MATHGA 2792: Continuous Time Finance, hw4
Zepeng Xiao, zx1004

1

Solution:

Follow-up of Homework 1:

The **Appendix 1** of **Robert Merton's Paper** provides an example where the convexity property could be violated locally:

1. Suppose that S_0, S_T, K are the current stock price, the stock price at maturity and the strike price, respectively.
   Define the model of S_T as: $S_T = \hat{Z}S_0$ where $\hat{Z}$ is the common stock per dollar return.

   Suppose that the density of $\hat{Z}$ is $f_{\hat{Z}}(z)$, consider constant risk-free rate r, then the call option price is:
   $$C(S_0) = e^{-rT}\int_{\frac{K}{S_0}}^{\infty}(\hat{Z}S_0 - K)d\big[f_{\hat{Z}}(z;\, S_0)\big]$$

2. Define a parameter of the distribution $\lambda \in (0,1)$
   Define the model of $\hat{Z}$ as:
   $$\hat{Z} \equiv Z + U(S_0,\lambda) \equiv Z + \epsilon\lambda g(S_0)$$
   where $g(S) \in (0,1)$, $g^{'}(S) < 0\ \forall\, S,$ and ϵ, Z are independent random variables.

   Suppose that ϵ, Z follows uniform distribution: $\epsilon \sim \mathbb{U}(-1,1)$, $Z \sim \mathbb{U}(1,3)$
   Then the density of $U(S_0,\lambda) = \epsilon\lambda g(S_0)$, Z are:

   $f_U(u) = \dfrac{1}{2\lambda g(S_0)}$, $-\lambda g(S_0) < u < \lambda g(S_0)$; $f_Z(z) = \dfrac{1}{2}$, $1 < z < 3,$ and $f_U(u) = f_Z(z) = 0$ otherwise

   $\Rightarrow d\big[f_{\hat{Z}}(z)\big] = \dfrac{1}{2}d\hat{Z},$ when $1 + \lambda g(S_0) \le \hat{Z} \le 3 - \lambda g(S_0)$

3. Suppose that $K \approx 2S_0$, then:
   $$C(S_0,\lambda) = e^{-rT}\int_{\frac{K}{S_0}}^{\infty}(\hat{Z}S_0 - K)d\big[f_{\hat{Z}}(z;\, S_0,\, \lambda)\big] = e^{-rT}\Big[\frac{K^2}{4S_0} - \frac{3K}{2} + \frac{9S_0}{4} - \frac{\lambda^2 g^2(S_0)S_0^2}{12}\Big]$$
   $$\Rightarrow \frac{\partial^2 C(S_0,\lambda)}{\partial S_0^2} = e^{-rT}\Big[\frac{K^2}{2S_0^3} + \frac{\lambda^2}{6}(2g(S_0)g^{'}(S_0) + S_0(g^{'}(S))^2 + S_0 g(S_0)g^{''}(S_0))\Big]$$

4. Suppose that we choose g, S_0 such that: $g^{''}(S_0) < \dfrac{2g(S_0)g^{'}(S_0) + S_0(g^{'}(S))^2 + \frac{3K^2}{\lambda^2 S_0^3}}{S_0 g(S_0)}$
   then $\dfrac{\partial^2 C(S_0,\lambda)}{\partial S_0^2} = e^{-rT}\Big[\dfrac{K^2}{2S_0^3} + \dfrac{\lambda^2}{6}(2g(S_0)g^{'}(S_0) + S_0(g^{'}(S))^2 + S_0 g(S_0)g^{''}(S_0))\Big] < 0,$
   where convexity property violated

Therefore, the model described above, is an example where the price of a Call option is not always convex in the price of the underlying.

2

Solution:

American options and negative rates:

1)

1. **Section 2**:
   Section 2 is mainly about the intuition of rational option pricing, especially for the Call option pricing. It firstly states some assumptions that rational option pricing should be held:
   - The price of an American option should not be lower than the price of the corresponding European, since American has possibilities to make more profits than European (when early exercies could produce more profits).
   - Options should be priced such that it is neither a dominant or dominated security. For example, given two Calls with all other parameters the same except the maturity, then the Call with longer maturity should have lower price. Also,given two Calls with all other parameters the same except the strike, then the Call with lower strike should have higher price.

