


FIG. 1. Flowchart of algorithm to derive machine epsilon

## II. DEVELOPMENT NUMERICAL CALCULATION PROGRAM ENVIRONMENT

Multiple PCs were used for this experiment. Since the programs are compatible with each other, no special preparations were made for each system. The following is an overview of the main PC used.

```
Model Name: MacBook Pro
Model Identifier: MacBookPro12,1
Processor Name: Dual-Core Intel Core i5
Processor Speed: 2.9 GHz
Number of Processors: 1
Total Number of Cores: 2
L2 Cache (per Core): 256 KB
L3 Cache: 3 MB
Hyper-Threading Technology: Enabled
Memory: 16 GB
System Firmware Version: 428.60.3.0.0
SMC Version (system): 2.28f7
```

Where, the version of Python used is `3.9.2`, and the version of the C compiler (GNU Compiler Collection) is `11.2.0`.

Figure 2 shows the development environment up to the point where the graph is drawn.

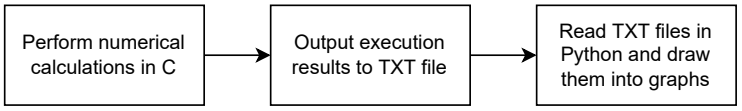


FIG. 2. The process of drawing a graph

## III. CALCULATION RESULTS

### A. Numerical integration method

The results of the calculation according to Principle section I A are shown in Figure 3.

The trapezoidal formula is almost linear on both logarithmic graphs, as the error decreases as Step Width $h$ is decreased. On the other hand, the error in the Simpson formula does not become small from around $10^{-3}$.

The results of the Debye model calculations are shown in Figure 4.

### B. Numerical solution of ordinary differential equations

Ordinary differential equations were computed according to Principle Section I C. The results are shown in Figure 5. The source code in the textbook reduces the range of $x$ that

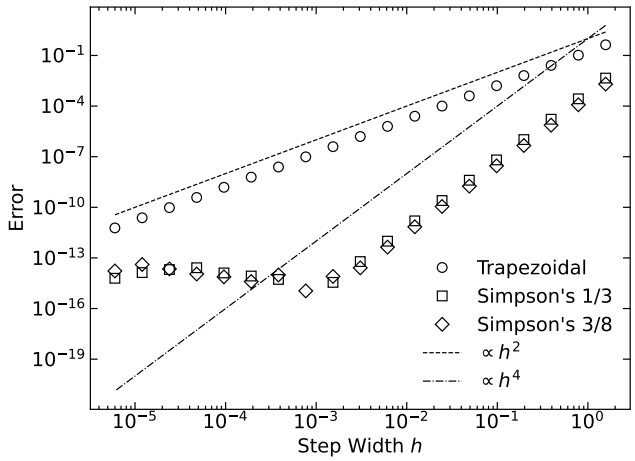


FIG. 3. Errors of the trapezoidal and Simpson rule are compared. The dashed line shows the line proportional to $h^2$, and the dashdot line shows the line proportional to $h^4$.

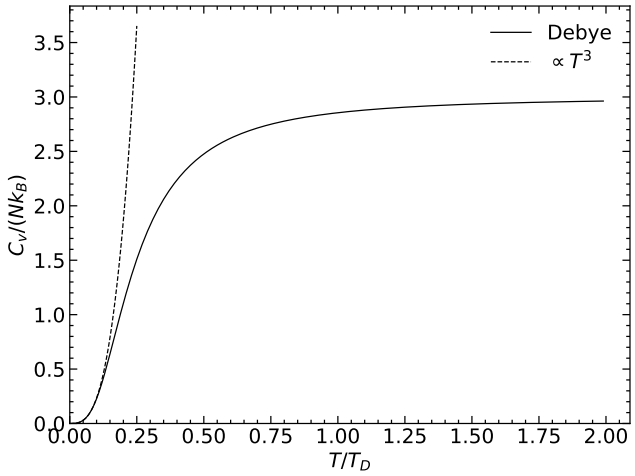


FIG. 4. Relationship between $t$ and the value of the left-hand side of Equation (19). The dashed line is proportional to the cube of temperature.

can be computed when the tick size $h$ is reduced. Therefore, we modified Source Code 1 so that the computable $x$ can be fixed and contrasted.

A similar numerical calculation performed with the Runge-Kutta method is shown in Figure 6.

In the Runge-Kutta method, while the error decreases as $h$ is increased, there are several places where the error is extremely small instantaneously when $h = 10000$, unlike when $h$ is other times. Also, the results of the calculation for Rutherford scattering using the Runge-Kutta method are shown in Figure 7.

