MFE Maths Assignment 4 — Naman Singhal

Q1

$$P(S,t) = e^{\left((n-1)\gamma + \frac{n(n-1)\sigma^2}{2}\right)(T-t)} S(t)^n$$

$$P_t = -\left((n-1)\gamma + \frac{n(n-1)\sigma^2}{2}\right) P$$

$$P_s = \frac{n}{S} P$$

$$P_{ss} = \frac{n(n-1)}{S^2} P$$

⇒ Using Black-Scholes

$$P_t + \gamma S P_s + \frac{\sigma^2}{2} S^2 P_{ss} - \gamma P$$

$$= -(n-1)\gamma P - \frac{n(n-1)}{2}\sigma^2 P + \gamma n P + \frac{\sigma^2}{2} n(n-1) P - \gamma P$$

$$= -(n-1)\gamma P + (n-1)\gamma P$$

$$= 0$$

∴ This satisfies black scholes

$$P(S, t=T) = S^n = f(S)$$

$$P(0,t) = 0$$

} Satisfies Boundary Conditions

Q2 (a) In reflected log coordinates:

$$V_s = \left(r - \frac{\sigma^2}{2}\right) V_x + \frac{\sigma^2}{2} V_{xx} - rV$$ — PDE

where $x = \log(S)$, $s = T - t$

Payout is $P(s,T) = (S-50)(100-S)$ if $50 \le S \le 100$ & $0 \le t \le 1$, and 0 otherwise

$$V(x,0) = (e^x - 50)(100 - e^x)$$ — Initial and Boundary conditions

$$V(\ln(50), s) = 0$$
$$V(\ln(100), s) = 0$$

$r = 0.05$

$\sigma = 0.4$

PDE $\Rightarrow$ $V_s = -0.03 V_x + 0.08 V_{xx} - 0.05 V$

(b) Using operators D_{+s}, D_{0x}, $D_{+x} D_{-x}$ the PDE becomes:

$$\frac{w_n^{m+1} - w_n^m}{\Delta s} = -0.03\left(\frac{w_{n+1}^m - w_{n-1}^m}{2\Delta x}\right) + 0.08\left(\frac{w_{n+1}^m - 2w_n^m + w_{n-1}^m}{\Delta x^2}\right) - 0.05\, w_n^m$$

$$w_n^{m+1} = \underbrace{\Delta s\left(\frac{-0.03}{2\Delta x} + \frac{0.08}{\Delta x^2}\right)}_{A} w_{n+1}^m + \underbrace{\left(1 - \Delta s\left(\frac{0.16}{\Delta x^2} + 0.05\right)\right)}_{B} w_n^m + \underbrace{\Delta s\left(\frac{0.03}{2\Delta x} + \frac{0.08}{\Delta x^2}\right)}_{C} w_{n-1}^m$$

$$w_n^{m+1} = A\, w_{n+1}^m + B\, w_n^m + C\, w_{n-1}^m$$

Question 2

(a) and first half of (b) are in the PDF

In [1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [2]:
```
# (b)
def finite_difference_solver(dx, ds):
    A = ds * (-0.03/(2*dx) + 0.08/(dx*dx))
    B = 1 - ds * (0.16/(dx*dx) + 0.05)
    C = ds * (0.03/(2*dx) + 0.08/(dx*dx))

    N = int((math.log(100) - math.log(50))/dx + 1)
    M = int(1/ds + 1)
    grid = np.zeros((N, M))

    # initial condition
    x_vals = math.log(50) + np.arange(0, N)*dx
    s_vals = np.arange(0, M)*ds
    grid[:, 0] = (np.exp(x_vals) - 50) * (100 - np.exp(x_vals))
    grid[0, :] = 0
    grid[-1, :] = 0
    # grid[:, 0]

    # solution:
    for i in range(M-1):
        grid[1:-1, i+1] = A*grid[2:, i] + B*grid[1:-1, i] + C*grid[0:-2, i]

    return x_vals, s_vals, grid
```