   Then in section 2, it is assumed that no dividends are made to the common stock (effects of dividends are described in **Section 3**), and current and future interest rates are positive (which is common in reality). By all assumptions showed above, section 2 states the following properties that rational option pricing may have:
   - The rational price for the European Call should be greater than $\max(0, S - \mathrm{PV}(K))$, where S is current stock price, and $\mathrm{PV}(K)$ is the present value of the strike price at maturity. This leads to two more properties: (1) The rational price for the American Call should be equal to the corresponding European when there are no dividends. (2) The rational price for the European Call at infinite maturity should be equal to the stock price.
   - The rational price for the American Call should be a convex function of the strike price, and is scale-invariant to some extent.

   After that, one more assumption is added that, the price of two American Calls should be the same if their parameters are the same and the return of their underlying common stock has the same distribution. It can be applied to state the properties for pricing the combination of calls, and the riskiness of stocks could be taken into consideration.
   - Given two American Calls with all other parameters the same, if one Call is the combination of the others such that the final returns on the stock are identically distributed, then the price of this Call should be lower than the price of its combination because of the higher riskiness.
   - Given two American Calls with all other parameters the same except the risk of the return on the stock, then the Call with larger risk should have lower price.
   - The rational price for the American Call should be homogeneous of degree one and be a convex function of the stock price, when the distribution of the return on the stock is independent of the level of stock price.

2. **Section 3**:
   Section 3 is mainly about how dividends and the change of strike price because of dividends K could influence rational option pricing. The price of the option will not be changed if there is an appropriate change of strike price after making dividends.

- An option is payout protected, which means that its value is invariant to the choice of payout policy, if the return on the stock is invariant to the fraction of the return represented by payouts, and if the price of the stock could be reduced by the value of payout(the number of shares that could be bought by E dollars, is increased $\frac{d}{S^x}$ percent, where d is the dollar amount of the payout, and S^x is the expayout price per share). This leads to one more property: If there is a finite number of changes in the exercise price of a payout-protected, perpetual Call, then it will not be exercised and its price will equal the common stock price.

3. **Section 4**:

   Section 4 is mainly about rational Put option pricing, sometimes Put option price could be uniquelly determined when its corresponding Call is determined, but it is not always.

   - The Put option price could be uniquelly determined when its corresponding Call is determined, if no dominant or dominated security exists, and the borrowing rate equals to the lending rate. This leads to two more properties: (1) The rational price for the European Put should be lower than the present value of the strike price at maturity. (2) The rational price for the European Put at infinite maturity should be zero.
   - The American Put option price could be uniquelly determined when its corresponding Call is determined, if it has positive possibilities to be exercised earlier. This will happen if there is a positive probability that the European Call price with same parameters, is lower than the discounted value of the strike price at some time before the maturity.

2)

Theorem 3 states that: The rational price for the European Call at infinite maturity should be equal to the stock price.

Theorem 3 have three interior assumptions: (1) No dividends, (2) Positive current and future interest rates, (3) no dominant or dominated security exists.

Theorem 3 holds since the European Call is exercised at infinite maturity, when the present value of strike price will be 0 under assumption (2), but the value of the common stock remains unchanged. So it can be regarded as get the common stock free.

Theorem 3 indicates that the option European Call Price is an non-decreasing function of the maturity when other parameters remain the same. The minimum possible price will be $\max(0, S-K)$ where S is current stock price, and K is the strike price, if the maturity is zero, and the maximum possible price will be S if the maturity is infinity, and cannot be greater than current stock price.

3)

When rates are negative but dividends are still positive. Holding the Call Option will take a cost since the present value of the strike price in the future will be greater than the strike price. The property of Call Options and Put Options, and the behavior of investors who hold Call Options and Put Options seems reversed.

1. Everyone will exercise the American Call immediately, hence the price of the American Call is greater than the corresponding European, and is equal to $\max(0, S-K)$ where S is current stock price, and K is the strike price.
2. The price of the European Call is a decreasing function of the maturity when other parameters remain the same, decrease from $\max(0, S-K)$ if the maturity is zero, to zero if the maturity is infinity.

3. Investors tend to not exercise American Put before maturity, since holding Put longer could increase the present value of the strike price, so payoff could be increased. Also, holding the common Stock could receive dividends. As a result, the price of the American Put is equal to the price of the European Put, and the price of the European Put could go to infinity if the maturity is infinity.

3

Solution:

Calibration of Stochastic Local Volatility models:

We are given that: $\frac{dS}{S} = l(S,t)\alpha_t dW,\ d\alpha_t = udt + vdZ,\ \langle dW, dZ\rangle = \rho dt$

1)

Denote C as the Call option prices.

We have derived the Dupire formula about local volatility in **Homework2, Exercise3** that:

$$\sigma^2_{\text{LV}}(T,K) = \frac{\frac{\partial C}{\partial T} + r_T K \frac{\partial C}{\partial K}}{\frac{1}{2}K^2 \frac{\partial^2 C}{\partial K^2}}$$

In this problem, we know that the risk-free rate $r = 0 \Rightarrow r_T = 0$

Therefore, in this problem,

$$v_{\text{loc}}(S,t) = \frac{2\frac{\partial C(S,t)}{\partial t}}{S^2 \frac{\partial^2 C(S,t)}{\partial S^2}} = \frac{2\partial_t C(S,t)}{S^2 \partial_{SS} C(S,t)}$$

2)

In this SLV model: $\frac{dS}{S} = l(S,t)\alpha_t dW,\ v_t = l^2(S,t)\alpha_t^2$

To calibrate this SLV model on the market: $\mathbb{E}[(l(S,t)\alpha_t)^2 \mid S_t = S] = \mathbb{E}[l^2(S,t)\alpha_t^2 \mid S_t = S] = v_{\text{loc}}(S,t)$

Therefore,

$$v_{\text{loc}}(S,t) = \mathbb{E}[v_t \mid S_t = S]$$

3)

We know from **2)** that: $v_{\text{loc}}(S,t) = \mathbb{E}[v_t \mid S_t = S] = \mathbb{E}[l^2(S,t)\alpha_t^2 \mid S_t = S] = l^2(S,t)\mathbb{E}[\alpha_t^2 \mid S_t = S]$

Suppose that $p(S,\alpha,t)$ be the joint density probability of (S_t, α_t), then by 2-D Fokker-Planck:

$$\frac{\partial p}{\partial t} + \frac{\partial (up)}{\partial \alpha} - \frac{1}{2}\frac{\partial^2 (l^2(S,t)\alpha^2 S^2 p)}{\partial S^2} - \frac{1}{2}\frac{\partial^2 (v^2 p)}{\partial \alpha^2} - \frac{\partial^2 (\rho l(S,t)\alpha S v p)}{\partial S \partial \alpha} = 0$$

Note that $\mathbb{E}[\alpha_t^2 \mid S_t = S] = \frac{\int_0^\infty \alpha_t^2 p(t,S,\alpha)d\alpha}{\int_0^\infty p(t,S,\alpha)d\alpha}$

Therefore, to ensure calibration, $l(S,t)$ should satisfy that:

$$l^2(S,t) = v_{\text{loc}}(S,t)\frac{\int_0^\infty p(t,S,\alpha)d\alpha}{\int_0^\infty \alpha_t^2 p(t,S,\alpha)d\alpha}$$

where $p(t,S,\alpha)$ should satisfies that: $p(0,S,\alpha) = 1$ if $S = S_0, \alpha = \alpha_0$, and 0 otherwise, and:

$$\frac{\partial p}{\partial t} + \frac{\partial (up)}{\partial \alpha} - \frac{1}{2}\frac{\partial^2 (l^2(S,t)\alpha^2 S^2 p)}{\partial S^2} - \frac{1}{2}\frac{\partial^2 (v^2 p)}{\partial \alpha^2} - \frac{\partial^2 (\rho l(S,t)\alpha S v p)}{\partial S \partial \alpha} = 0$$

4)

