CHAPTER 3

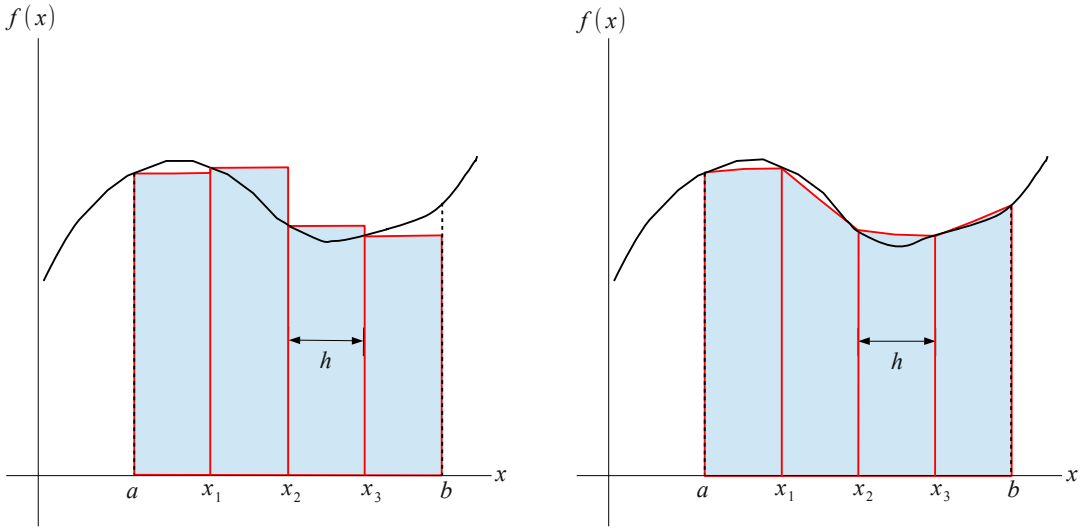
NUMERICAL INTEGRATION

Like derivatives, integrals are also common mathematical tools used in physics. Derivatives involve subtraction of similar values and suffer from bit-off errors. Integration is essentially addition and thus numerical integration is a bit more robust than numerical derivative. However, lots of addition incur significant round-off errors. A good algorithm uses less addition without the expense of accuracy. First, a simple method (rectangle rule) is used to demonstrate a basic idea of numerical integration. Like forward finite difference method of numerical derivative, this method is not accurate enough for practical use. However, a simple modification (trapezoidal rule), similar to mean finite difference method of numerical derivative, improves the accuracy. More advanced methods (Simpson rule) will be introduced.

Improper integral needs special attention. For example, when integral limits involve infinity like $\int_0^\infty f(x)\,\mathrm{d}x$, most numerical integration methods require infinitely many addition, which cannot be done. Another example is the case where integrand has integrable singularities like $\frac{1}{\sqrt{x}}$. Since we cannot evaluate the function value at $x = 0$, the common numerical methods fail. There is no systematic resolution to these problems. We need to deal with them on case-by-case basis. Several common practices will be introduced in this chapter. In addition, a magic method called Gaussian quadrature, which gives rather accurate results for certain types of integrals just by computing several points, will be discussed.

3.1 Rectangular rule

We want to integrate a function $f(x)$ from a to b. Similarly to the numerical derivative problems, there are two different cases. In one case, a closed form expression of the function is known and we can evaluate the



(a) Forward rectangular method. The large errors are clearly visible.

(b) Trapezoidal method. The errors are much smaller than those in the rectangular method.

Figure 3.1: Illustration of simple numerical integration methods

function value at any point of $x \in [a, b]$. In the other case, the function values are given as a finite sequence $f_n = f(x_n), n = 0, \cdots, N$ and the analytical expression of the function is unknown. In this section, we focus on the former case and the latter case will be discussed in a later chapter.

We begin with the Rieman's definition of integral:

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{N\to\infty} \sum_{n=0}^{N-1} f(x_n)\,h = \lim_{N\to\infty} \sum_{n=1}^{N} f(x_n)\,h \tag{3.1}$$

where $h = (b-a)/N$ and $x_n = a + n\,h$. Note that h depends on N. Numerical methods do not understand this kind of limit since it ends up with $\infty \times 0$. Beside, summing infinitely many terms costs infinite CPU time. We hope that sufficiently large N (i.e., sufficiently small $h > 0$) gives a value close to the exact integral. This is the rectangular rule:

$$\int_a^b f(x)\,\mathrm{d}x \approx \hat{I}_\mathrm{F} f(x) \equiv \sum_{n=0}^{N-1} f(x_n)\,h \tag{3.2a}$$

$$\int_a^b f(x)\,\mathrm{d}x \approx \hat{I}_\mathrm{B} f(x) \equiv \sum_{n=1}^{N} f(x_n)\,h \tag{3.2b}$$

where $\hat{I}_\mathbf{F}$ and $\hat{I}_\mathbf{B}$ are forward and backward rectangular rule operator, respectively.

Algorithm 3.1 Integration by the forward rectangular rule

The following steps evaluate Eq. 3.2a.

1. Set the step length: $h = \dfrac{b-a}{N}$
2. Set $s = 0.0$ where s should be double (float64).
3. Repeat steps 4-6 for $n = 0$ to $n = N - 1$:
4. $x = a + n * h$.
5. $s = s + f(x)$. [where $f(x)$ can be inline function or funtion subprogram.]
6. Go back to step 4 and repeat with new n.
7. The integral is given by $s * h$.

Steps 3-6 can be simplified using `linspace` and `sum` functions. See sample codes.

The forward rectangular rule is illustrated in Fig 3.1a. The integral [the area below the curve $f(x)$] is approximated by the sum of many small rectangles. However, the large errors are clearly seen in the illustration where the slope of curve is steep.

To investigate the error in the rectangular rule, we consider the small integral interval from x_n to $x_{n+1} = x_n + h$. Expanding the integral with respect to h (See Appendix 3.I), the integral is expressed as power series of h:

$$\int_{x_n}^{x_n+h} f(x)\,\mathrm{d}x = f(x_n)h + f'(x_n)\frac{h^2}{2} + f''(x_n)\frac{h^3}{3!} + +f^{(3)}(x_n)\frac{h^4}{4!} + \mathcal{O}\big(h^5\big). \tag{3.3}$$

Then, the whole integral in the forward scheme is expressed as

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{n=0}^{N-1}\int_{x_n}^{x_n+h} f(x)\,\mathrm{d}x = \sum_{n=0}^{N-1}\left[f(x_n)h + f'(x_n)\frac{h^2}{2} + \mathcal{O}\big(h^3\big)\right] \tag{3.4}$$

By neglecting h^2 and higher orders, we obtain the rectangular rule. Therefore, the error of the rectangular rule is the order of h^2 per segment. Since there are N segments, the total error is order of $h^2N = (b-a)h$. Hence, the total error is the order of h. You might think that if a very small value of h is used the error is negligible. Unfortunately, the round-off error gets too large when N is too large (See Example 1.8). In practice, this method is rarely used.

3.2 Trapezoidal rule

It is better to approximate the area using trapezoids as shown in Fig 3.1b.

$$\int_a^b f(x)\,\mathrm{d}x \approx \hat{I}_T f(x) \equiv \sum_{n=0}^{N-1}\frac{f(x_{n+1}) + f(x_n)}{2}\,h \tag{3.5}$$

$$= \boxed{\frac{h}{2}\left[f(a) + f(b)\right] + \sum_{n=1}^{N-1} f(x_n)h} \tag{3.6}$$

The trapezoidal rule is equivalent to the mean of the forward and backward rectangular rules, i.e., $\hat{I}_{\mathrm{T}} f(x) = \frac{1}{2}\left[\hat{I}_{\mathrm{F}} + \hat{I}_{\mathrm{B}}\right] f(x)$. Note also that the difference between the trapezoidal rule and the rectangular rule is only how the end points $f(a)$ and $f(b)$ are treated:

$$\hat{I}_{\mathrm{T}} f(x) = \hat{I}_{\mathrm{F}} f(x) + \frac{h}{2}\left[f(a) - f(b)\right] = \hat{I}_{\mathrm{B}} f(x) - \frac{h}{2}\left[f(a) - f(b)\right] \tag{3.7}$$

Nevertheless, this simple modification improve the accuracy significantly.