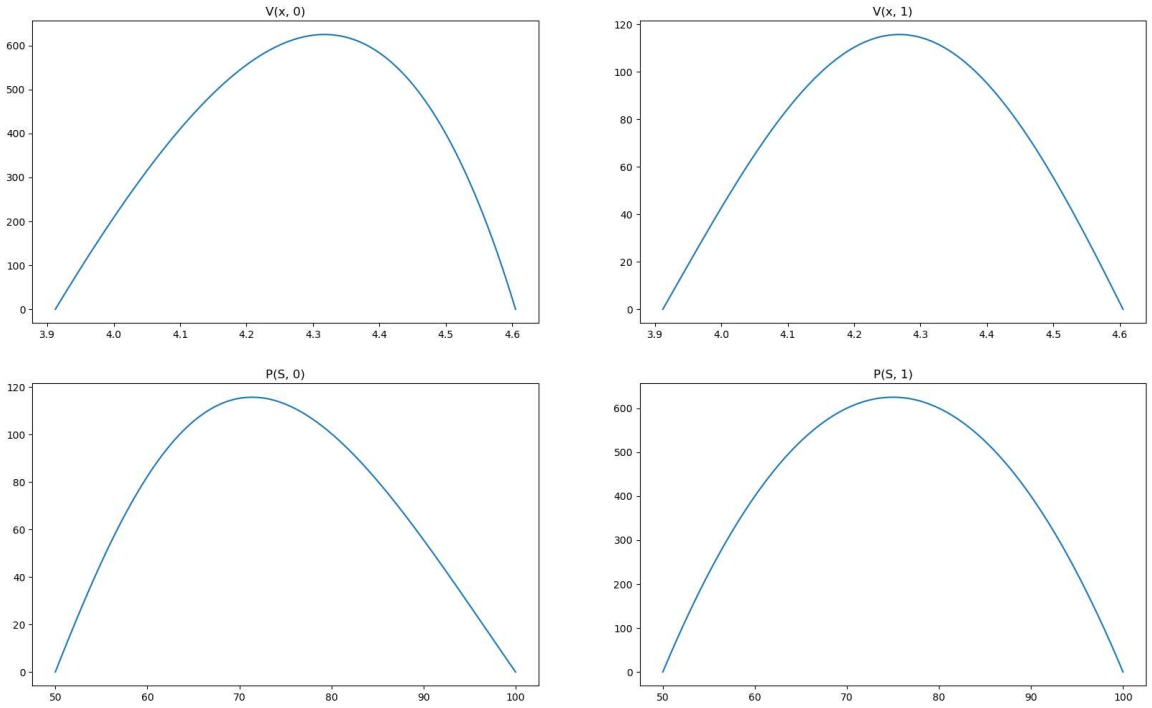
In [3]:
```
# (c)
k = 2
# Assuming delta x = 100 points means we should have 100 points between the limi
dx = (math.log(100) - math.log(50))/100
ds = dx*dx*k
print(k, dx, ds)
x_vals, s_vals, solution_grid = finite_difference_solver(dx, ds)
```

```
2 0.0069314718055994585 9.609060278364044e-05
```

In [4]:
```
solution_grid
```

Out[4]:
```
array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [17.26792595, 17.24079609, 17.21789868, ...,  3.40297642,
         3.40242428,  3.40187223],
       [34.41153179, 34.38368786, 34.35584278, ...,  6.81145668,
         6.81035151,  6.80924651],
       ...,
       [66.94109328, 66.76884075, 66.59658414, ...,  7.74017091,
         7.73890387,  7.73763704],
       [34.06038662, 33.88539419, 33.73880396, ...,  3.87704033,
         3.87640567,  3.87577111],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ]])
```

In [5]:
```
fig, ax = plt.subplots(2, 2, figsize=(20,12))
ax[0, 0].plot(x_vals, solution_grid[:, 0], label=f"V(x, 0)")
ax[0, 0].set_title(f"V(x, 0)")

ax[0, 1].plot(x_vals, solution_grid[:, -1])
ax[0, 1].set_title(f"V(x, 1)")

ax[1, 0].plot(np.exp(x_vals), solution_grid[:, -1])
ax[1, 0].set_title(f"P(S, 0)")

ax[1, 1].plot(np.exp(x_vals), solution_grid[:, 0])
ax[1, 1].set_title(f"P(S, 1)")

fig.show()
```



