Digital Option Pricing with Default Risk

Kaiwen Shen, Student ID 1009970239

November 10, 2023

1 Q1: Show that for all ρ, $\mathbb{P}\{\tau \leq T\} = p$

$$\begin{aligned}
\mathbb{P}\{\tau \leq T\} &= 1 - \mathbb{P}\{Y > a\} \\
&= 1 - 1 + \mathbb{P}\{Y \leq a\} \\
&= \mathbb{P}\{y \leq a\} \\
&= \Phi(\Phi^{-1}(p)) \\
&= p
\end{aligned}$$

Therefore $\mathbb{P}\{\tau \leq T\} = p \ \forall \ \rho$, Y distributed standard normal.

2 Q2: State and plot $\upsilon(\rho; p) = \upsilon(0; p)$

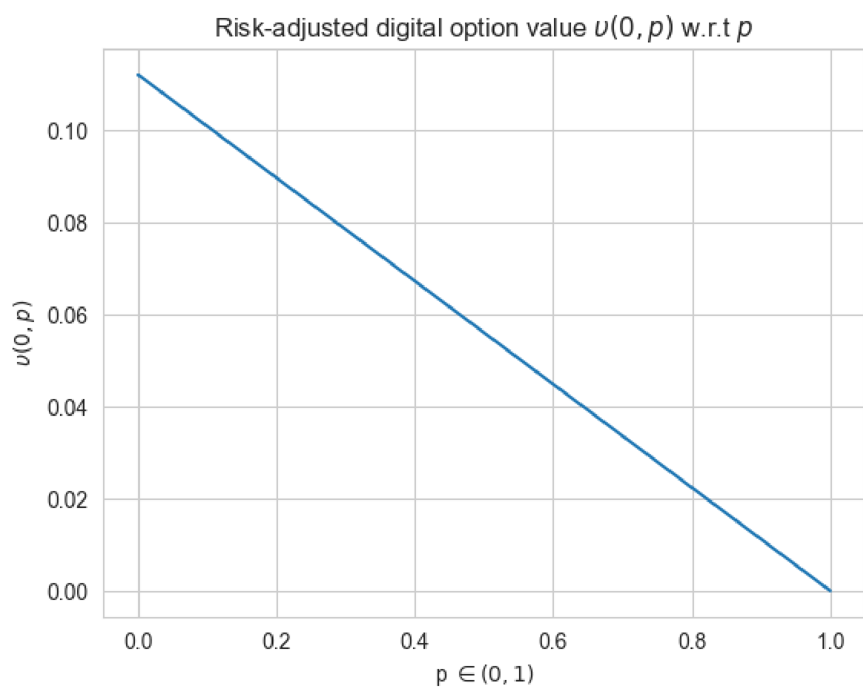

Figure 1: Q2: When $\rho = 0$, the value function υ with respect to the change in probability p

If $\rho = 0$, then Y defined as $Y = \rho Z + \sqrt{1-\rho^2} W = W$.

$$
\begin{aligned}
\upsilon(0,p) &= \mathbb{E}(1_{Y>a}1_{S\geq K}) \\
&= \mathbb{P}(W > \Phi^{-1}(p))\mathbb{P}(S \geq K) \\
&= (1-\Phi(\Phi^{-1}(p)))\mathbb{P}(s\exp(\sigma Z - \frac{1}{2}\sigma^2) \geq K) \\
&= (1-p)\left(1-\mathbb{P}\left(Z \leq \frac{\ln(\frac{K}{s}) - \frac{1}{2}\sigma^2}{\sigma}\right)\right) \\
&= (1-p)\left(1-\Phi\left(\frac{\ln(\frac{K}{s}) - \frac{1}{2}\sigma^2}{\sigma}\right)\right)
\end{aligned}
$$

for the context of all questions, we set $s = 1, \sigma = 0.2, K = 1.25$, then we have:

$$\upsilon(0,p) = (1-p)(1-\Phi(5\ln(1.25)+0.1))$$

We can plot $\upsilon(0,p)$as a function of p where $p \in (0,1)$, since all other parts are constant.