### C. Machine Epsilon

The result of the calculation according to Principle Section I E was `2.22045e-16`.

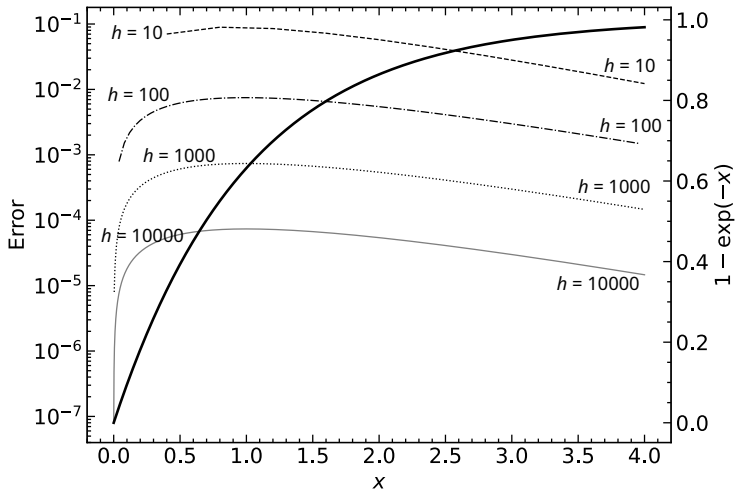


FIG. 5. The error is plotted on the left vertical axis, and the analytical value, $1 - \exp(-x)$, is plotted on the right vertical axis with a dark black line. This makes it easy to see where the error corresponds to on the curve. The increments $h$ were 10, 100, 1000, and 10000. They are represented by dashed and dotted gray lines.

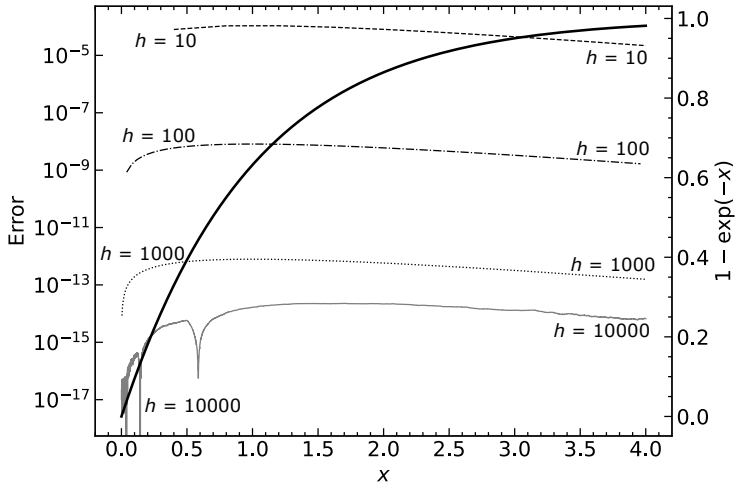


FIG. 6. The error is plotted on the left vertical axis, and the analytical value, $1 - \exp(-x)$, is plotted on the right vertical axis with a dark black line. This makes it easy to see where the error corresponds to on the curve. The increments $h$ were 10, 100, 1000, and 10000. They are represented by dashed and dotted gray lines.

## IV. CONSIDERATION

### A. Numerical integration method

The fact that the error of the Simpson formula is no longer smaller than that of the trapezoidal formula when Step Width $h$ is smaller than $10^{-3}$ is, in our opinion, a matter of principle. Simpson's formula is more accurate than the trapezoidal formula by approximating with higher order functions (quadratic and cubic functions), and when Step Width $h$ is made smaller, it no longer makes sense to approximate with higher order functions such as quadratic and cubic functions. The curve should now appear as a straight line. This sug-

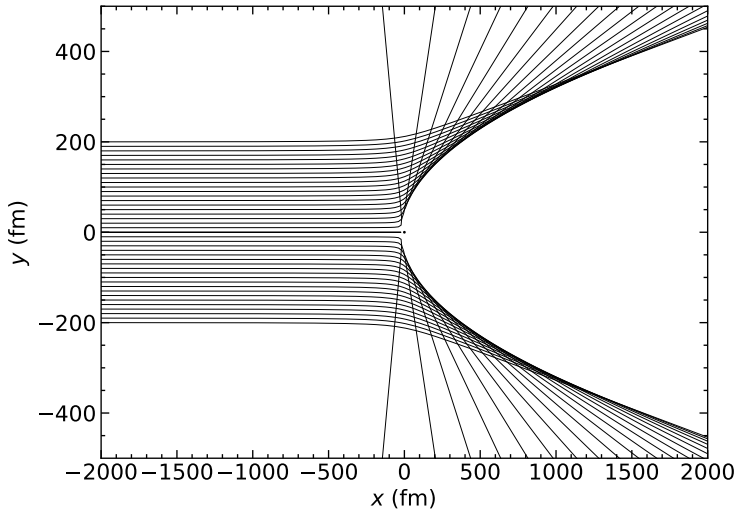


FIG. 7. Forty-one calculations along the lines of equations (34) through (37) with collision coefficients from −200 to 200 in increments of 10 are shown in the figure. A point is dotted at $(x, y) = (0, 0)$.

gests that the error of Simpson's formula is no longer smaller than that of the trapezoidal formula when Step Width $h$ is smaller than $10^{-3}$.

### B. Numerical solution of ordinary differential equations

The situation in which the Runge-Kutta method can easily predict the next point is when it is linear, in accordance with Principle Section I C. To verify this, we differentiated $1-\exp(-x)$ and plotted the slope. A linear situation means that the change in slope should be small. However, it does not correspond to that.

Therefore, it is thought to be due to something other than the principle of the Runge-Kutta method.

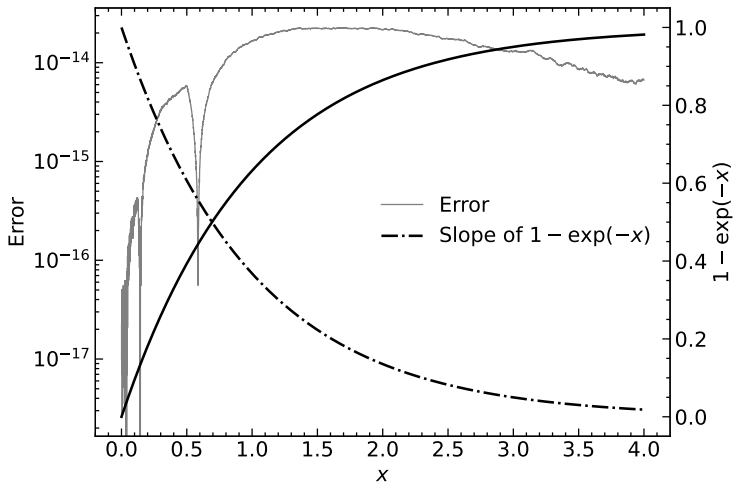


FIG. 8. From Figure 6, only $h = 10000$ was extracted, and the slope of $1 - \exp(-x)$ was added.

## V. CONCLUSION

It was shown that even the Simpson formula has an error limit even when the increment h is reduced. It should be well known that there is a danger that this can sometimes produce large errors, such as the butterfly effect, because minute errors will remain. It would also be good to investigate the cause of the sudden decrease in error in the Runge-Kutta method in the future.

## ACKNOWLEDGMENTS

I would like to express my gratitude to my friends for their advice and encouragement on various occasions.

### Appendix A: Source code used

Source Code 1. Modified source code from the Eulerian method section of the textbook

```
1   #include <stdio.h>
2   #include <math.h>
3
4   double calc_next_step(double,double,double(*)(double,double),
        double);
5
6   double f(double x, double y){
7       return (1.0-y);
8   }
9
10  int cac(int);
11
12  int main(void){
13      for(int j=1;j<10;j++){
14          int k = pow(10,j);
15          cac(k);
16      }
17      printf("Done!");
18      return 0;
19  }
20
21  double calc_next_step(double x, double y, double (*dydx)(double,
        double), double h){
22      return (y + h * dydx(x, y));
23  }
24
25  int cac(int N){
26      double x = 0.0, y = 0.0;
27      double xmax = 4.0;
28      double h = xmax/N;
29
30      FILE *fp;
31      char file[256];
32      sprintf(file, "eulerdata_%0.2d.txt", N);
33      fp = fopen(file, "w");
34
35      for(x = 0; x < xmax; x = x + h){
36          fprintf(fp,"%8.4f%14.5e%14.5e%14.5e\n",x,y,1.0-exp(-x),
                fabs(y - (1.0 - exp(-x))));
37          y = calc_next_step(x, y, f, h);
```

```
38          }
39
40          fclose(fp);
41
42          return 0;
43      }
```

Source Code 2. Derive machine Epsilon

```
1       #include <stdio.h>
2       #include <math.h>
3
4       int main(void){
5           double a, _e;
6           double e = 1.0;
7           do{
8               _e = e;
9               e = e/2.0;
10              a = 1.0+e;
11          }while(a > 1.0);
12          printf("%.5e\n",_e);
13      }
```