Problem 1)

A set of three integrals for testing numerical integration:

$$\int_0^1 x^3 \, \mathrm{d}x = \frac{1}{4}, \quad \int_0^1 e^x \, \mathrm{d}x = e - 1, \quad \int_0^\pi \sin x \, \mathrm{d}x \,. \tag{1}$$

(a) Write a program (that will be your code 1) that calculates an integral with a given integrand $f(x)$ in the region $[a, b]$ by one of the Newton-Cotes rules with $n \geq 1$ (the trapezoid rule, or 3-point Simpson's rule, or 5-point Boole's rule or higher) and test it on the set of three "test" integrals. Explore how the accuracy changes with the number of integrals.

(b) Upgrade/modify your code (that will be your code 2) to include the error estimate based on calculations for two step sizes h and $h/2$. You can find the error estimation in the lecture notes (see "Extrapolation and Romberg integration"). Test your code with the set of three test integrals. Now you have a tool to estimate accuracy of numerical integration.

(c) Write a program (that will be your code 3) that calculates an integral from $f(x)$ in the region $[a, b]$ by using Gauss quadratures for 10 points. Coefficients for Gauss quadratures can be found in Abramowitz and Tegun "Handbook of Mathematical Functions", or on the web. Again, test your code with the three test integrals.

(d) Find an integration routine (preferably adaptive) you can use with the language of your choice (Python, C++, or MatLab). That will be your code 4. Again, test your routine with the three test integrals.

(a) In general, we would like to approximate the integral

$$I = \int_a^b f(x) \, \mathrm{d}x \,, \tag{2}$$

which can be done as a weighted sum

$$I \approx \sum_i w_i f(x_i). \tag{3}$$

Of course, it befalls us to choose a good set of x_i and weights w_i in order to obtain a good approximation for I. A generic Newton-Cotes integration rule is derived by approximating the function $f(x)$ over the integration domain with an interpolating polynomial $L(x)$:

$$I = \int_a^b f(x) \, \mathrm{d}x \approx \int_a^b L(x) \, \mathrm{d}x \,. \tag{4}$$

In the Lagrange interpolation scheme, if we have a set of sample points $\{(x_0, f(x_0)), \ldots, (x_n, f(x_n))\}$, we can interpolate $f(x)$ with an n^{th} order polynomial

$$L(x) = \sum_{i=0}^{n} f(x_i) l_i(x), \tag{5}$$

where $\{l_i(x)\}$ is the interpolating basis and $l_i(x_j) = \delta_{ij}$. Thus, with the Newton-Cotes scheme

$$I \approx \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)\,\mathrm{d}x\,. \tag{6}$$

Thus, the weights are just the integrals of the interpolating polynomials. For this work, we implement the simplest Newton-Cotes rule, which is equivalent to the trapezoid rule. We first break the integration region into N intervals, giving us a sub-interval width $h = (b-a)/N$ and $N+1$ sample points $\{a = x_0, \ldots, x_i, x_{i+1}, \ldots, x_N = b\}$. Using a simple linear interpolation for the sub-interval $[x_i, x_{i+1}]$, our Lagrange basis is

$$l_1(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} = -\frac{x - x_{i+1}}{h}, \quad l_2(x) = \frac{x - x_i}{h}, \tag{7}$$

and

$$\int_{x_i}^{x_{i+1}} l_1(x)\,\mathrm{d}x = -\frac{(x - x_{i+1})^2}{h}\bigg|_{x_i}^{x_{i+1}} = \frac{h}{2} \tag{8}$$

$$\int_{x_i}^{x_{i+1}} l_2(x)\,\mathrm{d}x = \frac{(x - x_i)^2}{h}\bigg|_{x_i}^{x_{i+1}} = \frac{h}{2}. \tag{9}$$

Thus, over the sub-interval $[x_i, x_{i+1}]$

$$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \approx \frac{h}{2}(f(x_{i+1}) + f(x_i)). \tag{10}$$

If we now sum over all the sub-intervals, we find

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x = \sum_{i=0}^{N} \int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x &\approx \frac{h}{2}\sum_{i=0}^{N}[f(x_i) + f(x_{i+1})] \\ &= \frac{h}{2}[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)]. \end{aligned} \tag{11}$$

(b) Romberg integration is a method which uses the following motivation. Let us denote our approximate integral, which is a function of the step-size h of our sub-intervals, as $I(h)$, where the exact result

$$I = I(h) + \sum_{k=0}^{\infty} a_k h^{n+km}, \tag{12}$$

where n, m are some integers. In particular, if we compute our integral at two step-sizes h and h/N, we have

$$I = I(h) + a_0 h^n + \sum_{k=1}^{\infty} a_k h^{n+km} \tag{13}$$

$$I = I(h/N) + a_0 (h/N)^n + \sum_{k=1}^{\infty} a_k (h/N)^{n+km}. \tag{14}$$

We then have

$$0 = I(h) - I(h/N) + a_0 h^n \Big(1 - \frac{1}{N^n}\Big) \Rightarrow a_0 h^n = \frac{N^n}{N^n - 1}\Big[I(h/N) - I(h)\Big], \tag{15}$$

where we have neglected $\mathcal{O}(h^{n+m})$ terms. This is effectively gives us an estimate of our trapezoidal integrations and an extrapolated value for the integral

$$I \approx I(h/N) + a_0 (h/N)^n = I(h/N) + \frac{1}{N^n - 1}\Big[I(h/N) - I(h)\Big]. \tag{16}$$

(c) We now discuss Gauss-Legendre quadrature. First, we make the change of variables $x = \frac{b-a}{2}t + \frac{a+b}{2}$ such that

$$I = \int_a^b f(x)\,\mathrm{d}x = \int_{-1}^1 \frac{b-a}{2} f(x(t))\,\mathrm{d}t\,. \tag{17}$$

Again, we would like to find suitable weights and sample points $\{(w_i, x_i)\}$ such that

$$I \approx \sum_{i=1}^n w_i f(x_i). \tag{18}$$

Our inspiration here is the Taylor expansion

$$f(x(t)) = \sum_{n=0}^{2N-1} a_n t^n + \mathcal{O}(t^{2N}) \approx p_{2N-1}(t), \tag{19}$$

over the interval $[-1, 1]$. At this point, we would like to define our weights and sample points such that

$$\int_{-1}^1 p_{2N-1}(t)\,\mathrm{d}t = \sum_{i=1}^N w_i p_{2N-1}(t_i). \tag{20}$$

Following through, we have

$$\sum_{n=0}^{2N-1} a_n \frac{1-(-1)^{n+1}}{n+1} = \sum_{i=0}^N w_i \sum_{n=0}^{2N-1} a_n t_i^n \Rightarrow \sum_{i=1}^N w_i t_i^n = \frac{1+(-1)^n}{n+1} \tag{21}$$

for all $n \in \{0, \dots, 2N-1\}$. We thus have $2N$ – not necessarily linear – equations for $2N$ unknowns. Let us solve this for a the simplest case of $N = 1$. First, for $N = 1$, we have only two equations

$$\begin{cases} w_1 + w_2 = 2 \\ w_1 t_1 + w_2 t_2 = 0 \\ w_1 t_1^2 + w_2 t_2^2 = \dfrac{2}{3} \\ w_1 t_1^3 + w_2 t_2^3 = 0. \end{cases} \tag{22}$$

Carrying out the algebra, one finds $w_1 = w_2 = 1$ and $t_1 = -t_2 = 1/\sqrt{3}$. One could continue producing weights and sample points for larger N, but it turns out that we can be a little bit more clever in how we determine them. While the details are beyond the scope of this report, we can choose our points t_i such that they are roots of the N^{th} Legendre polynomial, $P_N(t)$. That is $P_N(t_i) = 0$. These roots have several nice properties, including that they are symmetrically distributed about $t = 0$ and all distinct. The corresponding weights are given by

$$w_i = \frac{2}{(1 - t_i^2)[P_N'(t_i)]^2}. \tag{23}$$

These weights and roots are tabulated within the *numpy* library, and as such, we can pulled up to an arbitrarily large N. Before concluding, it should be noted that the sucess of this method depends on the behavior of our integrand. That is, our function and its derivatives should be reasonably smooth in order to be well approximated by a sufficiently low-order polynomial over the integration range of interest.