3 Q3: State and plot $\upsilon(\rho;p) = \upsilon(-1;p)$

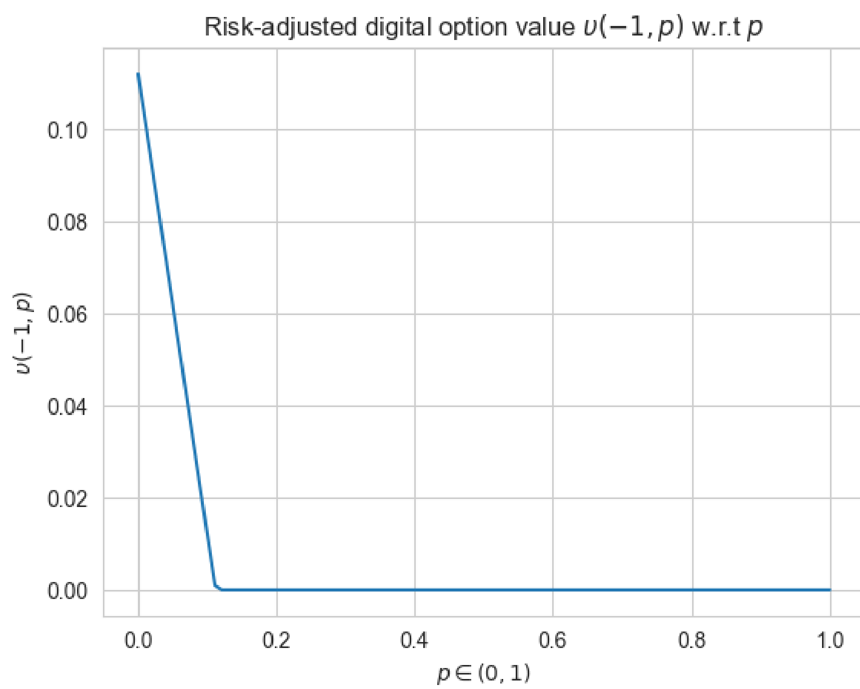

Figure 2: Q3: When $\rho = -1$, the value function υ with respect to the change in probability p

when $\rho = -1$, $Y = -Z$, the value function can be written as:

$$
\begin{aligned}
\upsilon(-1,p) &= \mathbb{E}\left(1_{-Z>\Phi^{-1}(p)}1_{s\exp\left(\sigma Z-\frac{1}{2}\sigma^2\right)}\right) \\
&= \mathbb{E}\left(1_{Z<-\Phi^{-1}(p)}1_{Z\geq\frac{\ln(\frac{K}{s}+\frac{1}{2}\sigma^2)}{\sigma}}\right) \\
&= \int_{\frac{\ln(\frac{K}{s}+\frac{1}{2}\sigma^2)}{\sigma}}^{-\Phi^{-1}(p)} \varphi(z)dz \\
&= \Phi(-\Phi^{-1}(p)) - \Phi\left(\frac{\ln(\frac{K}{s}+\frac{1}{2}\sigma^2)}{\sigma}\right) \\
&= 1-p-\Phi\left(\frac{\ln(\frac{K}{s}+\frac{1}{2}\sigma^2)}{\sigma}\right)
\end{aligned}
$$