Suppose that we konw S_0, α_0 at time $t = 0$, then we could choose S_l, S_u, $\alpha_l = 0$, α_u, $t_l = 0$, $t_u = T$, where $S_0 \in (S_l, S_u), \alpha_0 \in (0, \alpha_u)$ and construct the grid points between them.

i.e, we could numerically solve the leverage function $l(S,t)$ when $S \in (S_l, S_u), t \in (0, T)$ by choose an arbitrary large α_u that can be applied to numerically compute $\dfrac{\int_0^\infty p(t,S,\alpha)d\alpha}{\int_0^\infty \alpha_t^2 p(t,S,\alpha)d\alpha}$

Also, apply the explicit finite difference scheme to the PDE showed in **3)**, we have that:

$$\begin{aligned}
&\frac{p_{i,j}^{k+1} - p_{i,j}^k}{\Delta t} + \frac{up_{i,j+1} - up_{i,j-1}}{\Delta\alpha} - \frac{1}{2}\frac{l^2(S_{i+1},t_k)\alpha_j^2 S_{i+1}^2 p_{i+1,j}^k + l^2(S_{i-1},t_k)\alpha_j^2 S_{i-1}^2 p_{i-1,j}^k - 2l^2(S_i,t_k)\alpha_j^2 S_i^2 p_{i,j}^k}{\Delta S^2} \\
&- \frac{1}{2}\frac{\rho l(S_{i+1},t_k)\alpha_{j+1} S_{i+1} v p_{i+1,j+1}^k + \rho l(S_{i-1},t_k)\alpha_{j-1} S_{i-1} v p_{i-1,j-1}^k + 2\rho l(S_i,t_k)\alpha_j S_i v p_{i,j}^k}{2\Delta S\Delta\alpha} \\
&- \frac{1}{2}\frac{-\rho l(S_{i-1},t_k)\alpha_j S_{i-1} v p_{i-1,j}^k - \rho l(S_{i+1},t_k)\alpha_j S_{i+1} v p_{i+1,j}^k - \rho l(S_i,t_k)\alpha_{j-1} S_i v p_{i,j-1}^k - \rho l(S_i,t_k)\alpha_{j+1} S_i v p_{i,j+1}^k}{2\Delta S\Delta\alpha} \\
&- \frac{1}{2}\frac{v^2 p_{i,j+1}^k + v^2 p_{i,j-1}^k - 2v^2 p_{i,j}^k}{\Delta\alpha^2} = 0
\end{aligned} \tag{1}$$

Suppose that S_l, S_u, α_u, T is given, then the numerical scheme can be described as follows:

1. Construct grid points:
   - Let $\Delta S = \dfrac{S_u - S_l}{N_S}, \Delta\alpha = \dfrac{\alpha_u - \alpha_l}{N_\alpha} = \dfrac{\alpha_u}{N_\alpha}, \Delta t = \dfrac{t_u - t_l}{N_t} = \dfrac{T}{N_t}$
   - Define $S_i = S_l + i\Delta S$, $i = 0, \cdots, N_S$; $\alpha_j = j\Delta\alpha$, $j = 0, \cdots, N_\alpha$; $t_k = k\Delta t$, $k = 0, \cdots, N_t$
   - Let $p_{i,j}^k = p(t_k, S_i, \alpha_j)$ be the joint density probability of (S_{t_k}, α_{t_k})
2. Compute $l(S,0)$:
   - We know that $p(0,S,\alpha) = 1$ if $S = S_0, \alpha = \alpha_0$, and 0 otherwise, which means that $p_{i,j}^0$ is known $\forall\, i = 0, \cdots, N_S$, $j = 0, \cdots, N_\alpha$
   - Compute $\dfrac{\int_0^\infty p(t,S,\alpha)d\alpha}{\int_0^\infty \alpha_t^2 p(t,S,\alpha)d\alpha}$ by computig $\dfrac{\sum_{j=0}^{N_\alpha} p_{i,j}^0 \Delta\alpha}{\sum_{j=0}^{N_\alpha} \alpha_j^2 p_{i,j}^0 \Delta\alpha}$
   - Then $l^2(S_i,0) = v_{\text{loc}}(S_i,t)\dfrac{\sum_{j=0}^{N_\alpha} p_{i,j}^0 \Delta\alpha}{\sum_{j=0}^{N_\alpha} \alpha_j^2 p_{i,j}^0 \Delta\alpha} \quad \forall\, i = 0, \cdots, N_S$
3. Compute $p_{i,j}^1$:
   - From the discretized PDE showed above, we know that $p_{i,j}^1$ depends on: $l(S_{i-1},0)$, $l(S_i,0)$, $l(S_{i+1},0)$ and $p_{i,j}^0$, $p_{i-1,j-1}^0$, $p_{i+1,j+1}^0$, $p_{i-1,j}^0$, $p_{i+1,j}^0$, $p_{i,j-1}^0$, $p_{i,j+1}^0$
   - $p_{i,j}^0$ is known $\forall\, i = 0, \cdots, N_S$, $j = 0, \cdots, N_\alpha$ and we have computed $l(S_i,0)$ $\forall\, i = 0, \cdots, N_S$ in step 2
   - Therefore, $p_{i,j}^1$ could be computed by the discretized PDE $\forall\, i = 0, \cdots, N_S$, $j = 0, \cdots, N_\alpha$
4. Repeated step 2 and 3 until $t_{N_t} = T$ is reached.
   Finally, we could know $l(S_i,t_k)$ $\forall\, i = 0, \cdots, N_S$, $k = 0, \cdots, N_t$
   and $p_{i,j}^k$ $\forall\, i = 0, \cdots, N_S$, $j = 0, \cdots, N_\alpha$ $k = 0, \cdots, N_t$