Problem 2)

Evaluate the following integrals using your codes 2, 3, and 4:

(a) $\displaystyle\int_0^1 \frac{1}{1 - 0.998x^2}\,\mathrm{d}x$

(b) $\displaystyle\int_0^{2\pi} x\sin(30x)\cos(50x)$

(c) $\displaystyle\int_0^1 \frac{x}{e^x - 1}\,\mathrm{d}x$

(d) $\displaystyle\int_0^1 x\sin\left(\frac{1}{x}\right)\mathrm{d}x$

Report if one of your codes fails to compute some of the integrals. Explain why.

Problem 3)

Evaluate numerically following improper integrals (you may need to modify one of your codes). There are multiple ways to evaluate improper integrals. Explain your choice and results.

(a) $\displaystyle\int_0^{\infty} \frac{e^{-x^2}}{x^2 + 1}\,\mathrm{d}x$

(b) $\displaystyle\int_0^{\infty} \frac{x\sin x}{x^2 + 1}\,\mathrm{d}x$

(c) $\displaystyle\int_0^\infty e^{-\sqrt{x}}\cos(2x)\,\mathrm{d}x$

While there are nicer methods to handle improper integrals such as these, we can make a simple substitution to map our infinite domain of integration onto a finite domain:

$$x = \tan u \Rightarrow \int_a^b f(x)\,\mathrm{d}x = \int_{\arctan(a)}^{\arctan(b)} \frac{f(\tan u)}{\cos^2 u}\,\mathrm{d}u\,. \tag{24}$$

Notice that if $a = \infty$, then $\arctan a = \pi/2$, or if $b = -\infty$, then $\arctan b = -\pi/2$. With this, we can employ the integration techniques from Problem 1 with some caveats.

Problem 4)

Evaluate numerically following principal value integrals using one of techniques for princpal value integrals (again, you may need to modify one of your codes).

(a) $\displaystyle\int_0^1 \frac{1}{x^{1/3}}\,\mathrm{d}x$

(b) $\displaystyle\int_{-1}^1 \left(1+\frac{1}{x}\right)\mathrm{d}x$

(c) $\displaystyle\int_{-\infty}^\infty \frac{1}{(x-1)(x^2+1)}\,\mathrm{d}x$

Problem 5)

Evaluate numerically following double integrals.

(a) $\displaystyle\int_{-1}^1 \mathrm{d}x \int_0^2 \mathrm{d}y\,\sin\!\left(x^2+y^2\right)$

(b) $\displaystyle\int_0^1 \mathrm{d}x \int_0^{1-x} \mathrm{d}y\,\frac{1}{\sqrt{x+y}(1+x+y)^2}$

For these multi-dimensional integrations, we utilize the iterative intregration scheme. That is, our integral of interest is

$$I = \int_a^b \mathrm{d}x \int_{y_1(x)}^{y_2(x)} \mathrm{d}y\, f(x,y) = \int_a^b \mathrm{d}x\, F(x), \tag{25}$$

where

$$F(x) = \int_{y_1(x)}^{y_2(x)} \mathrm{d}y\, f(x, y). \tag{26}$$

Now we can set up $F(x)$ as a function, which employs one integration over the variable y at a given x, and set up a second integration of $F(x)$. While Gaussian quadrature performs nicer than Monte Carlo integration in one dimension over a large set of functions, one can appreciate here the utility of Monte Carlo methods for higher-dimensional integrations. If one uses Gaussian quadrature with n points for each integration dimension, then in total we will require n^D points for an integral over D variables, which is exponentially poor. However, as we have learned previously, if we sample the integrand, we may hope to reduce the number of points needed to evaluate the integral.