In [6]:
```
# t = T
np.exp(x_vals[np.argmax(solution_grid[:, 0])])
```

Out[6]:
```
74.74246243174692
```

In [7]:
```
# t = 0
np.exp(x_vals[np.argmax(solution_grid[:, -1])])
```

Out[7]: 71.20250977985357

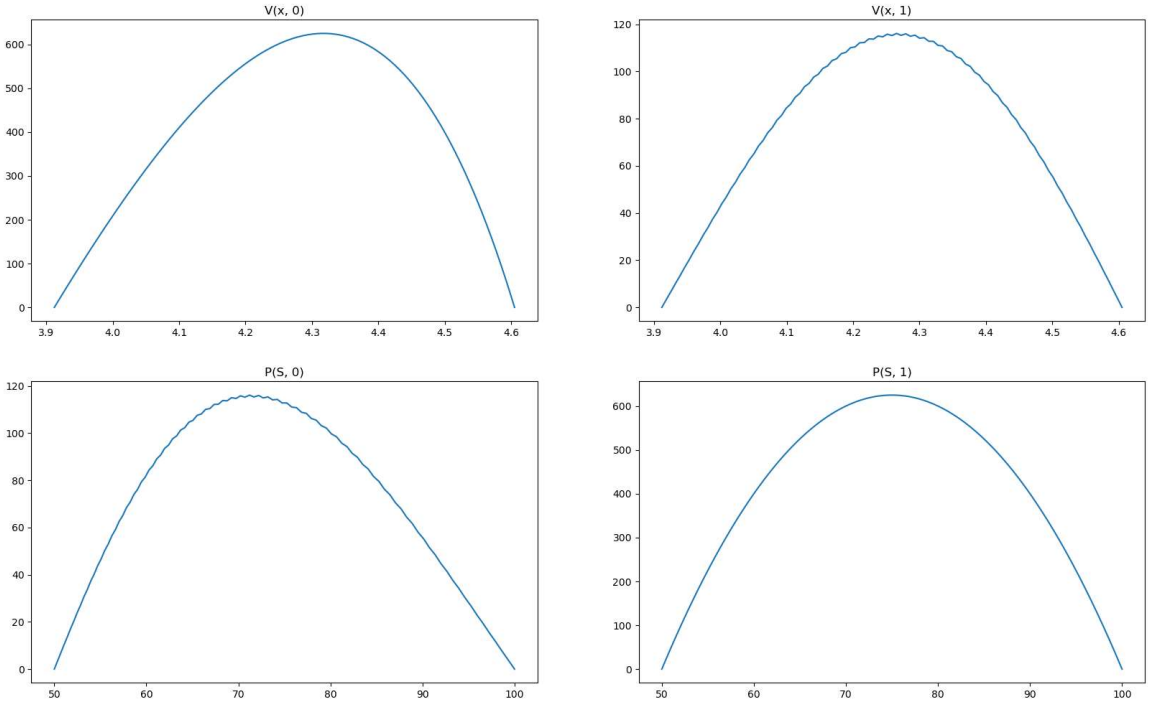
At t=T, the maximum value is at S = 75, while the maximum value is at S = 71.45 at t=0.

This is because of the risk free interest rate of 0.05 per year. For the expected stock price to be 75 at t=T, it would be expected to be at 75/(1+0.05) at t=0, assuming zero volatility in the price of the stock. Accordingly, the maximum value would be at 71.4286 at t=0.

In [8]:
```
# (d)

k = 6.26
# Assuming delta x = 100 points means we should have 100 points between the limi
dx = (math.log(100) - math.log(50))/100
ds = dx*dx*k
print(k, dx, ds)
x_vals, s_vals, solution_grid = finite_difference_solver(dx, ds)
fig, ax = plt.subplots(2, 2, figsize=(20,12))
ax[0, 0].plot(x_vals, solution_grid[:, 0], label=f"V(x, 0)")
ax[0, 0].set_title(f"V(x, 0)")

ax[0, 1].plot(x_vals, solution_grid[:, -1])
ax[0, 1].set_title(f"V(x, 1)")

ax[1, 0].plot(np.exp(x_vals), solution_grid[:, -1])
ax[1, 0].set_title(f"P(S, 0)")

ax[1, 1].plot(np.exp(x_vals), solution_grid[:, 0])
ax[1, 1].set_title(f"P(S, 1)")

fig.show()
```

Starts to destabilize at 6.26

```
6.26 0.0069314718055994585 0.00030076358671279457
```