Let us find the order of error by substituting the forward finite difference method, $f'(x_n) = \dfrac{f(x_n + h) - f(x_n)}{h} + \mathcal{O}(h)$ into Eq. (3.3):

$$\int_{x_n}^{x_{n+h}} f(x)\,\mathrm{d}x = f(x_n)h + f'(x_n)\frac{h^2}{2} + \mathcal{O}\big(h^3\big) = \frac{f(x_n) + f(x_{n+1})}{2}h + \mathcal{O}\big(h^3\big). \tag{3.8}$$

If h^3 and higher orders is ignored, we obtain the trapezoid rule. Hence, the trapezoidal rule is locally accurate up to h^2, better than the rectangular rule. The total error is the order of $h^3 N = (b-a)h^2$. The trapezoidal method is commonly used due to its simplicity and reasonable accuracy. Interestingly, if the function vanishes at the integral limits, $f(a) = f(b) = 0$, then the rectangular rule produces exactly the same result as the trapezoidal rule.

Algorithm 3.2 Integration by the trapezoidal rule

The following steps evaluate Eq. 3.5.

1. Set the step length: $h = \dfrac{b-a}{N}$
2. Set $s = 0.5 * (f(a) + f(b))$ where s should be double (float64).
3. Repeat steps 4-6 for $n = 1$ to $n = N - 1$:
4. $x = a + n * h$.
5. $s = s + f(x)$. [where $f(x)$ can be inline function or funtion subprogram.]
6. Go back to step 4 and repeat with new n.
7. The integral is given by $s * h$.

Steps 3-6 can be simplified using `linspace` and `sum` functions. See sample codes.

3.3 Simpson method

There is an even better method. In Eq. (3.3) the rectangular method ignored $f'(x)$ and all higher order derivatives. That means the function $f(x)$ is approximated by piece-wise constant functions (no slope). Note that you need only one function value $f(x_n)$ to calculate the area of the rectangle. To increase the accuracy, the slope of the function, $f'(x)$ in Eq. (3.3), is taken into account in the trapezoidal method. That means two function values $f(x_n)$ and $f(x_{n-1})$ are needed to compute the area of the individual segment. Natural extension to this line of approximation is to take into account the curvature or $f''(x)$. Noting that the evaluation of $f''(x)$ requires three data points, we utilize the another expansion similar to Eq. (3.3),

$$\int_{x_n}^{x_n - h} f(x)\,\mathrm{d}x = -f(x_n)h + f'(x_n)\frac{h^2}{2} - f''(x_n)\frac{h^3}{3!} + + f^{(3)}(x_n)\frac{h^4}{4!} + \mathcal{O}\big(h^5\big) \tag{3.9}$$

Using the expansions (3.3) and (3.9), we find the integral from x_{n-1} to x_{n+1} as

$$\int_{x_n-h}^{x_n+h} f(x)\,\mathrm{d}x = 2f'(x_n)h + 2f''(x_n)\frac{h^3}{3!} + \mathcal{O}\big(h^5\big) \tag{3.10}$$

Note that the fourth order term is canceled out, which makes this approximation accurate. Substituting the finite difference formula of the second order derivative [Eq (2.14) in Chapter 2] into Eq. (3.10), we find the integral

$$\int_{x_{n-1}}^{x_{n+1}} f(x)\,\mathrm{d}x = \left[\frac{1}{3}f(x_{n-1}) + \frac{4}{3}f(x_n) + \frac{1}{3}f(x_{n+1})\right] h + \mathcal{O}\big(h^5\big), \tag{3.11}$$

which leads to local error at the order of h^5. Repeating this formular, we obtain the Simpson rule

$$\int_a^b f(x)\,\mathrm{d}x \approx \hat{I}_\mathrm{S} \equiv \boxed{\sum_{j=0}^{N/2-1} \left[f(x_{2j}) + 4f(x_{2j+1}) + f(x_{2j+2})\right] \frac{h}{3}} + \mathcal{O}\big(h^4\big) \tag{3.12}$$

The error of the Simpson rule is the order of h^5 per segment and thus h^4 for the whole integral which is two orders of magnitude better than that of the trapezoidal rule.

Algorithm 3.3 Integration by the Simpson rule

The following steps evaluate Eq. 3.12.
The number of points N should be even.

1. Set the step length: $h = \dfrac{b-a}{N}$
2. Set $s = -f(a) - f(b)$ where s should be double (float64).
3. Repeat steps 4-6 for $j = 0$ to $j = N/2 - 1$:
4. $x = a + 2 * j * h$.
5. $s = s + 2.0 * f(x) + 4.0 * f(x + h)$.
6. Go back to step 4 and repeat with new j.
7. The integral is given by $s * h/3.0$.

Steps 3-6 can be simplified using `linspace` and `sum` functions. See sample codes.

EXAMPLE 3.1 Errors in various numerical integration methods

Let's integrate $\sin(x)$ from $x = 0$ to $x = \pi/2$. The exact answer is $\cos(0) - \cos(\pi/2) = 1$. Program 3.1 computes the integral using the rectangular, trapezoidal, and Simpson rules. The error of each rule is plotted in Fig. 3.2. As h decreases, the error also decreases with all methods. The Simpson rule has small errors even at $h = 0.1$.

Exercise 3.1 Numerically integrate $f(x) = \dfrac{\sin(x)}{1+x^2}$ from $x = 0$ to $x = \pi$.

Analytical solution by Mathematica is

$$\begin{aligned}\int_0^\pi \frac{\sin(x)}{1+x^2}\,\mathrm{d}x &= \frac{e}{4}\left[-2\mathrm{Ci}(i)+\mathrm{Ci}(i-\pi)+\mathrm{Ci}(i+\pi)+2\mathrm{Shi}(1)+i\mathrm{Si}(i-\pi)+i\mathrm{Si}(i+\pi)\right] \\ &\quad +\frac{1}{4e}\left[2\mathrm{Ci}(i)-\mathrm{Ci}(i-\pi)-\mathrm{Ci}(i+\pi)+2\mathrm{Shi}(1)+i\mathrm{Si}(i-\pi)+i\mathrm{Si}(i+\pi)\right] \qquad (3.13)\end{aligned}$$

where Ci, Si, and Shi are trigonometric integral functions. The answer should be real but Eq. 3.13 contains imaginary unit. It is not obvious that the imaginary parts are canceled out. This expression is too complicated to see the answer. Analytical solution is not always useful. Furthermore, these trigonometric integrals must be numerically evaluated. So, why don't we evaluate the original integral numerically from the beginning?

3.3.1 Adaptive quadrature

As demonstrated above, the accuracy of numerical integration depends on the choice of the grid interval h. In practice, finding an appropriate value for h is tedious. Especially when the integrand $f(x)$ changes rapidly in some region and smooth in other region (*stiff* function), a small value of h is necessary only for the rapidly changing region. If a constant h is used for the whole region, we may be wasting computer time. Technically speaking, it is possible to use different h but it is quite cumbersome to do it manually. Therefore, we ask computer to find the best value of h at each point, which is known as adaptive grid method.

The basic idea is simple. First, we integrate the function using three grid points x_1 and x_2 with interval $h^{(0)}$. Here the index $^{(0)}$ indicates the depth of adaptivity. Let us call the result of the integration $I^{(0)}$. Then, we integrate the function between x_1 and x_2 again using the interval $h^{(1)} = h^{(0)}/2$ and obtain a new result $I^{(1)}$. If the difference between $I^{(0)}$ and $I^{(1)}$ is smaller than a specified tolerance, we accept $I^{(1)}$ as accurate result and move on to the next segment. If not, we reduce the interval again as $h^{(2)} = h^{(1)}/2$. We repeat this until the error becomes small enough.

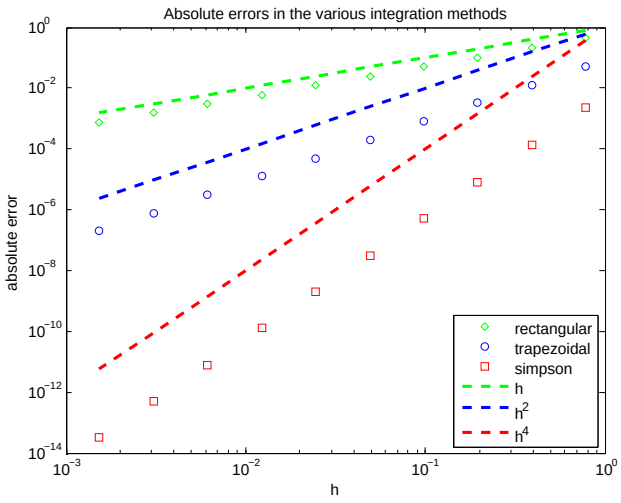


Figure 3.2: Output of Example 3.1.

EXAMPLE 3.2 Adaptive Quadrature

Let us numerically calculate the integral

$$I = \int_0^5 (4x - x^2)e^{-2x}\,\mathrm{d}x$$

whose exact value is $\frac{3}{4} + \frac{17}{4}e^{-10} \sim 0.75019294970149056062$. The integrand rises very rapidly and decreases to zero slowly. We will evaluate it using the adaptive quadrature. MATLAB has built-in function `integral(func,xmin,xmax)` which uses adaptive quadrature with a default error tolerance (Absolute error $= 10^{-10}$ and relative error $= 10^{-6}$). Instead of writing our own code, we will use it this time.

```
>> fprintf('%24.16e\n',integral(@(x) (4*x-x.^2).*exp(-2*x),0,5))
  7.5019294970149075e-01
```

3.4 Improper Integrals

When the integral limit involves infinity, the numerical methods we discussed above won't work since the number of grid points become infinity. If the integrand is singular at a point within the integral limit, again the regular method fails. We need special methods. In the following, we discuss some of simple ways to avoid such difficulty.

3.4.1 Improper Integrals: ∞ in Limits

If the upper limit is ∞ or the lower limit is $-\infty$, for example, $\displaystyle\int_0^\infty f(x)\,\mathrm{d}x$, all the methods we discussed so far cannot be used. One way to overcome this problem is to split the integral to two parts

$$\int_0^\infty f(x)\,\mathrm{d}x = \int_0^a f(x)\,\mathrm{d}x + \int_a^\infty f(x)\,\mathrm{d}x \tag{3.14}$$

where a is a positive constant. The first term in the right hand side can be integrated by the trapezoidal or simpson rule. The second term needs to be transformed to a numerically computable form by introducing a new variable $t = \dfrac{1}{x}$. Then, the integral we need to compute is

$$\int_a^\infty f(x)\,\mathrm{d}x = \int_0^{1/a} \frac{1}{t^2} f\left(\frac{1}{t}\right)\mathrm{d}t \tag{3.15}$$

The integral in the right hand side can be integrated by a standard method. However, the new form is not necessarily trouble free since the integrand is not defined at the lower bound (divided-by-zero error). If we can evaluate $\lim_{t\to 0} 1/t^2 f(1/t)$ analytically by hand, then standard methods such as the Simpson method works.

The following types of improper integral:

$$\int_0^\infty \mathrm{e}^{-x} f(x)\,\mathrm{d}x \qquad \text{Use Gauss-Laguerre quadrature.} \tag{3.16}$$

$$\int_{-\infty}^\infty \mathrm{e}^{-x^2} f(x)\,\mathrm{d}x \qquad \text{Use Gauss-Hermite quadrature.} \tag{3.17}$$

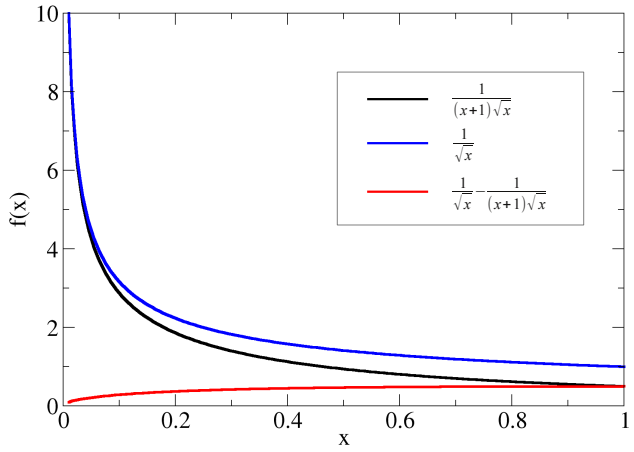


Figure 3.3: Due to the divergence at $x = 0$, it is difficult integrate the original function (black line). The blue line has the same singularity at $x = 0$ but can be analytically integrate. The difference (red line) does not have a singularity and hence common numerical integration works fine.

can be evaluated by the Gaussian quadrature which we will discuss in Section 3.5.

3.4.2 Improper Integrals II: Integrable Singularities

If the integrand has integrable singularities such as $\dfrac{1}{\sqrt{x}}$ within the integral limits, the standard methods fail. Such improper integrals are ubiquitous in physics. A common method is to isolate the singularity and integrate it analytically. Then, we integrate the remaining part by a numerical method.

EXAMPLE 3.3 Removal of Integral Singularity

Consider an improper integral

$$\int_0^a \frac{1}{(1+x)\sqrt{x}}\,\mathrm{d}x = \pi - 2\arctan\left(\frac{1}{\sqrt{a}}\right) \tag{3.18}$$

where a is a positive constant. This integral is finite despite that the integrand diverges at $x = 0$. For $x = \epsilon \ll 1$, the integrand can be expanded as

$$\frac{1}{(1+\epsilon)\sqrt{\epsilon}} \sim \frac{1-\epsilon}{\sqrt{\epsilon}} \sim \frac{1}{\sqrt{\epsilon}} \tag{3.19}$$

Hence, the singularity is $\dfrac{1}{\sqrt{\epsilon}}$. We split the integral in two parts as follows:

$$\begin{aligned}\int_0^a \frac{1}{(1+x)\sqrt{x}}\,\mathrm{d}x &= \int_0^a \frac{1}{\sqrt{x}}\,\mathrm{d}x + \int_0^a \left[\frac{1}{(1+x)\sqrt{x}} - \frac{1}{\sqrt{x}}\right]\mathrm{d}x \\ &= 2\sqrt{a} - \int_0^a \frac{\sqrt{x}}{1+x}\,\mathrm{d}x \end{aligned} \tag{3.20}$$

The last integral is not improper and thus can be integrated by a standard method. Indeed, Fig. 3.3 shows that the new integrand has no singularity at $x = 0$ and the curve is very smooth. In addition the integral of this function is small compared with the integral of the singular part. Hence, numerical error is reduced.

Program 3.2 computes integral (3.18). The proper part of the integral is done with the trapezoidal rule. The result is compared with the analytic solution. Using $a = 1$, $h = 0.01$, the program produces the output

```
Numerical = 1.571003957326e+00
    Exact = 1.570796326795e+00
```

Only the first three figures are correct, but which is acceptable in most cases.

3.5 Gaussian Quadrature

The Gaussian quadrature magically evaluates improper integrals utilizing the properties of orthogonal polynomials such as Legendre and Laguerre polynomials. Despite the integral limit is infinity, you need to evaluate the integrand only at several points. The theoretical justification of the Gaussian quadrature needs knowledge of special functions. Here we show only the formulas. See Appendix 3.II for the theory behind the Gaussian quadrature.

- Gaussian-Laguerre Quadrature

$$\int_0^\infty f(x)\mathrm{e}^{-x}\,\mathrm{d}x = \sum_{i=1}^{N} w_i f(x_i) \tag{3.21}$$

where weight w_i and abscissa x_i are given in Table 3.1.

- Gaussian-Hermite Quadrature

$$\int_{-\infty}^\infty f(x)\,\mathrm{e}^{-x^2}\,\mathrm{d}x = \sum_{i=1}^{N} w_i f(x_i) \tag{3.22}$$

where weight w_i and abscissa x_i are given in Table 3.2.

- Gaussian-Legendre Quadrature

$$\int_{-1}^1 f(x)\;\mathrm{d}x = \sum_{i=1}^{N} w_i f(x_i) \tag{3.23}$$

where weight w_i and abscissa x_i are given in Table 3.3.

These formula work well if $f(x)$ behaves like a polynomial as $x \to \infty$. It fails if $f(x)$ behaves like an exponential function.

EXAMPLE 3.4 Magical Gaussian Quadrature

The energy density of blackbody radiation at temperature T is given by

$$u(T) = \frac{8\pi(kT)^4}{(hc)^3} J \tag{3.24}$$

where h, c, and k are Plank constant[1], speed of light, and Boltzmann constant[2]. The factor J is a dimensionless constant determined by integral:

$$J = \int_0^\infty \frac{x^3}{\mathrm{e}^x - 1}\,\mathrm{d}x = \frac{\pi^4}{15} \tag{3.25}$$

Let us try to integrate it numerically and compare the result with the exact value. Since the integral bounds are 0 and ∞, we use the Gaussian-Laguerre quadrature. To use the Gaussian-Laguerre quadrature, the integrand must have e^{-x}. We can always create e^{-x} by multiplying $\mathrm{e}^x\,\mathrm{e}^{-x}$ to the integrand. Now we have the desired exponential function e^{-x}, but as a penalty another exponential function e^x is introduced in the integrand which is now $f(x) = \dfrac{x^3\mathrm{e}^x}{\mathrm{e}^x - 1}$. We need to make it sure that the extra e^x does not cause a problem. Since $f(x) \to x^3$ as $x \to \infty$, $f(x)$ behaves like a polynomial and thus the Gaussian quadrature is expected to give a good result. Program 3.3 calculate J using the Gauss-Laguerre quadrature. The output is

```
8 points Gaussian Laguerre Quadrature
Exact= 6.493939402267e+00
Gauss= 6.493935665353e+00
Error= 3.736914144348e-06
```

Despite that we evaluated the integrand only at 8 points, the agreement with the exact value is remarkable.

3.6 Applications in Physics

3.6.1 Period of Classical Oscillation I.

A classical particle with energy E is confined in a potential $U(x)$. (See Fig 3.4.) The particle oscillates between turning points x_1 and x_2. The period of oscillation[3] is given by

$$T = 2\int_{x_1}^{x_2} \frac{1}{v(x)}\,\mathrm{d}x \tag{3.26}$$

where the speed of the particle at x is given by

$$v(x) = \sqrt{\frac{2(E - U(x))}{m}} \tag{3.27}$$

The integral bounds are determined by solving $v(x) = 0$, which requires numerical root finding discussed in next Chapter.

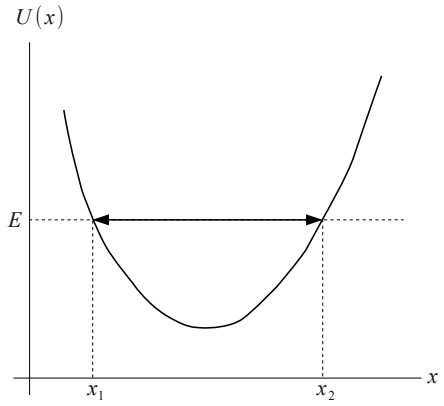


Figure 3.4: Classical Oscillation

The integral in Eq. (3.26) is improper since $v(x_1) = v(x_2) = 0$. We need to remove the integral singularities. First we expand the potential around the turning points:

$$U(x) = U(x_i) + U'(x_i)(x - x_i) + o(x^2) = E + U'(x_i)(x - x_i) + o(x^2), \qquad i = 1, 2 \tag{3.28}$$

If the potential is approximated by $U_i(x) = E + U'(x_i)(x - x_i)$, the speed becomes

$$v_i(x) = \sqrt{\frac{-2U'(x_i)(x - x_i)}{m}} \tag{3.29}$$

Note that this approximated speed approaches to the correct speed as $x \to x_i$. Utilizing it, we remove the singularities as follows.

$$\int_{x_1}^{x_2} \frac{1}{v(x)}\,\mathrm{d}x = \int_{x_1}^{x_0} \left[\frac{1}{v(x)} - \frac{1}{v_1(x)}\right]\mathrm{d}x + \int_{x_1}^{x_0} \frac{1}{v_1(x)}\,\mathrm{d}x \tag{3.30}$$

$$+ \int_{x_0}^{x_2} \left[\frac{1}{v(x)} - \frac{1}{v_2(x)}\right]\mathrm{d}x + \int_{x_0}^{x_2} \frac{1}{v_2(x)}\,\mathrm{d}x \tag{3.31}$$

$$\tag{3.32}$$

where x_0 is any point between x_1 and x_2. A good choice would be a point where the potential is minimum. The integrands inside the square brackets have no singularity and thus can be integrated using a common numerical method. The remaining integral can be easily evaluated analytically:

$$\int_{x_1}^{x_0} \frac{1}{v_1(x)}\,\mathrm{d}x = \sqrt{\frac{m}{-2U'(x_1)}} \int_{x_1}^{x_0} \frac{1}{\sqrt{x - x_1}}\,\mathrm{d}x = \sqrt{\frac{2mx_0}{|U'(x_1)|}} \tag{3.33}$$

$$\int_{x_0}^{x_2} \frac{1}{v_2(x)}\,\mathrm{d}x = \sqrt{\frac{m}{2U'(x_2)}} \int_{x_0}^{x_2} \frac{1}{\sqrt{x_2 - x}}\,\mathrm{d}x\ . = \sqrt{\frac{2mx_0}{|U'(x_2)|}} \tag{3.34}$$

3.6.2 Scattering by Yukawa Potential: Part 1

A particle of mass m_1 elastically collides with another particle of mass m_2 through a spherical potential $U(r)$ where r is the distance between the particles. The scattering angle θ defined in Fig. 3.5 depends on

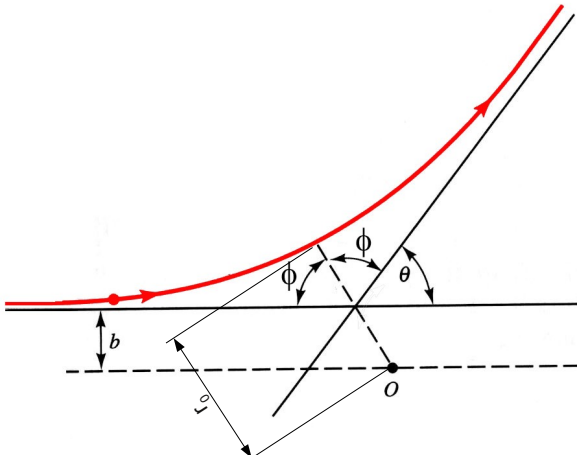


Figure 3.5: Geometry of scattering in relative coordinate.

