1 Basics in Numerical Methods

1.1 Gauss-Hermite Quadrature Pricing

The question we were given was to assume that a simple derivative has the form of $\Phi(S_T) = \frac{1}{S_T}$, and calculate its price F(t,s) using Gauss-Hermite quadrature with 100 uniform nodes of $s \in [1, 3]$. The underlying assumption is that the stock price follows a Geometric Brownian Motion, which makes our task to approximate the Riemann integral using quadrature and solve the following Stochastic Differential Equation (SDE):

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t$$

To solve for the GBM SDE over a time interval [0, T] on the risk-neutral underlying stock, we use:

$$S_T = S_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{(T-t)}z\right)$$

, where z is a standard normal distributed variable. Next we note that our derivative has the price of the form:

$$F(t,s) = e^{-r(T-t)} \, E^Q \left[\Phi(S_T)\right]$$

Where we can write the expression for $\Phi(S_T)$ as:

$$\Phi(S_T) = \frac{1}{S_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{T-t}\,z\right)}$$

Thus, the expectation can be expressed as follows:

$$E^Q[\frac{1}{S_T}] = \int_{-\infty}^{\infty} \frac{1}{S_t \exp\left(\left(r - \frac{1}{2}\sigma^2\right)(T-t) + \sigma\sqrt{T-t}\,z\right)} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \, dz = E[f(X)]$$

Gauss-Hermite Quadrature approximates the integral as follows:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) \, dx \approx \sum_{i=1}^{N} w_i \, f(x_i)$$

We make the substitution in our E[f(X)] :

$$x = 2z \quad \Rightarrow \quad z = 2x \quad \text{and} \quad dz = 2\,dx$$

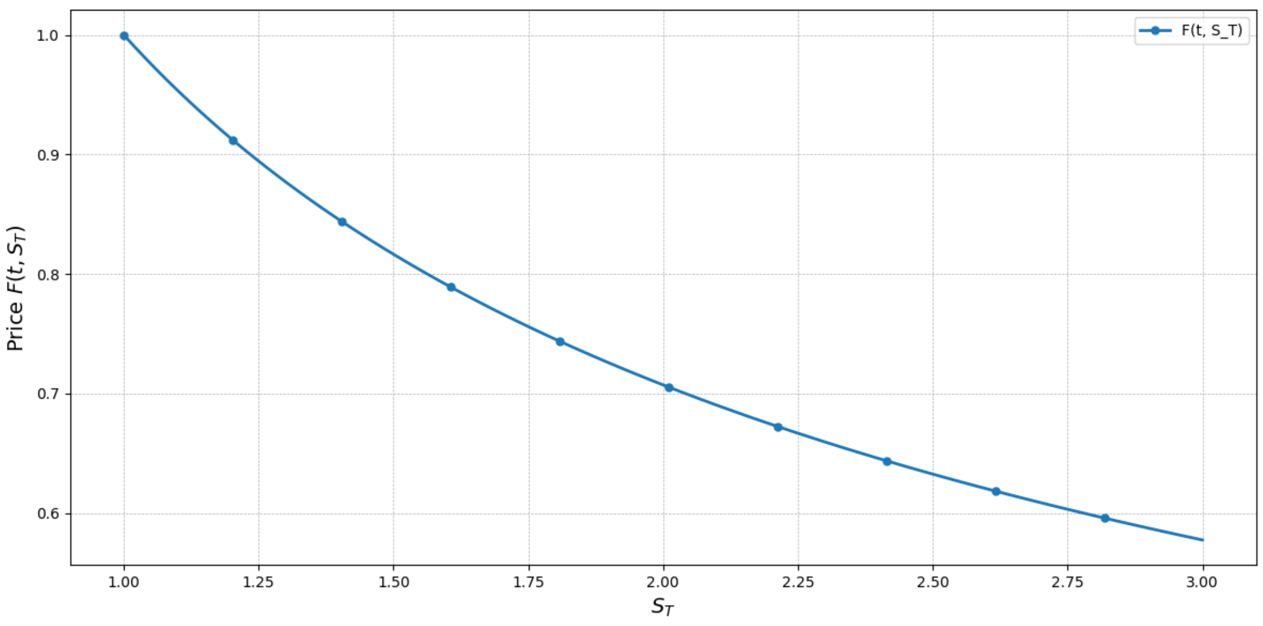
Our function $f(x)$ in GHQ then becomes:

$$f(x) = e^{-\sigma\sqrt{2(T-t)}\,x}$$

The full Gaussian Hermite Quadrature (GHQ) approximation will therefore become as follows:

$$\frac{1}{S_t\sqrt{\pi}} \cdot \exp\left(-\left(r - \frac{1}{2}\sigma^2\right)(T-t)\right) \sum_{i=1}^{N} w_i \, e^{-\sigma\sqrt{2(T-t)}\,x_i}$$

Importing the necessary nodes x_i and weights w_i online into Python, and approximating the integral using Gaussian Hermite Quadrature, I obtained the following plot:



1.2 Gauss-Laguerre Quadrature Approximation

This question we were asked was to compute the discounted utility U:

$$U(\lambda) = \int_0^\infty e^{-\rho t} \log(1 - e^{-\lambda t})\, dt$$

The approximation for the Discounted Utility by Gauss-Laguerre Qaudrature Approximation is the following:

$$\int_0^\infty e^{-x} f(x)\, dx \approx \sum_{i=1}^{N} w_i f(x_i)$$

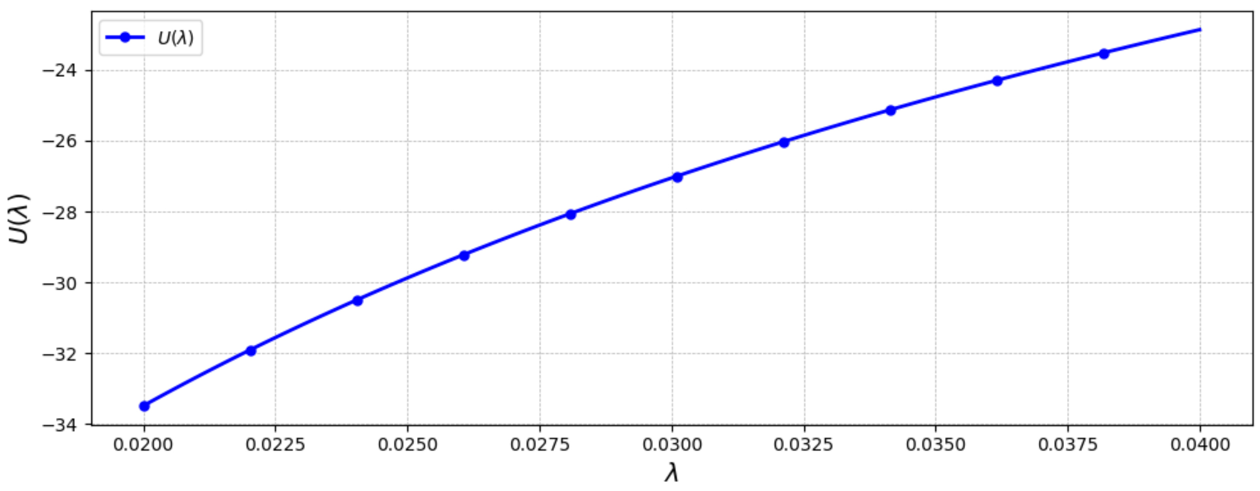
By variable substitution, we can change our Discounted Utility equation. Let $u = \rho t$, then $t = \frac{u}{\rho}$ and $dt = \frac{du}{\rho}$. Thus, the equation becomes:

$$U(\lambda) = \frac{1}{\rho} \int_0^\infty e^{-u} \ln\left(1 - e^{-\frac{\lambda}{\rho} u}\right)\, du$$

Applying Gaussian-Laguerre Quadrature, we approximate $U(\lambda)$ as:

$$U(\lambda) \approx \frac{1}{\rho} \sum_{i=1}^{N} w_i \ln\left(1 - e^{-\frac{\lambda}{\rho} x_i}\right)$$

We download the weights w_i and nodes x_i from online packages into Python, and for 100 uniform nodes of $\lambda \in [0.02, 0.04]$ where $\rho = 0.02$, I obtained the following plot:



1.3 Ordinary Differential Equation with Projection method

In this question, we were asked to solve the following differential equation by using projection method:

$$y'(x) = x + y(x),$$

with the initial condition:

$$y(0) = 0.$$

We can start by noting that this ODE has an analytical solution:

$$y'(x) - y(x) = x, \text{and } y(x) = -(x+1) + Ce^x,$$

Which by using:

$$y(0) = -(0+1) + Ce^0, C = 1$$

, thus exact solution is given by:

$$y(x) = e^x - x - 1$$

For the numerical approximation of $y(x)$ we will use a finite sum of basis functions:

$$y_N(x) = \sum_{k=0}^{N} c_k\,\varphi_k(x),$$

or

$$y_N(x) = \sum_{k=0}^{N} c_k x^k,$$

And thus, our derivative of the approximated solution will be:

$$y'_N(x) = \sum_{k=0}^{N} c_k k x^{k-1},$$

Our objective in this task will be to minimize the residual for our approximiation over the interval $[a, b] = [0, 4]$, and for this it is suitable to use the collocation mehtod. We enforce the residual to be zero at specific points (called collocation points): $R(x_i) = y'_N(x_i) - x_i - y_N(x_i) = 0$ for $i = 0, 1, ..., N$. In order to choose the appropiate collocation points, I will use the Chebyshev Nodes in the interval $[a, b] = [0, 4]$, in order to promote numerical stability.

The Chebyshev nodes are computed using:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2k-1)\pi}{2N}\right), \quad k = 1, 2, \ldots, N+1.$$

At each collocation point x_i we will have, after rearranging with respect to x_i, the following equation:

$$(\sum_{k=1}^{N} c_k k x^{k-1}) - (\sum_{k=0}^{N} c_k x^k) = x_i$$

With respect to our initial condition $y(0) = 0$:

$$y_N(0) = \sum_{k=0}^{N} c_k (0)^k = c_0 = 0$$

We can write the system of equations in matrix form, which will be equivalent to solving the Vandermonde Matrix, $Ac = b$, where:

- A is an $(N+1) \times N$ matrix with entries: $A_{i,k} = k x_i^{k-1} - x_i^k$
- b is a vector with entries: $b_i = x_i$
- c is also a vector: $c = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix}^T$