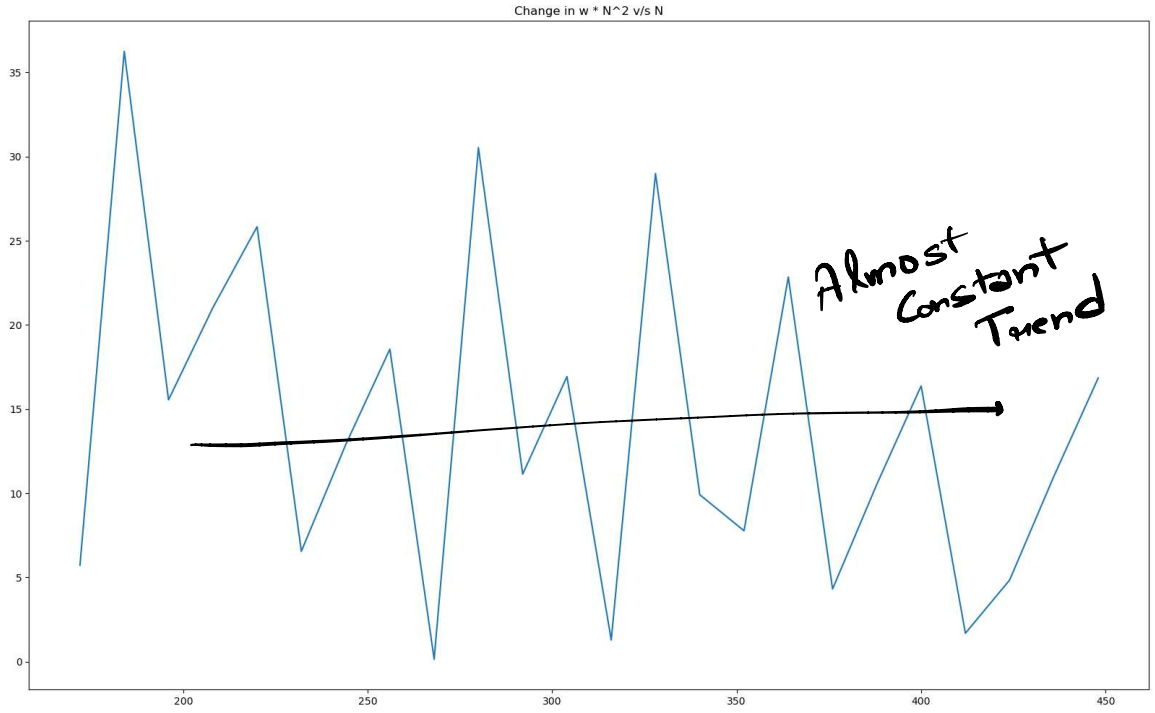
(d)

k can be around 6.25 without starting to loose stability. This value does not depend on delta x

In [12]:
```
# (e)

k = 1/4
# Assuming delta x = 100 points means we should have 100 points between the limi
soluation_vals = []
N_vals = np.arange(160, 451, 12)
print(N_vals)
for n_points in N_vals:
    dx = (math.log(100) - math.log(50))/n_points
    ds = dx*dx*k
    x_vals, s_vals, solution_grid = finite_difference_solver(dx, ds)
    max_val = solution_grid[int(len(x_vals)/2), -1]
    soluation_vals.append(max_val)
    print(n_points, max_val)
soluation_vals = np.array(soluation_vals)
```

```
[160 172 184 196 208 220 232 244 256 268 280 292 304 316 328 340 352 364
 376 388 400 412 424 436 448]
160 115.59040196286642
172 115.59020811457421
184 115.58913748076753
196 115.58873265797227
208 115.58824689184434
220 115.5877131683205
232 115.58759131320222
244 115.58737497969953
256 115.58709172351949
268 115.58708958662727
280 115.58670022555927
292 115.58656954934578
304 115.58638632778005
316 115.586399265924
328 115.58612978359922
340 115.5860439644684
352 115.58610673806686
364 115.58593431759488
376 115.58590376002368
388 115.58583372097705
400 115.58573136480322
412 115.58574134757554
424 115.58571433485147
436 115.58565635988577
448 115.58557239850771
```

In [21]:
```
ord_of_convergence = 2
fig, ax = plt.subplots(1, figsize=(20,12))
ax.plot(N_vals[1:], np.abs(soluation_vals[1:] - soluation_vals[:-1]) * np.power(
ax.set_title(f"Change in w * N^2 v/s N")
fig.show()
```



(e) The order of convergence is 2 based on the above graph, since the change in value / delta x ^ 2, seems to become constant as N increases.

In [ ]: