Numerical Errors in Computation

Oves Badami

May 2023

1 Numerical Methods

The goal of the numerical methods is to get to an approximate solution and not the exact solution. The difference between the exact and approximate solution is very creatively called as the error. How much error can we tolerate would depend on the application at hand for example doing the simulation at room temperature the update in the electrostatic potential from the Poisson solver should be lesser than the thermal voltage, $\frac{k_B T}{q}$. The thermal voltage at room temperature is 26 mV so generally the error that is tolerated is about 3 mV (personal choice). Why it should be lesser than the thermal voltage ??

2 Errors

Now that we know what is a meant by error, $E_T = V_T - V_A$, i.e. difference between the true value (V_T) and approximate value (V_A). Many times dependSo whaing on the true value the magnitude of E_T would also vary. For example if the true value is small then in most of the cases E_T would also be small. Hence we may incorrectly think that the error is small. So a better measure for the error is get the fractional relative error

$$E_{TF} = \frac{E_T}{V_T} \tag{1}$$

Multiplying E_R would give you the error in percentage terms. However in most of the cases we don't know the true value. If we knew the true value why would we need to calculate the approximate value. In this case we need to define approximate error

$$E_{AF} = \frac{V^n - V^{n+1}}{V^{n+1}} \tag{2}$$

where V^i is the value after the i^{th} iteration. Recall what is meant by the word iteration from initial lectures. And the simulations are continued until the error is below the limit specified. Something like 0.1 times the thermal voltage for the Poisson solver.

3 Roundoff Error

Since the computers cannot represent numbers with infinite precision, hence we have to consider only the finite number of digits. This is true for not only the irrational numbers but even for the rational numbers. The error that results from finite precision of the computers is called as the roundoff error. Do you think that these errors are negligible and shouldn't be a cause of problem ?

Now the point is why do we have to roundoff any number. This has got to do with the way the numbers are stored in the computers (something that you have already learnt in the Introduction to ICDT course). This also results in our ability to represent only some of the numbers and not all of them within the range. Interestingly the gap between the numbers increases as the magnitude of the number increases. To see this we will work out an example in the class.

3.1 Floating Point representation

Before we workout an example on this let us first discuss how the numbers are stored in the computers. What is meant by floating point representation. Any numbers can in general be represented as mb^e where m is mantissa, b is the base of the number and e is the exponent. Can you see why this number format is called as the floating point representation ? **The decimal point is allowed to float unlike the case of the fixed point representation**. For example 0.029411765 is represented as 0.2941×10^{-1} (in the base 10).**Why not as 0.0294$\times 10^0$? Number of significant digits and what if the number gets smaller further** This necessitates that m as

$$\frac{1}{b} \leq m < 1 \tag{3}$$

This means that for a base 10 number m would lie between 0.1 and 1, for a base 2 m would lie between 0.5 and 1. In general the number would be stored as $s_n s_e m_1 m_2 m_3 \cdots e_1 e_2 e_3 \cdots$. where s_n is the sign bit of the number and s_e is the sign bit of the exponent (1 for negative and 0 for positive)

To illustrate these concepts let us develop a floating point numbers which has 2 bits for exponent and 3 bits for mantissa. So the number would be like $s_n s_e e_1 e_0 m_2 m_1 m_0$. What will be the smallest positive number We will have $s_n = 0$, $s_e = 1$, $e_1 e_0 = 11$, what about mantissa ? The smallest value of the mantissa would be $m_2 m_1 m_0 = 100$ why not 001, 010 etc. Because the smallest value of the mantissa can be 0.5. So the smallest number would be
0111100 = 0.0625
0111101 = 0.078125
0111110 = 0.093750
0111111 = 0.109375
Next set would be the when $e_1 e_0 = 10$
0110100 = 0.125
0110101 = 0.156250
0110110 = 0.187500
0110111 = 0.218750
The Δ (the difference between the two successive numbers) in the first case is 0.015625 while in the second case is 0.03125. This is due to the increase in the exponent case.