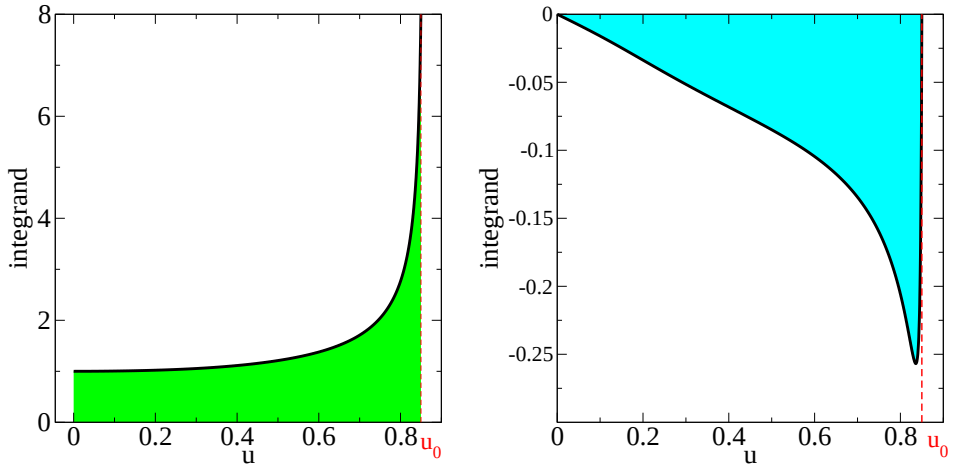


Figure 3.6: The left panel shows the original integrand. The green area need to be numerically integrated. The right panel shows the integrand after the singularity is removed. The blue area need to be integrated. Note the difference in scale between two plots. The blue area is much smaller than the green area. Parameter values $k = a = E = 1$ are used.

the impact parameter b and the energy of the system E. For mathematical convenience, we introduce a new angle ϕ as shown in Fig. 3.5. Note that $\theta = \pi - 2\phi$. This scattering problem can be analytically solved up to the following integral[4]:

$$\phi = \int_{r_0}^{\infty} \frac{b}{r^2} \frac{1}{\sqrt{1 - \dfrac{b^2}{r^2} - \dfrac{U(r)}{E}}} \, \mathrm{d}r \ . \tag{3.35}$$

The lower integral limit r_0 is the closest distance between two particles determined by the equation

$$1 - \frac{b^2}{r_0^2} - \frac{U(r_0)}{E} = 0 \, . \tag{3.36}$$

For the Coulomb potential this integral can be analytically carried out.(Rathurford scattering).[4] We want to find the scattering angle for Yukawai potential (screened Coulomb potential):

$$U(r) = \frac{k}{r}\mathrm{e}^{-r/a}$$

where k and $a > 0$ are constant.

This integral is improper in two reasons. One is that the upper integral limit is infinity. The other is that the integrand diverges at $r = r_0$. See the singularity in Fig. 3.6. The first difficulty can be easily resolved. Introducing a variable $u = \dfrac{1}{r}$. Eqs. (3.35) and (3.36) are respectively transformed to

$$\phi = \int_0^{u_0} \frac{b}{\sqrt{1 - b^2u^2 - \dfrac{U(1/u)}{E}}}\, du \tag{3.37}$$

and

$$1 - b^2u_0^2 - \frac{U(1/u_0)}{E} = 0\,. \tag{3.38}$$

Removing the singularity is a bit more difficult. The method used in Example 3.3 is helpful. Noting that the Rutherford scattering can be solved analytically, we consider scattering by Coulomb potential $U_{\mathrm{C}}(r) = \dfrac{k\mathrm{e}^{-r_0/a}}{r}$. Expressing in variable u, the scattering angle by U_{C} is analytically obtained:

$$\phi_C = \int_0^{u_0} du \frac{b}{\sqrt{1 - b^2u^2 - c\,u}} \tag{3.39}$$

$$= \sin^{-1}\left(\frac{2b^2u_0 + c}{\sqrt{c^2 + 4b^2}}\right) - \sin^{-1}\left(\frac{c}{\sqrt{c^2 + 4b^2}}\right)\,. \tag{3.40}$$

where $c = \dfrac{ke^{-r_0/a}}{E}$. Now the scattering angle by the Yukawa potential is given by $\phi = \phi_{\mathrm{C}} + \Delta\phi$ where

$$\Delta\phi = \phi - \phi_{\mathrm{C}} \tag{3.41}$$

$$= \int_0^{u_0} du \left[\frac{b}{\sqrt{1 - b^2u^2 - c\,u\,\mathrm{e}^{-(1/u - 1/u_0)/a}}} - \frac{b}{\sqrt{1 - b^2u^2 - c\,u}}\right] \tag{3.42}$$

The integrand in the square bracket is no loner singular at $u = u_0$ because the singularity in the two terms is exactly canceled. Hence, $\Delta\phi$ can be evaluated by a standard numerical integral. Figure 3.6 shows that the integrand of Eq. (3.42) does not have the divergence any more. We can use simple numerical integration algorithms such as the trapezoidal or the Simpson rule to integrate Eq. (3.42). One issue is that the integrand has sharp change near u_0. Try N=100, 500, 1000 in the Simpson rules. If the results do not change significantly, you have sufficiently accurate results.*

We are still not ready to write a program yet. We must find r_0 by solving Eq. (3.36) or (3.38). Unfortunately, there is no analytical solution for the Yukawa potential. We need a numerical root finding method, which we will discuss in next chapter.

3.6.3 Debye Model of Heat Capacity

Based on Debye theory, the heat capacity of a solid at temperature T is given by

$$C_V = 9k_{\mathrm{B}}N\left(\frac{T}{\theta_D}\right)^3 \int_0^{\theta_D/T} \frac{x^4\mathrm{e}^x}{(\mathrm{e}^x - 1)^2}\,\mathrm{d}x \tag{3.43}$$

*We investigate the same problem with a different method in Chapter 4

where θ_D is the Debye temperature, N is the number of atoms, and k_B is Boltzmann's constant.[5] Since the upper limit of the integral depends on temperature, we may need to use different numerical methods for different temperature.

(1) Before going to calculation, first we normalize quantities. The heat capacity of ideal gas consisting of N particles is $\frac{3}{2}k_\mathrm{B}N$. Using this as a unit of heat capacity, the heat capacity of the material is $\tilde{C}_V = 2C_v/3k_\mathrm{B}N$. We measure temperature using the Debye temperature as unit. The normalized temperature is $\tilde{T} = T/\theta_D$. Now, the original expression is written as

$$\tilde{C}_V = 6\tilde{T}^3 \int_0^{1/\tilde{T}} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x - 1)^2}\,\mathrm{d}x \tag{3.44}$$

which does not include very large number. More importantly, this expression does not depend on the actual material (neither N nor θ_D). Therefore, the result is universal.

(2) When the temperature is similar or larger than the Debye temperature ($\tilde{T} \gtrsim 1$), the upper limit of the integral is about 1 or smaller. There is no numerical difficulty to integrate it numerically. The Simpson or trapezoidal rule is sufficient.

(3) When the temperature is exactly zero, the upper limit of the integral is infinity. As $x \to \infty$, the integrand behaves as $x^4\mathrm{e}^{-x}$. Hence, the integral is finite and the Gaussian-Laguerre quadrature should work well. However, it is not necessary to calculate the integral since the factor $\tilde{T}^3$ in front of the integral vanishes. Thus, $\tilde{C}_V = 0$.

(4) When the temperature is much lower than Debye temperature but still above zero, the upper limit of the integrals can very large. The direct integration using the Simpson rule may fail. As we learned in part (3), the Gaussian quadrature works well if the upper limit is infinity. Utilizing it we split the integral in two parts.

$$\int_0^{1/\tilde{T}} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x - 1)^2}\,\mathrm{d}x = \int_0^{\infty} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x - 1)^2}\,\mathrm{d}x - \int_{1/\tilde{T}}^{\infty} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x - 1)^2}\,\mathrm{d}x \tag{3.45}$$

$$= \frac{4\pi^4}{15} - \int_0^{\tilde{T}} \frac{\mathrm{e}^{1/s}}{s^6(\mathrm{e}^{1/s} - 1)^2}\,\mathrm{d}s \tag{3.46}$$

where $s = 1/x$ is used in the second line. The first integral in the lhs is analytically calculated. In the second integral, the upper limit is $\tilde{T} \ll 1$ and the Simpson rules is sufficient. Note that when $s \to 0$, the integrand vanishes.

3.6.4 Heat Capacity of Free Electron Gas

The heat capacity of free electron gas is given by

$$C_\mathrm{e} = \frac{3}{2}k_\mathrm{B}N\,\frac{k_\mathrm{B}T}{\varepsilon_F} \int_{-\varepsilon_F/k_\mathrm{B}T}^{\infty} \frac{x^2 \mathrm{e}^x}{(\mathrm{e}^x + 1)^2}\,\mathrm{d}x \tag{3.47}$$

where ε_F is Fermi energy.[6] See Section 3.6.3 for the meaning of other symbols. Expressing the above equation using the normalized heat capacity $\tilde{C} = 2C/3k_\mathrm{B}N$ and $\tilde{T} = k_\mathrm{B}T/\varepsilon_F$. (see Section 3.6.3), Equation (3.47) is simplied to

$$\frac{\tilde{C}}{\tilde{T}} = \int_{-1/\tilde{T}}^{\infty} \frac{x^2 \mathrm{e}^x}{(\mathrm{e}^x + 1)^2}\,\mathrm{d}x \tag{3.48}$$

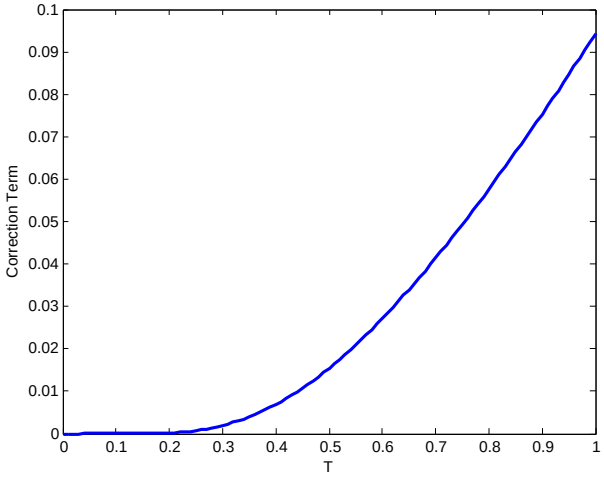


Figure 3.7: Correction term, the integral in Eq. (3.50).

This integral is simpler than Eq. (3.44) since there is no singularity. On the other hand, the upper bound is infinity. For a typical metal at a room temperature or lower, $\tilde{T} \ll 1$ and thus the lower limit is often replaced with $-\infty$. Then, we find an analytical solution

$$\frac{\tilde{C}}{\tilde{T}} \approx \int_{-\infty}^{\infty} \frac{x^2 \mathrm{e}^x}{(\mathrm{e}^x+1)^2}\,\mathrm{d}x = \frac{\pi^2}{3} \tag{3.49}$$

If we want to know the heat capacity at a higher temperature we need to integrate the original equation numerically. However, we don't have to evaluate the whole integral. We just need to find the deviation from the low temperature limit (3.49).

$$\begin{aligned}
\frac{\tilde{C}}{\tilde{T}} &= \frac{\pi^2}{3} - \int_{-\infty}^{-1/\tilde{T}} \frac{x^2 \mathrm{e}^x}{(\mathrm{e}^x+1)^2}\,\mathrm{d}x \\
&= \frac{\pi^2}{3} - \int_{-\infty}^{-1/\tilde{T}} \frac{x^2 \mathrm{e}^x}{(\mathrm{e}^x+1)^2}\,\mathrm{d}x \\
&= \frac{\pi^2}{3} + \int_{0}^{\tilde{T}} \frac{\mathrm{e}^{1/s}}{(\mathrm{e}^{1/s}+1)^2}\,\mathrm{d}s \\
&= \frac{\pi^2}{3} + \int_{0}^{\tilde{T}} \frac{\mathrm{e}^{-1/s}}{(\mathrm{e}^{-1/s}+1)^2}\,\mathrm{d}s
\end{aligned} \tag{3.50}$$

In the last line, $\exp(-2/s)$ is multiplied to both the numerator and denominator so that no large number appears. The integral in the last line can be estimated by a standard numerical method.

Figure 3.47 shows that the correction becomes significant only for $\tilde{T} > 0.2$. The Fermi energy of a typical material is about $1\,eV$ to $10\,eV$, which corresponds to temperature $10^4\,K$ to $10^5\,K$. Hence, the correction term is not necessary until $T = 2000\,K$ to $20000\,K$. At this temperature, the metal is melt. Thus, the correction term may be safely ignored.

Problems

3.1 Integrate $x \cos x$ from $x = 0$ to $x = \pi$ using Trapezoidal and Simpson rules. Compare the results with the exact solution $\int x \cos x \, \mathrm{d}x = \cos x + x \sin x$.

3.2 Plot the molar heat capacity of copper ($\theta_D = 309\,K$) from $T = 0\,K$ to $T = 1083\,K$ using the Debye theory shown in Section 3.6.3.

3.3 Write a code to produce Fig. 3.7.

Appendix

3.I Expansion of Integral with h

Consider the integral as a function of h as

$$F(h) = \int_{x_n}^{x_n+h} f(x)\,\mathrm{d}x \tag{3.51}$$

and expand it in a McLaughlin series

$$F(h) = F(0) + F'(0)h + \frac{1}{2}F''(0)h^2 + \frac{1}{3!}F^{(3)}(0)h^3 + \frac{1}{4!}F^{(4)}(0)h^4 + o(h^5)\,. \tag{3.52}$$

Obviously, $F(0) = 0$. The first derivative of $F(x)$ can be computed as

$$F'(h) = \frac{\mathrm{d}}{\mathrm{d}h}\int_{x_n}^{x_n+h} f(x)\,\mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}z}\int_{x_n}^{z} f(x)\,\mathrm{d}x = f(z) = f(x_n + h) \tag{3.53}$$

where $z = x_n + h$. Higher order derivatives are now simply $F^{(k)}(h) = f^{(k-1)}(x_n + h)$. Substituting these results into the expansion (3.52), we obtain Eq, (3.3).

3.II Justification of Gaussian Quadrature

To be written.