5) The code and result for computing $l(S,t)$ given a specific case is showed below:

In [1]:
```
import numpy as np
#import warnings
#warnings.filterwarnings("ignore")
```

Case description:

$u = 0, v = 0.5\alpha, \rho = 0, S_0 = 100, \alpha_0 = 1$

The initial implied volatility surface is flat at 20% for all strikes and maturities $\Rightarrow v_{\text{loc}}(S,t) \equiv 20\%$

$u = 0, \rho = 0 \Rightarrow$ the discretized PDE could be simplifed as:

$$\frac{p_{i,j}^{k+1} - p_{i,j}^{k}}{\Delta t} - \frac{1}{2}\frac{l^2(S_{i+1},t_k)\alpha_j^2 S_{i+1}^2 p_{i+1,j}^{k} + l^2(S_{i-1},t_k)\alpha_j^2 S_{i-1}^2 p_{i-1,j}^{k} - 2l^2(S_i,t_k)\alpha_j^2 S_i^2 p_{i,j}^{k}}{\Delta S^2} - \frac{1}{2}\frac{v^2 p_{i,j+1}^{k} + v^2 p_{i,j-1}^{k} - 2v^2 p_{i,j}^{k}}{\Delta \alpha^2} = 0$$

$\Rightarrow$

$$p_{i,j}^{k+1} = p_{i,j}^{k} + \Delta t\Big[\frac{1}{2}\frac{l^2(S_{i+1},t_k)\alpha_j^2 S_{i+1}^2 p_{i+1,j}^{k} + l^2(S_{i-1},t_k)\alpha_j^2 S_{i-1}^2 p_{i-1,j}^{k} - 2l^2(S_i,t_k)\alpha_j^2 S_i^2 p_{i,j}^{k}}{\Delta S^2} + \frac{1}{2}\frac{v^2 p_{i,j+1}^{k} + v^2 p_{i,j-1}^{k} - 2v^2 p_{i,j}^{k}}{\Delta \alpha^2}\Big]$$

We choose $S_l = 50,\ S_u = 150,\ N_s = 100,\ \alpha_l = 0,\ \alpha_u = 2,\ N_\alpha = 20,\ t_l = 0,\ t_u = 1,\ N_t = 200$ to solve $l(S,t)$:

In [2]:

```
# Define parameters
vLoc = 0.2 ** 2
S0, alpha0 = 100., 1.
Sl, Su, Ns = 50., 150., 100
alphaL, alphaU, Nalpha = 0., 2., 50
tl, tu, Nt = 0., 1., 200
# v is a function of alpha
def vAlpha(alpha):
    return 0.5*alpha

# compute discretized step
dS = (Su - Sl)/Ns
dAlpha = (alphaU - alphaL)/Nalpha
dt = (tu - tl)/Nt

# initialize the matrix to store l(S,t) for all Si and tk
leverage = np.zeros((Ns + 1, Nt + 1))
# initiliaze the matrix to store p(t, S, alpha) forall Si, alpha_j at time k and k + 1
density_k = np.zeros((Ns + 1, Nalpha + 1))
density_kp = np.zeros((Ns + 1, Nalpha + 1))

# step 2, compute l(S, 0)
# define the density at time 0:
density_k[int((S0 - Sl)/dS), int((alpha0 - alphaL)/dAlpha)] = 1
for i in range(Ns + 1):
    # compute the expectation of alpha^2 when S = Si, t = 0
    sum1, sum2 = 0, 0
    for j in range(Nalpha + 1):
        sum1 += density_k[i, j] * dAlpha
        sum2 += (j * dAlpha) ** 2 * density_k[i, j] * dAlpha
    if (sum2 == 0):
        reci_E = 0
    else:
        reci_E = sum1 / sum2

    # compute l(Si, 0)
    leverage[i, 0] = (vLoc * reci_E) ** 0.5

# repeat step 2 and step 3
# firstly compute density at time tk, then compute l(S, tk)
for k in range(1, Nt):
```

```
# compute the density by discretized PDE
density_kp = np.zeros((Ns + 1, Nalpha + 1))
for i in range(Ns + 1):
    if (i == 0):
        for j in range(Nalpha + 1):
            if (j == 0):
                term1 = 0.5*(vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            elif (j < Nalpha):
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] +
                             vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            else:
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
    elif (i < Ns):
        for j in range(Nalpha + 1):
            if (j == 0):
                term1 = 0.5*(vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] +
                                             leverage[i-1, k - 1]**2 * (Sl+(i-1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            elif (j < Nalpha):
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] +
                             vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] +
                                             leverage[i-1, k - 1]**2 * (Sl+(i-1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            else:
```

```
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i+1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i + 1, j] +
                                             leverage[i-1, k - 1]**2 * (Sl+(i-1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
    else:
        for j in range(Nalpha + 1):
            if (j == 0):
                term1 = 0.5*(vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i - 1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1 - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            elif (j < Nalpha):
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] +
                             vAlpha((j + 1)*dAlpha)**2 * density_k[i, j + 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i-1, k - 1]**2 * (Sl+(i-1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)
            else:
                term1 = 0.5*(vAlpha((j - 1)*dAlpha)**2 * density_k[i, j - 1] -
                             2 * vAlpha(j*dAlpha)**2 * density_k[i, j]) / (dAlpha) ** 2
                term2 = 0.5*(j*dAlpha)**2 * (leverage[i - 1, k - 1]**2 * (Sl+(i+1)*dS)**2 * density_k[i - 1, j] -
                                             2*leverage[i, k - 1]**2 * (Sl+i*dS)**2 * density_k[i, j]) / (dS)**2
                density_kp[i, j] = max(density_k[i, j] + dt*(term1 + term2), 0)

# update the density
density_k = density_kp.copy()

# compute 1(S, tk)
for i in range(Ns + 1):
    # compute the expectation of alpha^2 when S = Si, t = tk
    sum1, sum2 = 0, 0
    for j in range(Nalpha + 1):
        sum1 += density_k[i, j] * dAlpha
        sum2 += (j * dAlpha) ** 2 * density_k[i, j] * dAlpha
    if (sum2 == 0):
        reci_E = 0
    else:
        reci_E = sum1 / sum2
```

```
        # compute l(Si, tk)
        leverage[i, k] = (vLoc * reci_E) ** 0.5
```

In [3]:
```
# plot the result

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline


Slin = np.linspace(Sl, Su, num = Ns + 1, endpoint = True)
tlin = np.linspace(tl, tu, num = Nt + 1, endpoint = True)
Tgrid, Sgrid = np.meshgrid(tlin, Slin)
ax = plt.axes(projection='3d')
ax.plot_surface(Tgrid, Sgrid, leverage, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.set_title('Surface of leverage function l(S,t)');
```