ECON6080/8080 Assignment 4

Total marks: 10 marks. Marks will depend on accuracy and clarity. Speed is not an issue.

Use Python to answer all questions.

Q1 [2.5 marks]

Below is the table taken from Judd's textbook, where $(x + 0.05)^+ = \max(0, x + 0.05)$.

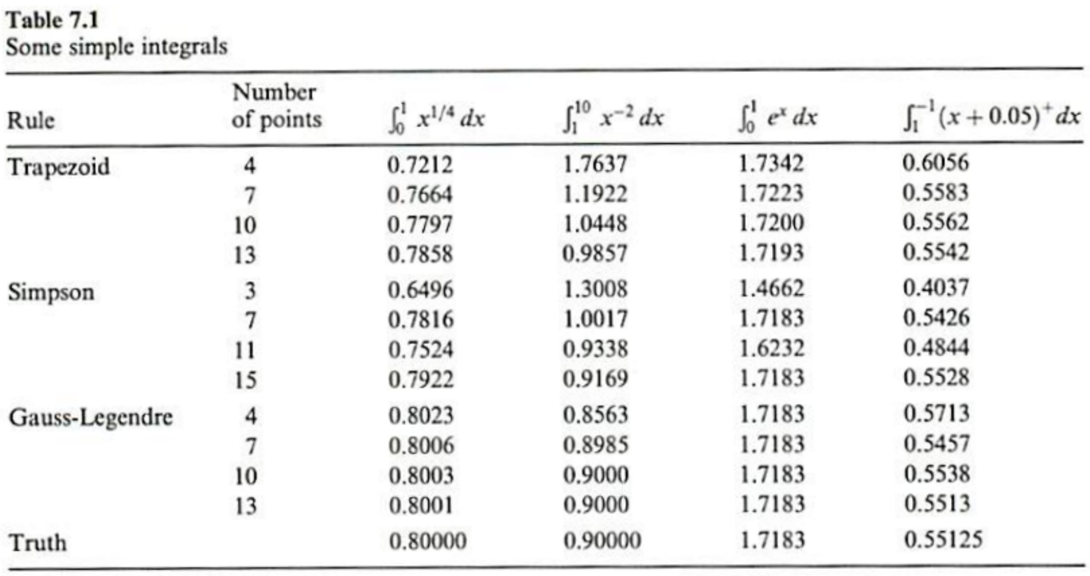

Table 7.1
Some simple integrals

Rule	Number of points	$\int_0^1 x^{1/4}\,dx$	$\int_1^{10} x^{-2}\,dx$	$\int_0^1 e^x\,dx$	$\int_1^{-1} (x+0.05)^+\,dx$
Trapezoid	4	0.7212	1.7637	1.7342	0.6056
	7	0.7664	1.1922	1.7223	0.5583
	10	0.7797	1.0448	1.7200	0.5562
	13	0.7858	0.9857	1.7193	0.5542
Simpson	3	0.6496	1.3008	1.4662	0.4037
	7	0.7816	1.0017	1.7183	0.5426
	11	0.7524	0.9338	1.6232	0.4844
	15	0.7922	0.9169	1.7183	0.5528
Gauss-Legendre	4	0.8023	0.8563	1.7183	0.5713
	7	0.8006	0.8985	1.7183	0.5457
	10	0.8003	0.9000	1.7183	0.5538
	13	0.8001	0.9000	1.7183	0.5513
Truth		0.80000	0.90000	1.7183	0.55125

Figure 1: Table for Q1

Reproduce the 16 numbers for the Gauss-Legendre quadrature in the table as follows. First calculate the weights and nodes using the `numpy.polynomial.legendre.leggauss` function for $n = 4, 7, 10, 13$. Then compute the approximate integral using $\sum_{i=1}^{n} f(x_i)w_i$ for the four functions. You are not allowed to use the `scipy.integrate.quad` function.

Q2 [2.5 marks]

Use the Gauss-Hermite quadrature to approximate the mean, the variance, the skewness, and the kurtosis of a normal distribution $N(\mu, \sigma^2)$ for N (the number of nodes) in $\{3, 5, 10, 15\}$. Compare them with the true values.

You can choose (μ, σ^2) freely, but are not allowed to use the `scipy.integrate.quad` function. Proceed as in Q1, but use the `numpy.polynomial.hermite.hermgauss` function instead when computing the nodes and weights.

Q3 [2.5 marks]

In the lecture we only casually checked that the Monte Carlo integration has some non-negligible variance. Here we do it more formally.

We approximate

$$\int_0^1 x^{1/4} dx = 0.8$$

using simulation.

In order to evaluate the error variance, we will run K simulations, each of which has the sample size of N. Hence we have $\{\{x_n(k)\}_{n=1}^N\}_{k=1}^K$, and

$$I(k) = \frac{1}{N} \sum_{n=1}^{N} f(x_n(k))$$

is an estimate of the integral for k-th simulation. We use a sample variance of $\{I(k)\}_{k=1}^K$ as an estimate of Monte Carlo integration variance.

Fix $K = 1000$. Using the uniform random number generator and $N = 10$, 100, and 1000, examine how the sample variance changes.

I suggest you to fix the seed to make sure that I get the same result when I run your code.

Q4 [2.5 marks]

In the lecture, we applied the Random Walk Metropolis Hastings algorithm to a sample from a normal distribution with unknown mean and known variance.

Here we consider a sample from a normal distribution with KNOWN mean and UNKNOWN variance. Because the standard deviation σ must be positive, it is common to use the inverse Gamma distribution (it puts zero probability to negative values) as its prior.

Implement a Metropolis Hastings algorithm for this case (it doesn't have to be the Random Walk one). Note that the analytical posterior will be different from the case with known variance, and you do not need to compare the result with the analytical posterior. (It is available. See e.g. Chapter 12 in `https://eml.berkeley.edu/books/choice2.html` if you are interested.)