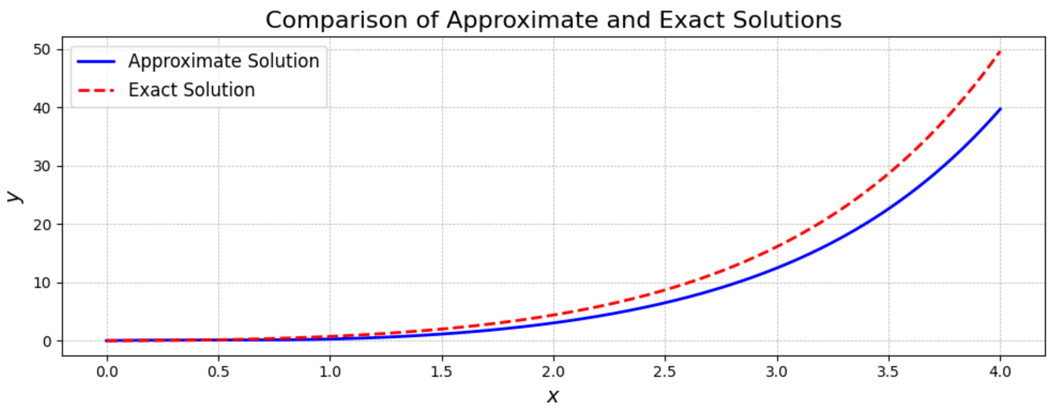
The system of equations in matrix form is:

$$\begin{bmatrix} A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} & A_{0,5} \\ A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{5,1} & A_{5,2} & A_{5,3} & A_{5,4} & A_{5,5} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

And therefore, the problem will become determining $c = A^{-1}b$, which will then be used to construct the linear combination of the monomial basis functions:

$$y_{approx}(x) = c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5$$

Solving this, we will get the approximated solution as shown below:



We can see that the approximated method did not do a quite good fit, compared to the analytical solution. Increasing the monomial basis should, in theory, increase the fit and minimizing the error.

1.4 TPBVP in Life-Cycle Problem

In both task's, forward shooting and projection method, we want to maximize a consumers lifetime utility U:

$$U = \int_0^\infty e^{-\rho t} \log(c_t)\, dt,$$

and our two point boundary value problem (TPBVP) will be defined by the following:

$$\begin{cases} \dot{c}_t = c_t(r - \rho) \\ \dot{a}_t = ra_t - c_t \end{cases}$$

1.4.1 Analytical solution

To solve the optimal control problem analytically, we use the Maximum Principle. The Hamiltonian H is defined as:

$$H = e^{-\rho t} \ln c_t + \lambda_t(ra_t - c_t),$$

where λ_t is the costate variable associated with the wealth constraint.

Our First Order Condition (FOC) is the following:

$$\frac{\partial H}{\partial c_t} = e^{-\rho t} \cdot \frac{1}{c_t} - \lambda_t = 0 \Rightarrow \lambda_t = \frac{e^{-\rho t}}{c_t}.$$

The adjoint equation is:

$$-\dot{\lambda}_t = \frac{\partial H}{\partial a_t} = \lambda_t r \Rightarrow \dot{\lambda}_t = -r\lambda_t.$$

From both the adjoint equation and the FOC, we find the optimal consumption path to be:

$$c_t = c_0 e^{(r-\rho)t},$$

where $c_0 = \frac{1}{\lambda_0}$ is the initial consumption.

To find analytically solution for the asset path, we start by substituting c_t into the state equation, giving us:

$$\dot{a}_t = ra_t - c_0 e^{(r-\rho)t}.$$

To solve, we rewrite it as:

$$\dot{a}_t - ra_t = -c_0 e^{(r-\rho)t}.$$

Using the integrating factor $\mu(t) = e^{-rt}$, we multiply both sides by e^{-rt} and simplify:

$$\frac{d}{dt}\left(e^{-rt} a_t\right) = -c_0 e^{-\rho t}.$$

Integrating both sides from 0 to t, we obtain:

$$e^{-rt} a_t - a_0 = -c_0 \int_0^t e^{-\rho t}\, dt.$$

After computing the integral and rearranging, we get:

$$a_t = e^{rt}\left(a_0 - \frac{c_0}{\rho}(1 - e^{-\rho t})\right).$$

This represents the asset path a_t under the optimal consumption strategy.

1.4.2 Forward Shooting Method

The Forward Shooting Method transforms the TPBVP into a series of IVP's by iteratively guessing the initial conditions (here, c_0), and adjusting them based on the discrepancy at the terminal point. This method is particularly effective for problems where boundary conditions are specified at different points, as is the case here with $a(0) = 1$ and $a(T) = 0$.

To initiate the shooting method, we begin with an initial guess for the initial consumption $c_0(1)$. A common strategy is to set lower and upper bounds based on economic intuition or boundary conditions. For instance:

$$c_{\text{lower}} = 0, \quad c_{\text{upper}} = a_0 = 1.$$

The initial guess is often taken as the midpoint:

$$c_{\text{guess}} = \frac{c_{\text{lower}} + c_{\text{upper}}}{2}.$$

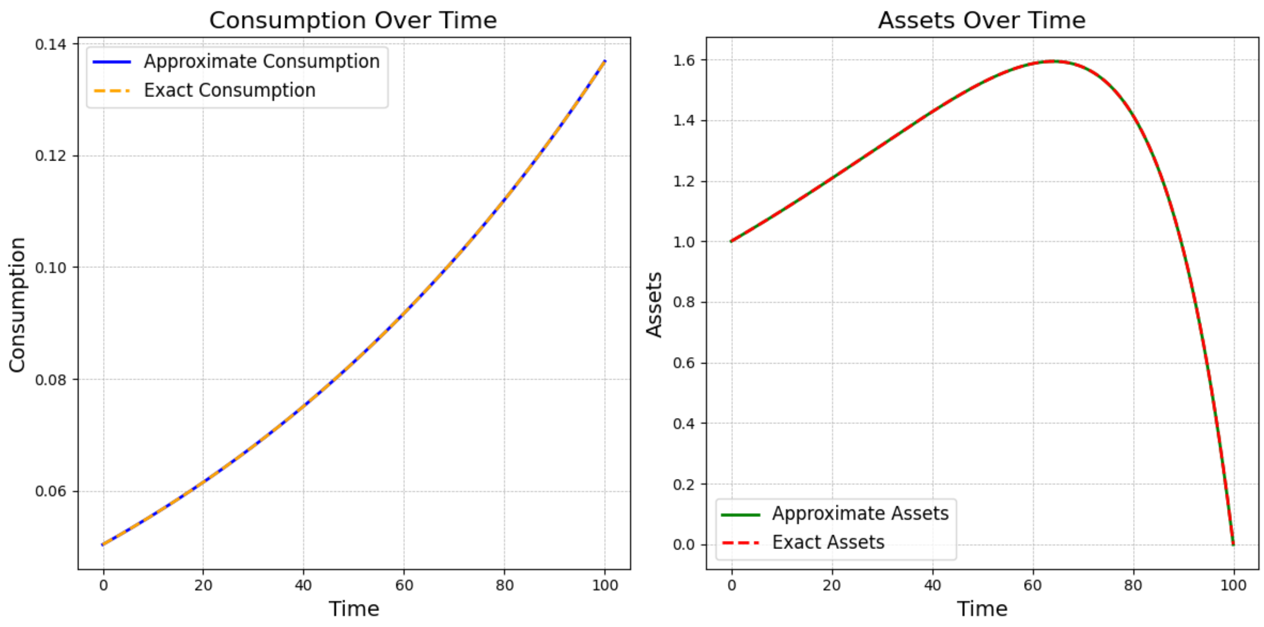
For each guessed c_0, we can express the system of ODEs as follows:

$$f(t, Y) = \begin{bmatrix} \dot{a}_t \\ \dot{c}_t \end{bmatrix} = \begin{bmatrix} ra_t - c_t \\ c_t(r-\rho) \end{bmatrix}.$$

To estimate the solution of the given ODEs, we compute intermediate slopes (k_1, k_2, k_3, k_4) and combine them to update the state vector Y by using RK4 algorithm:

$$
\begin{aligned}
k_1 &= f(t_n, Y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_2\right), \\
k_4 &= f(t_n + h, Y_n + hk_3), \\
Y_{n+1} &= Y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

The forward shooting algorithm then uses the RK4 approach to model the course of consumption and assets over time after assuming an initial consumption estimate. By using a tolerance of 0.001 and a stepsize h equal to 0.01, in addition to the other parameter values given in the task description, we get these plots:



As shown in the plots, the approximation achieved by the forward shooting method, combined with RK4, appears to closely match the exact solution for both asset path and consumption path. This unexpectedly high accuracy suggests a possible issue in the python implementation, as the approximation may be ”too good to be true.” However, I have not been able to pinpoint what causes this overfitting from the Forward shooting method.

1.4.3 Projection Method

Our second sub task, was to solve the same TPBVP by using the Projection method. The Projection method approximates the solution by expressing the asset and consumption paths as polynomial functions and determining their coefficients to satisfy the problem's dynamics and boundary conditions.

To enhance numerical stability and accuracy, Chebyshev nodes are employed as collocation points within the interval $[0, T]$. The n Chebyshev nodes $\{t_k\}_{k=1}^n$ in the interval $[a, b]$ are defined as:

$$t_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, 2, \ldots, n.$$

For the interval $[0, T]$, the Chebyshev nodes become:

$$t_k = \frac{T}{2}\left(1 + \cos\left(\frac{2k-1}{2n}\pi\right)\right), \quad k = 1, 2, \ldots, n.$$

In order to apply the Projection mehtod, we need to rewrite our asset $a(t)$ and consumption $c(t)$ functions, since these will be approximated using polynomial expansions of degree n:

$$a(t) \approx \sum_{k=0}^{n} \alpha_k \cdot t^k \qquad \text{and} \qquad c(t) \approx \sum_{k=0}^{n} \beta_k \cdot t^k.$$

where $\{\alpha_k\}_{k=0}^n$ and $\{\beta_k\}_{k=0}^n$ are the polynomial coefficients to be determined. To determine the coefficients α_k and β_k, we enforce the system's dynamics and boundary conditions at the Chebyshev nodes. At each Chebyshev node t_k, the asset dynamics must satisfy the following, expressed in terms of the polynomial approximations:

$$\sum_{k=1}^{n} k\alpha_k \cdot t_k^{k-1} = r \cdot \left(\sum_{k=0}^{n} \alpha_k \cdot t_k^k\right) - \sum_{k=0}^{n} \beta_k \cdot t_k^k.$$

Similarly, the consumption dynamics must satisfy:

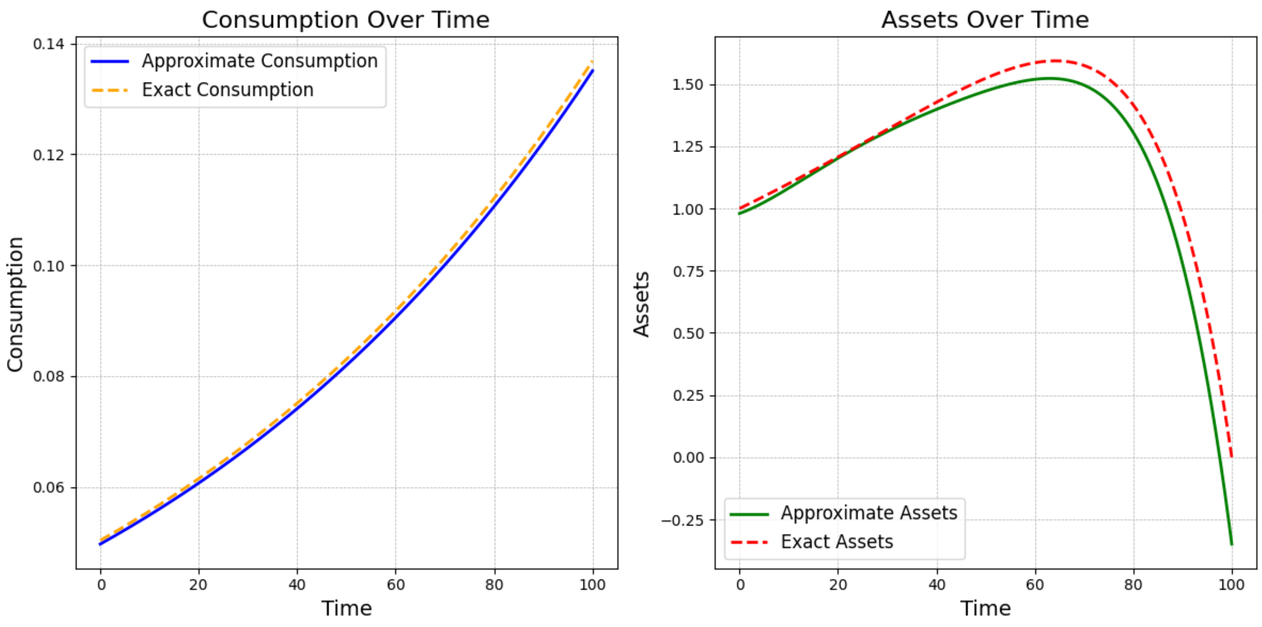
$$\sum_{k=1}^{n} k\beta_k \cdot t_k^{k-1} = (r - \rho) \cdot \left(\sum_{k=0}^{n} \beta_k \cdot t_k^k\right).$$

In addition to these dynamic constraints, we impose the boundary conditions:

$$a(0) = a_0 = \sum_{k=0}^{n} \alpha_k \cdot 0^k = \alpha_0, \tag{1}$$

$$a(T) = a_T = \sum_{k=0}^{n} \alpha_k \cdot T^k. \tag{2}$$

As the same in task 1.3, we can express the problem for the system of equations in matrix form, which will be equivalent to solving the Vandermonde Matrix, $\mathbf{A} \cdot \mathbf{c} = \mathbf{b}$,, which yields: $\mathbf{c} = \mathbf{A}^{-1} \cdot \mathbf{b}$ By these equations, and the parameters expressed from the task description I obtained the following plots:



As we can see from the plots, the Projection method is closely aligning to the exact solution for the consumption and is also able to approximate quite good for the assets. Compared to the Forward shooting method, which we have discussed earlier produces "too good to be true" results, I assume that this Forward shooting algorithm should have produced some similarly results as the Projection method. I have not been able to pinpoint what causes this approximation from the Projection method to be almost identical as the analytical solution for the consumption path.

2 Derivatives Pricing

For this whole section, we had the following Asian option where its price is determined as follows:

$$\text{Option Price} = e^{-rT} E\left[\max\left(\overline{S}_T - K,\, 0\right)\right].$$

Where the underlying asset price $S(t)$ is modeled using the Geometric Brownian Motion (GBM), and is defined by the stochastic differential equation (SDE):

$$dS(t) = rS(t)\,dt + \sigma S(t)\,dW(t),$$

, which has a solution, representing the asset price at time t, is given by:

$$S(t) = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W(t)\right),$$

where S_0 is the initial asset price. For an Asian option with M monitoring dates, the arithmetic average $\overline{S}_T$ is calculated as:

$$\overline{S}_T = \frac{1}{M}\sum_{j=1}^{M} S(t_j),$$

where $t_j = j \cdot \Delta t$ and $\Delta t = \frac{T}{M}$.