This exercise clearly shows that we are limited to represent the smallest number. If the number that we have to represent becomes smaller than the smallest number that can be represent an underflow is said to have occurred. Using the exact the same logic we are also limited to the largest number that can be represented (many of you all must have faced this issue in the ITP). When this occurs an overflow is said to have taken place. It is necessary that we choose range of the numbers that we will work appropriately. So why dont we directly use largest possible representation ... (1) the would take larger memory. This may not be such a concern for many programs but can get critical for real life examples ... Apollo space craft[RAM 4kB and hard drive 72 kB](2) The speed of the execution would also slow down.

It is also obvious that we cannot represent irrational numbers (because they have infinite number of significant digits). But even for the rational numbers we cannot get one to one match and hence they cannot be represented precisely. This error is also called as quantization error. The way these numbers are stored results in the fact that as the numbers get larger and larger. Thus the quantization error increases as the magnitude of the number increases. Direct consequence of the increase in the Δ.

However, with the availability of the modern computers (good RAM and storage capacities) we can used extended precision to overcome this issue. However, there still remains the problem for low temperatures where the Fermi-Dirac function, and other exponential tend to blow up. So we need to go for even higher precision or some other methodologies (which are beyond the scope of this course). These issues are particularly problematic when we want to evaluate exponential.

In summary: The underlying cause of the round off error is that the mantissa can hold only finite number of bits (significant digits). Even with the range that is allowed we can represent finite numbers only (quantization) and this would increase with the increase in the magnitude of number.

Okay so we have discussed the underlying cause of the error. But how do the computers do retain the number of significant digits. By rounding off or chopping of. The error that results from rounding off is lesser than the chopping off but sometimes chopping is done as it is simple and faster.

This the round off error.

3.2 Typical situations when these errors arise ?

3.2.1 Subtractive Cancellation

Let us say that we have to subtract two numbers that are almost of the same magnitude. because of the finite number of mantissa the round of error might lead to round off error. For example consider that you have to calculate the root of a quadratic equation $ax^2 + bx + c = 0$ THe roots of these equations are

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4}$$

But if the coefficients are such that $b^2 >> 4ac$ then we will have the case of subtracting the numbers that are almost equal. In the worst case this difference might become 0. A possible and elegant solution in this case is that we can rewrite the roots

$$x_{1,2} = \left(\frac{b \pm \sqrt{b^2 - 4ac}}{-2c} \right)^{-1} \tag{5}$$

However to be 100% sure that we have covered all grounds it is imperative that we employ extended precision.