3.III Gaussian Quadrature: Weights and Abscissas

Table 3.1: Weights and Abscissas for Gaussian-Laguerre quadrature

N	x	w
2	$5.8578\ 6437\ 6269\ 0495\times10^{-1}$	$8.5355\ 3390\ 5932\ 7376\times10^{-1}$
	$3.4142\ 1356\ 2373\ 0950$	$1.4644\ 6609\ 4067\ 2624\times10^{-1}$
4	$3.2254\ 7689\ 6193\ 9231\times10^{-1}$	$6.0315\ 4104\ 3416\ 3360\times10^{-1}$
	$1.7457\ 6110\ 1158\ 3466$	$3.5741\ 8692\ 4377\ 9969\times10^{-1}$
	$4.5366\ 2029\ 6921\ 1280$	$3.8887\ 9085\ 1500\ 5384\times10^{-2}$
	$9.3950\ 7091\ 2301\ 1331$	$5.3929\ 4705\ 5613\ 2745\times10^{-4}$
6	$2.2284\ 6604\ 1792\ 6069\times10^{-1}$	$4.5896\ 4673\ 9499\ 6359\times10^{-1}$
	$1.1889\ 3210\ 1672\ 6230$	$4.1700\ 0830\ 7721\ 2099\times10^{-1}$
	$2.9927\ 3632\ 6059\ 3141$	$1.1337\ 3382\ 0740\ 4498\times10^{-1}$
	$5.7751\ 4356\ 9104\ 5105$	$1.0399\ 1974\ 5314\ 9075\times10^{-2}$
	$9.8374\ 6741\ 8382\ 5899$	$2.6101\ 7202\ 8149\ 3206\times10^{-4}$
	$1.5982\ 8739\ 8060\ 1702\times10^{+1}$	$8.9854\ 7906\ 4296\ 2124\times10^{-7}$
8	$1.7027\ 9632\ 3051\ 0100\times10^{-1}$	$3.6918\ 8589\ 3416\ 3753\times10^{-1}$
	$9.0370\ 1776\ 7993\ 7991\times10^{-1}$	$4.1878\ 6780\ 8143\ 4296\times10^{-1}$
	$2.2510\ 8662\ 9866\ 1307$	$1.7579\ 4986\ 6371\ 7181\times10^{-1}$
	$4.2667\ 0017\ 0287\ 6588$	$3.3343\ 4922\ 6121\ 5652\times10^{-2}$
	$7.0459\ 0540\ 2393\ 4657$	$2.7945\ 3623\ 5225\ 6725\times10^{-3}$
	$1.0758\ 5160\ 1018\ 0995\times10^{+1}$	$9.0765\ 0877\ 3358\ 2131\times10^{-5}$
	$1.5740\ 6786\ 4127\ 8005\times10^{+1}$	$8.4857\ 4671\ 6272\ 5315\times10^{-7}$
	$2.2863\ 1317\ 3688\ 9264\times10^{+1}$	$1.0480\ 0117\ 4871\ 5104\times10^{-9}$

Table 3.2: Weights and Abscissas for Gaussian-Hermite Quadrature. Abscissas are symmetric with respect to $x = 0$. Therefore, for every positive abscissa x there is negative one $-x$. Only non-negative abscissas are shown.

N	$\pm x$	w
2	0.7071067811	0.8862269254
3	0	1.1816359006
	1.2247448713	0.2954089751
4	0.5246476232	0.8049140900
	1.6506801238	0.0813128354
5	0	0.9453087204
	0.9585724646	0.3936193231
	2.0201828704	0.0199532420
6	0.4360774119	0.7246295952
	1.3358490740	0.1570673203
	2.3506049736	0.0045300099
7	0	0.8102646175
	0.8162878828	0.4256072526
	1.6735516287	0.0545155828
	2.6519613568	0.0009717812
8	0.3811869902	0.6611470125
	1.1571937124	0.2078023258
	1.9816567566	0.0170779830
	2.9306374202	0.0001996040

Table 3.3: Weights and Abscissas for Gaussian-Legendre Quadrature. Abscissas are symmetric with respect to $x = 0$. Therefore, for every positive abscissa x there is negative one $-x$. Only non-negative abscissas are shown.

N	$\pm x$	w
2	0.5773502692	1
3	0	0.8888888889
	0.7745966692	0.5555555556
4	0.3399810436	0.6521451549
	0.8611363116	0.3478548451
5	0	0.5688888889
	0.5384693101	0.4786286705
	0.9061798459	0.2369268851
6	0.2386191861	0.4679139346
	0.6612093865	0.3607615730
	0.9324695142	0.1713244924
7	0	0.4179591837
	0.4058451514	0.3818300505
	0.7415311856	0.2797053915
	0.9491079123	0.1294849662
8	0.1834346425	0.3626837834
	0.5255324099	0.3137066459
	0.7966664774	0.2223810345
	0.9602898565	0.1012285363

MATLAB Source Codes

Program 3.1

```
%*************************************************************************
%*     Example  3.1                                                      *
%*     filename: ch03pr01.m                                              *
%*     program listing number: 3.1                                       *
%*                                                                       *
%*  This program integrate sin(x) from x=0 to x=pi using rectangular,    *
%*  trapezoidal and simpson methods.  Absolute errors are plotted.       *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  01/14/2019.                                   *
%*************************************************************************

clear all;

% Set the lower and upper bound of the integration
a=0;
b=pi/2;

% Header of the output
display('    Absolute error in various numrical integration')
fprintf(' %3s %21s %24s %24s\n','N','Rectangular','Trapezoidal','Simpson');

% loop over different N
for k=1:10
    N=2^k;
    h(k)=(b-a)/N;

    % evaluate the variable and function
    for i=0:N
         x(i+1)=a+i*h(k);
         f(i+1)=sin(x(i+1));
    end

    % Rectangular rule
    rect=sum(f(1:N))*h(k);

    % Trapezoidal rule
    trap=sum(f(2:N))*h(k) + (f(1)+f(N+1))*h(k)/2;

    % Simpson rule
    simp=(2*sum(f(1:2:N-1))+4*sum(f(2:2:N))-f(1)+f(N+1))*h(k)/3;

    % Evaluation of errors (the exat answer is 1)
    err_rect(k)=abs(1-rect);
    err_trap(k)=abs(1-trap);
    err_simp(k)=abs(1-simp);

    %print out the results
    fprintf('   %5d %24.16e %24.16e %24.16e \n',...
        N,err_rect(k),err_trap(k),err_simp(k));
end

% Order of the errors
h2=h.^2;
h3=h.^3;
h4=h.^4;
```

```
% Plot the results
subplot(1,1,1)
p=loglog(h,err_rect,'d',h,err_trap,'o',h,err_simp,'s',...
    h,h,'--',h,h2,'--',h,h4,'--');

% Format the plot
title('Absolute errors in the various integration methods');
xlabel('h');
ylabel('absolute error');
set(p(1),'Color','green');
set(p(2),'Color','blue');
set(p(3),'Color','red');
set(p(4),'Color','green','LineWidth',2);
set(p(5),'Color','blue','LineWidth',2);
set(p(6),'Color','red','LineWidth',2);
legend(p,{'rectangular','trapezoidal','simpson','h','h^2','h^4'});
legend(p,'Location','SouthEast');
```

▲▲▲

Program 3.2

```
%*********************************************************************
%*     Example 3.3                                                   *
%*     filename: ch03pr02.m                                          *
%*     program listing number: 3.2                                   *
%*                                                                   *
%*     This program integrates 1/(sqrt(x)*(1+x)) from x=0 to x=1     *
%*     by removing singularity at x=0.  Trapezoidal rule is used     *
%*     for the proper part of integral.                              *
%*                                                                   *
%*     Programed by Ryoichi Kawai for Computational Physics Course   *
%*     Revised on 01/07/2014.                                        *
%*********************************************************************

clear all;
a = 1.0;  % upper bound
N = 100;  % number of segments
h = a/N;  % width of segments

% integration of sqrt(x)/(1+x) with trapezoidal rule
S = sqrt(a)/(1+a)/2;  %bundary value devided by 2
for i=1:N-1
    x = i*h;
    f = sqrt(x)/(1+x);
    S = S +f;
end
proper = S*h;  % integral of proper part

singular = 2*sqrt(a);  % singular part

total = singular - proper;
exact = pi - 2*atan(1/sqrt(a));

fprintf('Numerical = %18.12e\n',total);
fprintf('    Exact = %18.12e\n',exact);
```

▲▲▲

Program 3.3

```
%*************************************************************************
%*     Example  3.4                                                      *
%*     filename: ch03pr03.m                                              *
%*     program listing number: 3.3                                       *
%*                                                                       *
%*  This program numerically integrates x^3*exp(x)/(exp(x)-1) from       *
%*  x=0 to infinity using 8-point Gaussian Laguerre Quadrature.          *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  01/06/2014.                                   *
%*************************************************************************
clear all;

N=8;

x=[1.7027963230510100e-1, 9.0370177679937991e-1,...
   2.2510866298661307,    4.2667001702876588,   ...
   7.0459054023934657,    1.0758516010180995e+1,...
   1.5740678641278005e+1, 2.2863131736889264e+1];
w=[3.6918858934163753e-1, 4.1878678081434296e-1,...
   1.7579498663717181e-1, 3.3343492261215652e-2,...
   2.7945362352256725e-3, 9.0765087733582131e-5,...
   8.4857467162725315e-7, 1.0480011748715104e-9];

for i=1:N
    f(i)=x(i)^3*exp(x(i))/(exp(x(i))-1);
end

Gauss=sum(w.*f);
Exact=pi^4/15;
fprintf('%i points Gaussian Laguerre Quadrature\n',N);
fprintf(' Exact=%18.12e\n Gauss=%18.12e\n Error=%18.12e\n',...
Exact,Gauss,abs(Exact-Gauss));
```

Examples in Python

EXAMPLE 3.2 Adaptive Quadrature

A Python package called SciPy has built-in function `quad(func,xmin,xmax)` which uses adaptive quadrature with a default error tolerance.

```
>>> import numpy as np
>>> import scipy.integrate as spint
>>> y=spint.quad(lambda x: (4.0*x-x**2)*np.exp(-2.0*x),0.0,5.0)
>>> print(y)
```

You can create a function and pass it to a subprogram by 'lambda x: (4.0*x-x**2)*np.exp(-2.0*x)'. This is a unique capability of Python.

Python Source Codes

Program 3.1

```
# -*- coding: utf-8 -*-
"""
%*************************************************************************
%*  Example  3.1                                                         *
%*  filename: ch03pr01.py                                                *
%*  program listing number: 3.1                                          *
%*                                                                       *
%*  This program integrate sin(x) from x=0 to x=pi using rectangular,    *
%*  trapezoidal and simpson methods.  Absolute errors are plotted.       *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  01/14/2019.                                   *
%*************************************************************************
"""

import numpy as np
import scipy.integrate as integrate
import matplotlib.pyplot as plt

opt=input("Use SciPy [y/n] ")
if opt=='y':
print("SciPy integrate will be used.\n")
else:
print("SciPy will not be used.\n")

# Set the lower and upper bound of the integration
a=0.
b=np.pi/2.
# Header of the output
print("{0:^75}".format('Absolute error in various numrical integration'))
print("{0:^6} {1:^23} {2:^24} {3:^24} \n"
.format('N','Rectangular','Trapezoidal','Simpson'))

kmax=10
h=np.zeros(kmax+1)
err_rect=np.zeros(kmax+1)
err_trap=np.zeros(kmax+1)
err_simp=np.zeros(kmax+1)

for k in range(0,kmax):
N=2**(k+1)
h[k]=(b-a)/N

x = a + np.linspace(a,b,N+1)
f = np.sin(x)

rect=f[0:N].sum()*h[k]

if opt=='y':
trap=integrate.trapz(f,x)
simp=integrate.simps(f,x)
else:
trap=f[1:N].sum()*h[k]+(f[0]+f[N])*h[k]/2.
simp=(2.0*f[0:N-1:2].sum()+4.0*f[1:N:2].sum()-f[0]+f[N])*h[k]/3.

err_rect[k]=abs(1.-rect)
```

```
err_trap[k]=abs(1.-trap)
err_simp[k]=abs(1.-simp)

print("{0:5d} {1:24.16e} {2:24.16e} {3:24.16e}"
.format(N,err_rect[k],err_trap[k],err_simp[k]))

del x
del f

# Plot data
h2=h**2
h3=h**3
h4=h**4

plt.ioff()
plt.figure(figsize=(6,5))
plt.loglog(h,err_rect, 'og', label='rectangular')
plt.loglog(h,err_trap, 'ob', label='trapezoidal')
plt.loglog(h,err_simp, 'or', label='simpson')
plt.loglog(h,h,'--g',label='$h$')
plt.loglog(h,h2,'--b',label='$h^2$')
plt.loglog(h,h4,'--r',label='$h^4$')
plt.legend(loc=4)
plt.xlabel('h')
plt.ylabel('Integral')
plt.show()
```

▲▲▲

Program 3.2

```
#!/usr/bin/env python3
"""
%*********************************************************************
%*  Example  3.3                                                     *
%*  filename: ch03pr02.py                                            *
%*  program listing number: 3.2                                      *
%*                                                                   *
%*  This program integrates 1/(sqrt(x)*(1+x)) from x=0 to x=1        *
%*  by removing singularity at x=0. Trapezoidal rule is used         *
%*  for the proper part of integral.                                 *
%*                                                                   *
%*     Programed by Ryoichi Kawai for Computational Physics Course.  *
%*     Last modification:  01/11/2017.                               *
%*********************************************************************
"""
import numpy as np

a = 1.0 # upper bound
N = 100 # number of segments
h = a/N # width of segments
# integration of sqrt(x)/(1+x) with trapezoidal rule
S = np.sqrt(a)/(1.0+a)/2.0; # bundary value devided by 2
for i in range(1,N):
    x = i*h
    f = np.sqrt(x)/(1+x)
    S = S +f

proper = S*h # integral of proper part
singular = 2*np.sqrt(a)
# singular part
total = singular - proper
```

```
exact = np.pi - 2*np.arctan(1/np.sqrt(a))
print("Numerical = {0:18.12e}".format(total))
print("    Exact = {0:18.12e}".format(exact))
```

▲▲▲

Program 3.3

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
%*************************************************************************
%*  Example  3.4                                                         *
%*  filename: ch03pr03.py                                                *
%*  program listing number: 3.3                                          *
%*                                                                       *
%*  This program numerically integrates x^3*exp(x)/(exp(x)-1) from       *
%*  x=0 to infinity using 8-point Gaussian Laguerre Quadrature.          *
%*                                                                       *
%*     Programed by Ryoichi Kawai for Computational Physics Course.      *
%*     Last modification:  01/12/2019.                                   *
%*************************************************************************
"""
import numpy as np

def f(x):
    return x**3*np.exp(x)/(np.exp(x)-1.)

if __name__ == "__main__":
    N=8
# evaluation points and weights for 8 point Gaussian quadrature
    x=np.array([1.702796323051010e-1, 9.037017767993799e-1,
                2.2510866298661307,    4.2667001702876588,
                7.0459054023934657,    1.0758516010180995e+1,
                1.5740678641278005e+1, 2.2863131736889264e+1])

    w=np.array([3.6918858934163753e-1, 4.1878678081434296e-1,
                1.7579498663717181e-1, 3.3343492261215652e-2,
                2.7945362352256725e-3, 9.0765087733582131e-5,
                8.4857467162725315e-7, 1.0480011748715104e-9])

    gauss=(w*f(x)).sum()  #Gaussian quadrature
    exact=np.pi**4/15.
    print("{0:3d} point Gaussian Laguerre Quadrature".format(N))
    print("  Exact={0:18.12e}\n Gauss={1:18.12e}\n  Error={2:18.12e}"
          .format(exact,  gauss, abs(exact-gauss)))
```

Bibliography

[1] David Griffiths. *Introduction to Quantum Mechanics*. Pearson Prentice Hall, 2nd edition, 2005.

[2] Stephen J. Blundell and Katherine M. Blundell. *Concepts in Thermal Physics*. Oxford University Press, 2nd edition, 2010.

[3] John R. Taylor. *Classical Mechanics*. University Science Books, 2005.

[4] Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics*. Addison Wesley, 3rd edition, 2002.

[5] Charles Kittel. *Introduction o Solid State Physics*. Wiley, 8th edition, 2004. Chapter 5.

[6] Charles Kittel. *Introduction o Solid State Physics*. Wiley, 8th edition, 2004. Chapter 6.