2.1 A: Crude Monte Carlo

The Crude Monte Carlo method estimates the option price by simulating a large number of possible price paths, computing the corresponding payoffs, and averaging them. Starting from the initial asset price S_0, compute the asset price at each subsequent time step using the GBM discretization formula:

$$S_{i,j} = S_{i,j-1} \exp\left(\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}Z_{i,j-1}\right),$$

where $S_{i,j}$ is the asset price for simulation i at time t_j. We compute the arithmetic average of the asset prices, calculate the payoff for each path, discounting the payoff's back to present value and estimate the option price by averaging all the discounted payoff's across all simulations. Other equations used is for Variance, Standard Error and Confidence Interval:

- Variance:
$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(f(x_i) - \bar{f})^2$$
- Standard Error:
$$SE = \frac{\sigma}{\sqrt{N}}$$
- Confidence Interval (CI) at 0.05 significance level:
$$CI(0.05) = \bar{f} \pm Z \cdot SE$$

By using the values given in the task description, along with $N = 10000$ for the number of MC simulations and the stated equations, I obtained the following values:

- **Asian Call Option Price:** 1.8588
- **Variance of Discounted Payoffs:** 7.5619
- **95% Confidence Interval:** [1.8049, 1.9127]

2.2 B: Antithetic Sampling

As we saw from the result from the Crude Monte Carlo method, it has a high variance. To get a more "accurate" assessment of the "real" option price, we might attempt to minimize it and this is where the Antithetic sample method comes in. Antithetic sampling is a variance reduction technique used in MC to enhance efficiency and accuracy of our estimators. The core idea of this technique is to exploit the negative correlation between paired simulations to cancel out some of the random variability inherent in the MC estimation process.

By generating antithetic pairs $(Z, -Z)$, we ensure that one simulation of the payoff path tends to overestimate while its counterpart tends to underestimate the payoff path. This balance reduces the overall variance of the estimator. Let $\bar{C}$ be the estimator of the option price:

$$\bar{C} = \frac{1}{M_{\text{sim}}} \sum_{i=1}^{M_{\text{sim}}} \bar{C}_i,$$

where $\bar{C}_i = \frac{1}{2}(C_i + C'_i)$, and C_i and C'_i are the discounted payoffs of antithetic pairs.

The variance of the estimator $\bar{C}$ is:

$$\text{Var}(\bar{C}) = \frac{1}{M_{\text{sim}}^2} \sum_{i=1}^{M_{\text{sim}}} \text{Var}(\bar{C}_i) + \frac{2}{M_{\text{sim}}^2} \sum_{i<j} \text{Cov}(\bar{C}_i, \bar{C}_j).$$

Due to the negative correlation between C_i and C'_i, the covariance terms $\text{Cov}(C_i, C'_i)$ are negative, thereby reducing the overall variance of $\bar{C}$. By only flipping the sign, we will save the computational cost of producing completely new paths. We will use the same equations as above for the Variance, Standard Error and Confidence Interval, along with the same parameter values. I obtained the following values:

- **Asian Call Option Price:** 1.8427
- **Variance of Discounted Payoffs:** 2.01057
- **95% Confidence Interval:** [1.8149, 1.8705]

By comparison to the Crude MC, we see that the Variance of the discounted payoffs is way more than halved. In addition to a more narrower Confidence Interval for the Antithetic Sampling, which indicates higher precision in the estimate. These findings shows that the Antithetic sampling method have effectively reduced the variance of the estimator and increased the precision of the Asian call option price estimate.

2.3 C: Control Variate

Control Variates methods is also a variance reduction technique used in MC simulations to enhance the efficiency and accuracy of estimators. This technique uses a slightly different approach than the Antithetic sampling. The core idea is to use a control variate Y that is correlated with the payoff X and has a known expected value $E[Y]$. By adjusting the original estimator with the control variate, we can reduce the variance of the estimator without introducing bias. There are several techniques to implement a control variate, however there are two possible criteria:

- 1. The control variate Y needs to have high correlation with the target variable X
- 2. The control variate Y must have a known expected value $E[Y]$

Therefore I have chosen the arithmetic average $Y = \overline{S}_T$ itself, as its expectation value $E[\overline{S}_T]$ can be derived analytically under the GBM assumption:

$$E[S_T] = \frac{S_0}{M}\sum_{j=1}^{M} e^{rt_j} = \frac{S_0}{M}\sum_{j=1}^{M} e^{rj\Delta t}.$$

Given the control variate $Y = \bar{S}_T$ with known expectation $E[Y]$, the adjusted estimator for the option price θ is:

$$\theta = \frac{1}{N}\sum_{i=1}^{N} \left(X_i - c\left(Y_i - E[Y]\right)\right).$$

The optimal coefficient c that minimizes the variance of the adjusted estimator is given by:

$$c = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(Y)}.$$

To modify the original payoffs by incorporating the control variate, the adjusted payoff for each i is:

$$\text{Adjusted Payoff}_i = X_i - c\left(Y_i - E[Y]\right).$$

Next procedures will be to discount the adjusted payoffs and averaging them, in order to obtain the estimated option price for the Asian option.