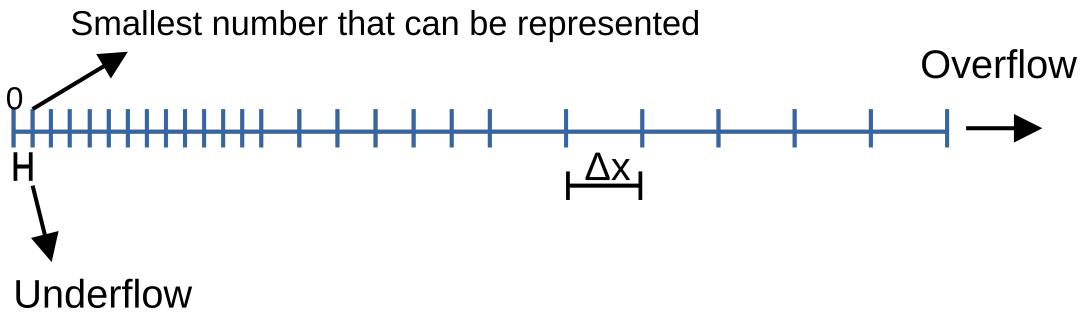

Figure 1: Illustration of the concepts for the round off error

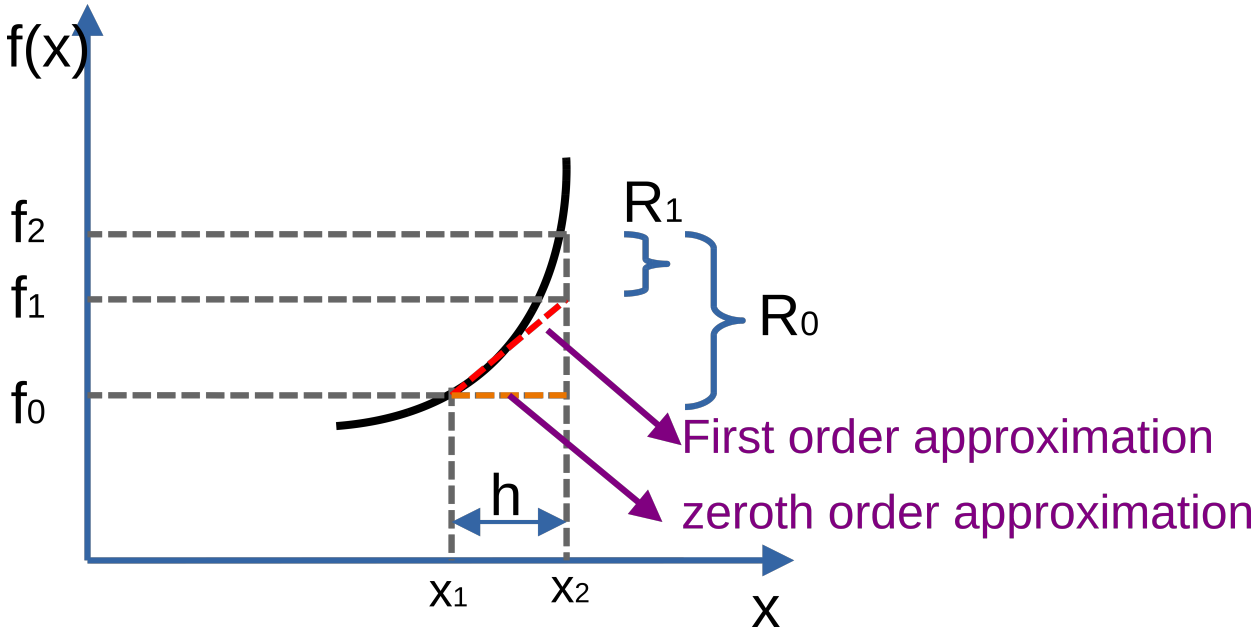

Figure 2: Schematic illustrating the impact of inclusion of the higher order terms

3.2.2 Series calculations

These errors are so small why should I worry about them. In most of the practical cases many many computations are interdependent. So these errors would accumulate over these calculations and at the end because of these errors the result may or may not make sense. A practical application where these make their mark is the application of Monte Carlo technique for computing the current in modern day devices.

4 Truncation Error

Truncation error as the name itself suggest is the error that is a result of truncating an exact expression so as to consider only the "dominant terms". Consider a Taylor's series expansion of a function, $f(x)$, around a point x_{i+1}

$$f(x_{i+1}) = f(x_i) + hf(x_i) + f'(x_i)\frac{h^2}{2!} + f''(x_i)\frac{h^3}{3!} + \cdots \tag{6}$$

where $h = x_{i+1} - x_i$. Thus the approximate values of $f(x_i)$ can be written as

$$f(x_{i+1}) \approx f(x_i) \tag{7}$$
$$f(x_{i+1}) \approx f(x_i) + hf'(x_i) \tag{8}$$
$$f(x_{i+1}) \approx f(x_i) + hf'(x_i) + \frac{h^2}{2!}f''(x_i) \tag{9}$$

Similarly you can add higher and higher order terms to get a better approximate results (**Physical/Graphical interpretation**). An important point to note here is that if the h is sufficiently small then the higher order terms can be neglected. **Physical/Graphical interpretation**

A completely valid question is why should we care about the truncation error in this course. Since this is a numerical method course we will often rely on numerical derivative/integration calculations. In such cases, if we want to be reasonably accurate about our calculations, we have to ensure that $h = x_2 - x_1$ is such that