Where, plugin the s, K, σ value, we have

$$\upsilon(-1,p) = 1-p-\Phi(5\ln(1.25)+0.1)$$

and since option value can only be positive or 0, we have

$$\upsilon(-1,p) = (1-p-\Phi(5\ln(1.25)+0.1))_+$$

4 Q4: "Brute Force" Monte Carlo

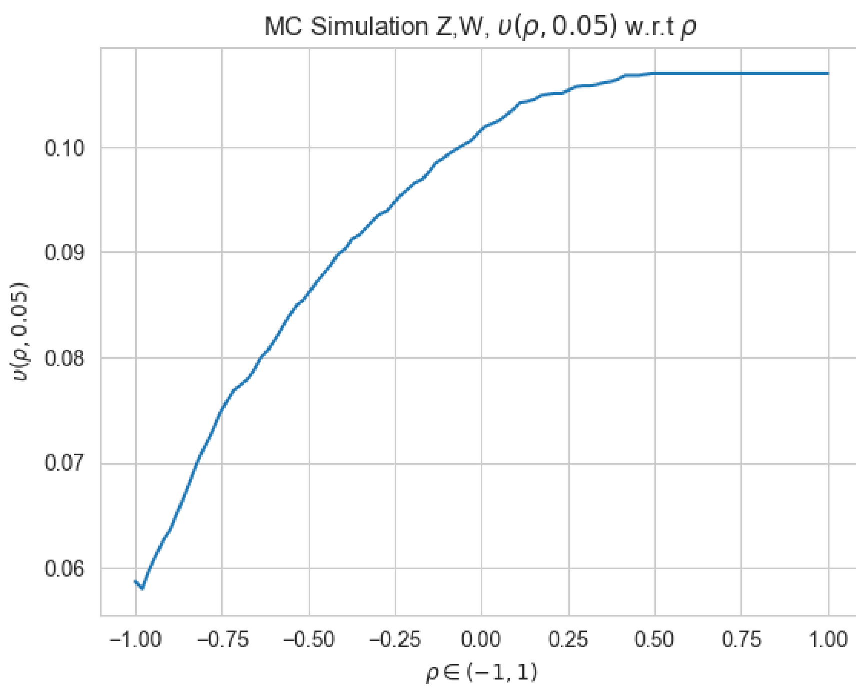

Figure 3: Q4: Monte Carlo Simulation to generate Z and W, risk-adjusted digital option value with respect to ρ

For $p = 0.05$, we generate two Gaussian Random variables, Z and W independent and identically distributed. Then we use Z to generate the underlying price, use Z and W to generate Y, then plot the risk-adjusted digital option value with respect to ρ

```python
def MC1_upsilon_p005(rho):
    a = stats.norm.ppf(0.05)
    S = s * np.exp(0.2 * Z - 0.5 * 0.2 ** 2)
    Y = rho * Z + np.sqrt(1 - rho ** 2) * W
    upsilon = np.where((S >= K) & (Y > a), 1, 0)
    return np.mean(upsilon)

N = 10_000
Z = stats.norm.rvs(size=N)
W = stats.norm.rvs(size=N)

plt.plot(np.linspace(-1, 1, 100), [MC1_upsilon_p005(rho)
    for rho in np.linspace(-1, 1, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(r"$\upsilon(\rho,0.05)$")
plt.title(r"MC Simulation Z,W, $\upsilon(\rho,0.05)$ w.r.t $\rho$")
plt.show()
```

5 Q5: Monte Carlo Partial Differential via First Principle

Under the same Monte Carlo simulation, we produce the partial differential of the risk-adjusted digital option with respect to the correlation, by using the approximation:

$$\frac{\partial \upsilon(\rho, 0.05)}{\partial \rho} \approx \frac{\upsilon(\rho + \varepsilon, 0.05) - \upsilon(\rho, 0.05)}{\varepsilon}$$

For a small value ε, where we picked $\varepsilon = 0.01$ for figure 4

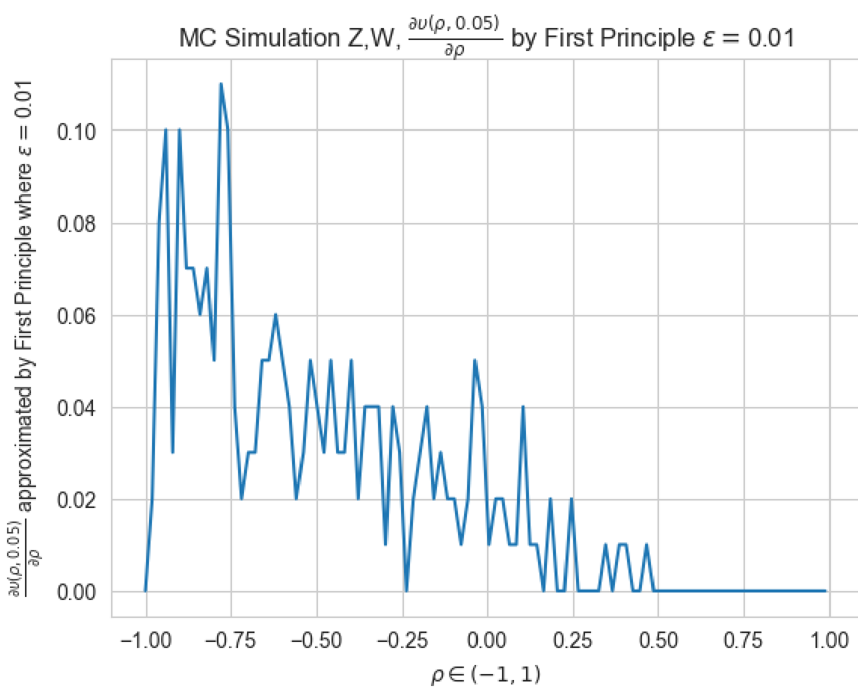

Figure 4: Q5: Monte Carlo Simulation to generate Z and W, calculate the partial differential of the risk-adjusted digital option value with respect to ρ, using the First Principle of Differentiation, excluding the case where $\rho = 1$ because when $\rho = 1$, $\rho + \varepsilon > 1$, which is outside of the correlation range.

```python
epsilon = 0.01
def MC1_corr_risk(rho, epsilon):
    return 1 / epsilon * (MC1_upsilon_p005(rho + epsilon)
    - MC1_upsilon_p005(rho))

plt.plot(np.linspace(-1, 0.99, 100), [MC1_corr_risk(rho, epsilon)
    for rho in np.linspace(-1, 0.99, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(
    r"$\frac{\partial\upsilon(\rho,0.05)}{\partial \rho}$
    approximated by First Principle where
    $\varepsilon=$" + f'{epsilon}')
plt.title(
    r"MC Simulation Z,W, $\frac{\partial\upsilon(\rho,0.05)}
    {\partial \rho}$ by First Principle
    $\varepsilon=$" + f'{epsilon}')
plt.show()
```

6 Q6: Derive Conditional Expectation

$$
\begin{aligned}
h(Z;p) &= \mathbb{E}\left(1_{\rho Z+\sqrt{1-\rho^2}W} 1_{S\geq k}\right) \\
&= 1_{S\geq K}\mathbb{E}\left(1_{\sqrt{1-\rho^2}W>a-\rho Z}\right) \\
&= 1_{S\geq K}\mathbb{E}\left(1_{W>\frac{a-\rho Z}{\sqrt{1-\rho^2}}}\right) \\
&= 1_{S\geq K}\Phi\left(\frac{\rho Z-a}{\sqrt{1-\rho^2}}\right)
\end{aligned}
$$

7 Q7: Derive Partial Differential for Conditional Expectation

$$
\begin{aligned}
\frac{\partial h(Z;p)}{\partial \rho} &= 1_{S\geq K}\Phi'\left(\frac{\rho Z-a}{\sqrt{1-\rho^2}}\right)\left(\frac{\rho Z-a}{\sqrt{1-\rho^2}}\right)' \\
&= 1_{S\geq K}\varphi\left(\frac{\rho Z-a}{\sqrt{1-\rho^2}}\right)\left(\frac{Z-a\rho}{(1-\rho^2)^{\frac{3}{2}}}\right)
\end{aligned}
$$

8 Q8: Monte Carlo with One Random Sampling Digital Option Value

We take advantage of the equation derived in the section 6 about conditional expectation, and reduce our random sampling by half:

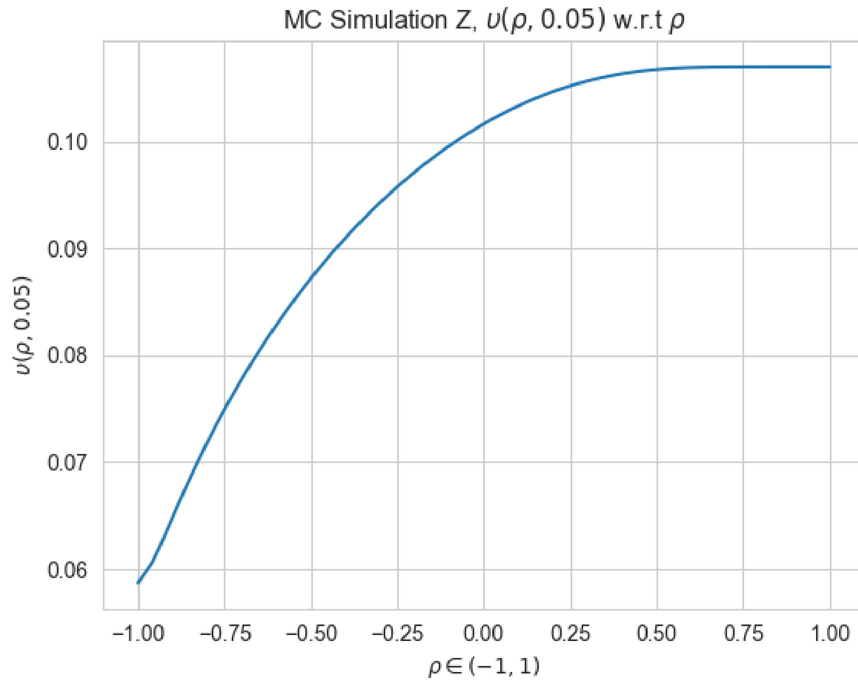

Figure 5: Q8: Monte Carlo Simulation to generate Z Only, risk-adjusted digital option value with respect to ρ

```
def MC2_upsilon_p005(rho):
    a = stats.norm.ppf(0.05)
    indicator = np.where(s * np.exp(0.2 * Z - 0.5 * 0.2 ** 2) > K, 1, 0)
    return np.mean(indicator *
    stats.norm.cdf((rho * Z - a) /
    np.sqrt(1 - rho ** 2)))

plt.plot(np.linspace(-1, 1, 100),
    [MC2_upsilon_p005(rho)
    for rho in np.linspace(-1, 1, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(r"$\upsilon(\rho,0.05)$")
plt.title(r"MC Simulation Z,
    $\upsilon(\rho,0.05)$ w.r.t $\rho$")
plt.show()
```

9 Q9: Monte Carlo with One Random Sampling Partial Differential

We take advantage of the equation derived in the section 7 about conditional expectation, and reduce our random sampling by half:

```
def MC2_corr_risk(rho):
    a = stats.norm.ppf(0.05)
    indicator = np.where(s * np.exp(0.2 * Z - 0.5 * 0.2 ** 2) > K, 1, 0)
    return np.mean(indicator *
        stats.norm.pdf((rho * Z - a) / np.sqrt(1 - rho ** 2))
        * (Z - a * rho) / (1 - rho ** 2) ** 1.5)

plt.plot(np.linspace(-1, 1, 100),
    [MC2_corr_risk(rho)
    for rho in np.linspace(-1, 1, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(r"$\frac{\partial\upsilon(\rho,0.05)}{\partial \rho}$")
plt.title(r"MC Simulation Z, $\frac{\partial\upsilon(\rho,0.05)}
{\partial \rho}$ w.r.t $\rho$")
plt.show()
```

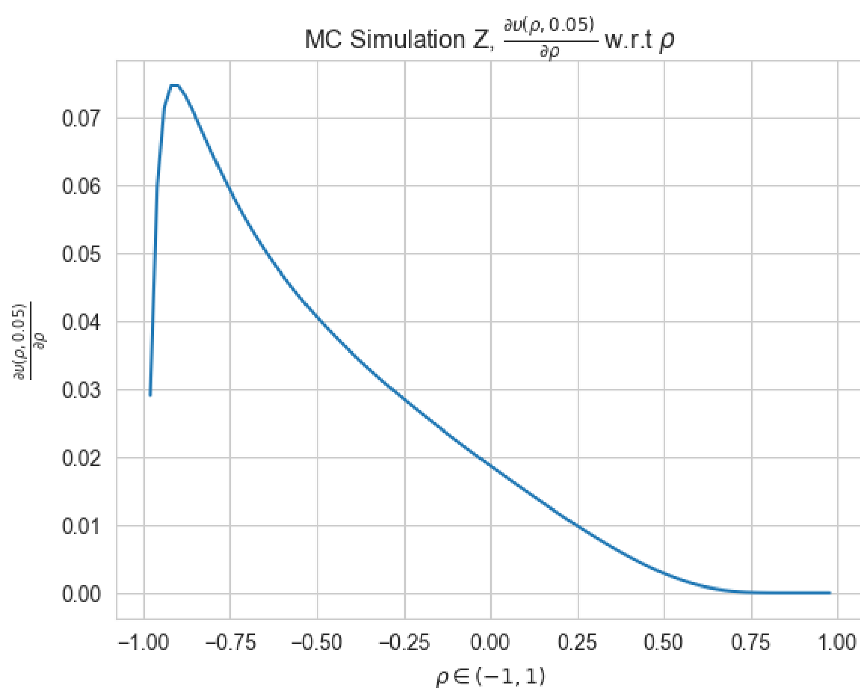

Figure 6: Q9: Monte Carlo Simulation to generate Z Only, calculate the partial differential of risk-adjusted value of digital option with respect to ρ

10 Q10: Numerical Integration for Option Risk-adjusted Value

```
M = 200
b = 5
def numerical_upsilon(rho):
    a = stats.norm.ppf(0.05)
    x = np.array([-b + 2 * b * i / M for i in range(1, M)])
    w = np.array([stats.norm.cdf(x[0])] +
        [stats.norm.cdf(x[i]) -
            stats.norm.cdf(x[i - 1])
            for i in range(1, len(x))])
    indicator = np.where(s * np.exp(0.2 * x - 0.5 * 0.2 ** 2) > K, 1, 0)
    return np.average(indicator *
        stats.norm.cdf((rho * x - a) /
        np.sqrt(1 - rho ** 2)),
        weights=w)
#%%
plt.plot(np.linspace(-1, 1, 100),
    [numerical_upsilon(rho)
    for rho in np.linspace(-1, 1, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(r"$\upsilon(\rho,0.05)$")
plt.title(r"Numerical Integration, $\upsilon(\rho,0.05)$ w.r.t $\rho$")
```

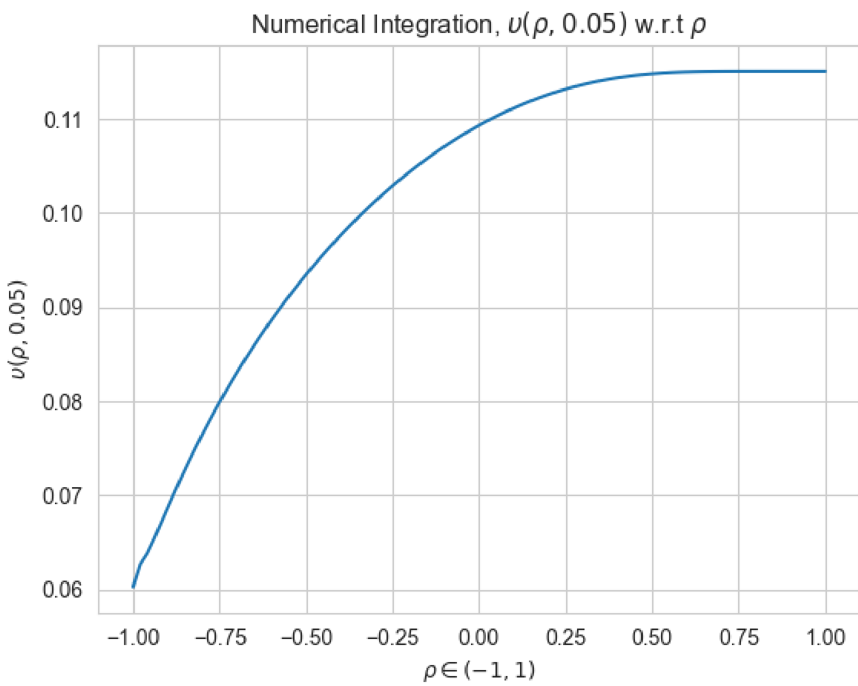

Figure 7: Q10: Using numerical integration to replace Monte Carlo simulation, calculate the risk-adjusted value of digital option

11 Q11: Numerical Integration for Partial Differential of Option Risk-adjusted value

```
def numerical_upsilon_partial(rho):
    a = stats.norm.ppf(0.05)
    x = np.array([-b + 2 * b * i / M for i in range(1, M)])
    w = np.array([stats.norm.cdf(x[0])] +
                 [stats.norm.cdf(x[i]) - stats.norm.cdf(x[i - 1])
                  for i in range(1, len(x))])
    indicator = np.where(s * np.exp(0.2 * x - 0.5 * 0.2 ** 2) > K, 1, 0)
    return np.average(
        indicator * stats.norm.pdf((rho * x - a) / np.sqrt(1 - rho ** 2))
        * (x - a * rho) / (1 - rho ** 2) ** 1.5,
        weights=w)

plt.plot(np.linspace(-1, 1, 100),
[numerical_upsilon_partial(rho)
for rho in np.linspace(-1, 1, 100)])
plt.xlabel(r"$\rho \in (-1,1)$")
plt.ylabel(r"$\frac{\partial\upsilon(\rho,0.05)}{\partial \rho}$")
plt.title(r"Numerical Integration,
$\frac{\partial\upsilon(\rho,0.05)}
{\partial \rho}$ w.r.t $\rho$")
plt.show()
```

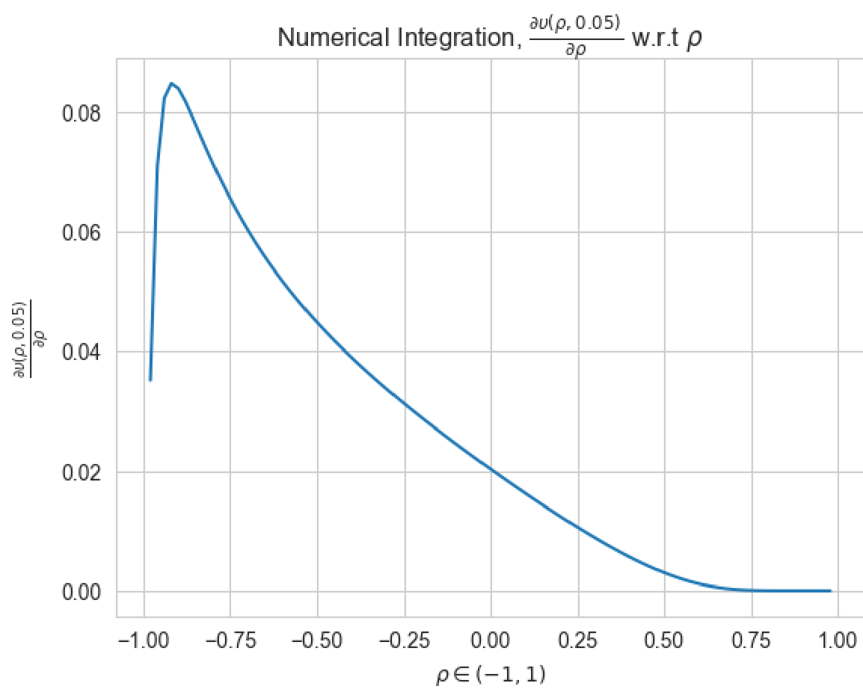

Figure 8: Q11: Using numerical integration to replace Monte Carlo simulation, calculate the partial differential of risk-adjusted value of digital option with respect to ρ