We will use the same equations as above for the Variance, Standard Error and Confidence Interval, along with the same parameter values. I obtained the following values from the Control Variate technique:

- **Asian Call Option Price:** 1.8519
- **Variance of Discounted Payoffs:** 1.65099
- **95% Confidence Interval:** [1.8267, 1.8770]

Comparing these results with both the Antithetic sampling and Crude MC, we see that the Control variate techniques demonstrates superior variance reduction and precision in the estimated option price, outperforming both of the two other techniques. It has the lowest variance of all three methods, and has the narrowest confidence interval, resulting in a more reliable estimate of the Asian option price. The performance stems from its use of a highly correlated variable, which has reduced the variance more effectively than Antithetic sampling's negative correlation and sign-flipping.

2.4 D: Quasi Monte Carlo with Halton Sequence

Finally, we use the Quasi MC numerical approach to this sub-question. The nature of regular MC is the reason this is a wise decision. A pseudo sequence, or deterministic sequence that appears to be random—that is, they only approximate randomness—is what an MC variable is. Because MC approaches are deterministic by nature, it makes sense to concentrate on identifying the most effective deterministic sequences since they should further minimize variance.

Halton sequences are a popular choice for generating low-discrepancy sequences, which aim to uniformly cover the sampling space, thereby reducing variance and improving convergence rates.

Since I code in Python, there is a module from SciPy QMC where the Halton sequence generator is available with optional scrambling. Scrambling introduces randomness to reduce any patterns or correlations that may arise from the deterministic nature of the sequence, while preserving the low-discrepancy properties, which ultimately leads to faster convergence. However, for the purpose of this task I initiated this Scramble argument to False in order to maintain a purely deterministic sequence.

Halton sequences are defined for each dimension using distinct prime bases. For a d-dimensional space, the i-th element of the Halton sequence is:

$$u_i = (u_{i,1}, u_{i,2}, \dots, u_{i,d}),$$

where each $u_{i,j}$ is the i-th number in the Halton sequence for base p_j, with p_j being the j-th prime number.

To simulate asset price paths, the uniform Halton sequence samples u_i are transformed into standard normal random variables Z_i using the inverse cumulative distribution function (CDF) of the standard normal distribution:

$$Z_{i,j} = \Phi^{-1}(u_{i,j}),$$

where Φ^{-1} is the inverse CDF of the standard normal distribution, and $j = 1, 2, \dots, M$.

Starting from the initial asset price S_0, the asset price at each subsequent time step is computed using the Geometric Brownian Motion (GBM) discretization formula:

$$S_{i,j} = S_{i,j-1} \exp\left(\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}\, Z_{i,j}\right).$$

And then we do the same procedure as before, finding the arithmetic average of all the stock price paths, payoffs, discounted payoffs and estimating option price by the arithmetic average of all the discounted payoffs.

I obtained the price for the Asian option to be: 1.8178. The Halton sequence has no variance since it is deterministic. This also suggests that the estimate cannot be linked to a confidence interval. Nonetheless, we observe that the price is around the price of the other methods.

2.5 Call on Call (Compounded Option)

2.5.1 Price of Asian Option

In this last question, we were asked to find the price of a compound option, specifically a call-on-call option. In this case, the first option, C^a, is an arithmetic Asian call option on an American call option, C^c. The American call option is on a stock that pays dividends and has risk-neutral properties of GBM. For this question, the first procedure I thought of was to use a Binomial Method combined with a Monte Carlo simulation for the American call and the Asian option, however after doing some research I quickly found out the following:

- Computational Intensity: Incorporating the binomial method within a Monte Carlo simulation requires building a binomial tree for each simulated path and each monitoring date
- Trade-off: Even though this method is accurate, the binomial method may not be the most efficient when combined with Monte Carlo simulations for path-dependent options

This was when I found the Longstaff-Schwartz Method (LSM). This is not a method we have been taught in lesson, however I was curious and thought it would be a valuable experience. In short terms the Longstaff-Schwartz Method estimates the value of an American option by simulating multiple price paths, calculating continuation values through regression at each exercise point, and determining the optimal exercise strategy by comparing these continuation values to the immediate exercise payoffs. By moving backward through time, this approach identifies the optimal exercise policy when considering the early exercise feature of American options. I will in the following sections, try to explain step by step how the LSM can be implemented as a solution for this questions.

Equations

First thing for the LSM, is to estimate the value of the American option by simulating multiple price paths (I defined n = 10000), and we know its properties is a GBM:

$$dS_t = (r - \rho)S_t\,dt + \sigma S_t\,dz_t$$

$$S_{t+1} = S_t \exp\left(\left(r - \rho - \frac{1}{2}\sigma^2\right)(dt) + \sigma\sqrt{(dt)}z_t\right)$$

, where z_t is a standard normal distributed variable.

Matrix

Suggested by Longstaff and Schwartz (2001), these simulated paths can be organized into a matrix where each row represents a simulated paths and each column represents a time step. The matrix of simulated asset prices is structured as follows:

Time/Path	t_1	t_2	$\dots$	t_T
Path 1	$S_{1,1}$	$S_{1,2}$	$\dots$	$S_{1,T}$
Path 2	$S_{2,1}$	$S_{2,2}$	$\dots$	$S_{2,T}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Path N	$S_{N,1}$	$S_{N,2}$	$\dots$	$S_{N,T}$

Here:

- $S_{i,j}$ represents the simulated stock price at time t_j along path i.
- N is the total number of simulated paths.
- T is the total number of time steps.

Regression model

Starting from the penultimate time point (i.e., t_{T-1}) and working backward to the initial time t_1, we estimate the continuation value at each time step using regression techniques. The continuation value C_t is the expected discounted future payoff of the option if it is not exercised at time t.

The regression model used to estimate the continuation value is:

$$C_t = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Where:

- C_t is the continuation value at time t.
- α and β_n are regression coefficients.
- X_n are the chosen basis functions (functions of the state variables, e.g., the asset price S_t).

For simplicity, I have chosen a polynomial of degree 2 (quadratic polynomial) as the basis functions:

$$X_1 = S_t, \quad X_2 = S_t^2$$

Following Longstaff and Schwartz (2001), quadratic polynomials are often used due to their balance between simplicity and flexibility.

Optimal Exercise Strategy

At each time step and for each path, we compare the immediate exercise value with the estimated continuation value to decide whether to exercise the option or to continue holding it. The immediate exercise value is $E_t = \max(S_t - K,\, 0)$, and the continuation value from the regression model is C_t

The optimal exercise decision $d_{i,j}$ at time t_j along path i is determined by:

$$d_{i,j} = \begin{cases} 1, & \text{if } E_{i,j} > C_{i,j} \\ 0, & \text{if } E_{i,j} \leq C_{i,j} \end{cases}$$

Where:

- $d_{i,j} = 1$ indicates that the option is exercised at time t_j along path i.
- $d_{i,j} = 0$ indicates that we continue to hold the option.

This results in a decision matrix:

Time/Path	t_1	t_2	$\dots$	t_T
Path 1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,T}$
Path 2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,T}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Path N	$d_{N,1}$	$d_{N,2}$	$\dots$	$d_{N,T}$

Each element $d_{i,j}$ in the matrix takes the value 1 or 0, representing the exercise decision at each time step for each path.

Cash Flow Matrix

Based on the exercise decisions, we can construct a cash flow matrix that records the payoffs at each time step for each path:

$$\text{CF}_{i,j} = \begin{cases} E_{i,j}, & \text{if } d_{i,j} = 1 \\ 0, & \text{if } d_{i,j} = 0 \end{cases}$$

The cash flow matrix is then:

Time/Path	t_1	t_2	$\dots$	t_T
Path 1	$\text{CF}_{1,1}$	$\text{CF}_{1,2}$	$\dots$	$\text{CF}_{1,T}$
Path 2	$\text{CF}_{2,1}$	$\text{CF}_{2,2}$	$\dots$	$\text{CF}_{2,T}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Path N	$\text{CF}_{N,1}$	$\text{CF}_{N,2}$	$\dots$	$\text{CF}_{N,T}$

This matrix represents the cash flows (payoffs) received at each time step for each simulated path:

- If the option is exercised at time t_j along path i, $\text{CF}_{i,j} = E_{i,j}$.
- If the option is not exercised, $\text{CF}_{i,j} = 0$, except possibly at maturity t_T, where the intrinsic value is realized if not exercised earlier.

Once we have estimated the values of the American option $C_{\text{American},j}$ at specific monitoring dates t_j, we can compute the payoff of the arithmetic Asian option, which depends on the average of these American option values.

The payoff of the Asian option is as given from the task description:

$$C_{\text{Asian}} = \max\left(\frac{1}{5}\sum_{j=1}^{5} C_{\text{American},j} - K_{\text{Asian}},\ 0\right)$$

By using the parameter values given in the task description, $n = 10000$ for number of price paths and $n_{timesteps} = 100$, I found the price of the Asian option to be $C^a = 2.2646$

2.5.2 Delta and Gamma

Delta (Δ) and Gamma (Γ) are fundamental Greeks that measure the sensitivity of an option's price to changes in the underlying asset's price, and in this case it is our Asian option. It's definitions is the following:

$$\Delta = \frac{\partial V}{\partial S}, \Gamma = \frac{\partial^2 V}{\partial S^2},$$

Where V is our Asian option price and S is the underlying stock.

From L6 in Derivatives, the definitions of derivative of a function $V(S_0)$, representing the option price as a function of the underlying asset price S_0, is defined as:

$$V'(S_0) = \lim_{h \to 0} \frac{V(S_0 + h) - V(S_0)}{h}$$

This definition motivates the finite difference approximation for the first derivative when h is a small, finite value:

$$V'(S_0) \approx \frac{V(S_0 + h) - V(S_0)}{h}$$

Similarly, the second derivative of $V(S_0)$ is defined as:

$$V''(S_0) = \lim_{h \to 0} \frac{V'(S_0 + h) - V'(S_0)}{h}$$

Using the finite difference approximation for the first derivative from above, we obtain the finite difference approximation for the second derivative:

$$V''(S_0) \approx \frac{V'(S_0 + h) - V'(S_0)}{h} \approx \frac{\frac{V(S_0+2h)-V(S_0+h)}{h} - \frac{V(S_0+h)-V(S_0)}{h}}{h}$$

$$= \frac{V(S_0 + 2h) - 2V(S_0 + h) + V(S_0)}{h^2}$$

Using the same parameter values as above, including $h = 1.5$ (or $delta_S$ in the code), we obtained that $\Delta = 0.5543$ and $\Gamma = 0.0373$