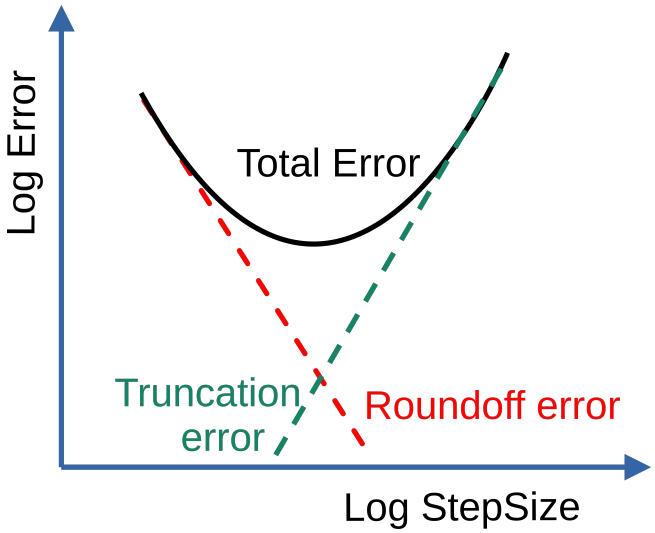

Figure 3: Schematic illustrating total error

these calculations are correct. For example we will often write the numerical derivative as

$$f'(x) = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{10}$$

This itself is an approximation as you can see from the Taylor series by truncating it only to linear terms. More about this when we will discuss the Numerical differentiation

So how do we overcome the truncation error... the answer is that we need to improve upon the existing formulations. Like in the case of the derivative we need to use higher order terms for this purpose. However, it may not always be the case.

5 Total Error

The total numerical error in a computation is sum of the round off error and truncation error As we have discussed the truncation error is due to chopping off the higher order terms. As it can be seen below

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2!}f''(x_i) + \cdots \tag{11}$$

if we reduce h the truncation error reduces. That is if we are interested in only the zeroth order approximation $f(x_{i+1}) \approx f(x_i)$ and if we reduce the h by half then the error would also reduce by half and if we consider the linear approximation then reduction in the h by half would reduce the error by a quarter. So the nice idea would be to keep on reducing h. The answer is ofcourse NOT.

If we keep on reducing the h the the roundoff error increases which again increases the error. Hence while doing simulations it is imperative that we are cognizant of this particular fact.

Ideally we would like to have a large step size so that we can reduce the computational burden but as seen the roundoff errors increases.

Fig 4.2 of Chapra

So the point is how do we go about it. The general rule of thumb is that we start from the coarse value of h and the keep on reducing h until the answer more or less saturates. Generally the round off errors for most of the cases occur at sufficiently small values of h which are not be used in most of the practical cases of semiconductors. They would become significant in doing simulations at low temperatures.

Consider a case of differentiation

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - Kh^2 \tag{12}$$

where Kh^2 is the truncation error.

Let $\overline{f}(x)$ be the rounded form of the $f(x) = \overline{f}(x) + e_x$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + \frac{e_{i+1} - e_{i-1}}{2h} - Kh^2 \tag{13}$$

So the total error is

$$TE = \frac{e_{i+1} - e_{i-1}}{2h} - Kh^2 \tag{14}$$

6 Error Propagation

Consider we want to evaluate a function, $f(x)$. Let us say that we don't know the exact value of x and let us say its approximate value is $\overline{x}$. Now because of the approximate value of x we will also get the approximate (incorrect value of x). This is critical because we have to do a self-consistency between the Poisson and the carrier concentration then an approximate solution might lead to divergence. How do we mitigate it ? For example consider we want to calculate the value the drift current in the depletion region of a diode

$$J_n = qn\mu_n E = qn(x)\mu_n(E)E(x) \tag{15}$$

If we there is an error in the calculation of x can be let us due to round-off error. This would mean that the n would be incorrectly estimated because in general carrier concentration is position dependent, naturally the electric field would also be different thus now the J_n that is estimated is also incorrect. If we do further analysis using this J_n then subsequently those calculations would also have error.

On the positive side, remember that we are doing numerical calculation which as we have discussed deals only with the approximate solutions.