1) $g_1(x,y) = y \geq 0$

$g_2(x,y) = 2-y \geq 0$

$g_3(x,y) = 4-x \geq 0$

$L(x,y,\lambda_1,\lambda_2,\lambda_3) = f(x,y) + \lambda_1 g_1(x,y) + \lambda_2 g_2(x,y) + \lambda_3 g_3(x,y)$

$$\frac{\partial L}{\partial x} = 1-y-\lambda_3 = 0$$

$$\frac{\partial L}{\partial y} = 1-x-\lambda_1+\lambda_2 = 0$$

$$\frac{\partial L}{\partial \lambda_1} = y = 0$$

$$\frac{\partial L}{\partial \lambda_2} = 2-y = 0$$

$$\frac{\partial L}{\partial \lambda_3} = 4-x = 0$$

$x=4,\ y=2,\ \lambda_1 = \lambda_2 = \lambda_3 = 0$

$\lambda_3 = 1,\ \lambda_2 = 0$

critical point is at $(4,2)$ and at the boundary points $(0,0)$, $(0,2)$ and $(4,0)$

$f(0,0) = 0$

$f(0,2) = 0$

$f(4,0) = 4$

$f(4,2) = 4$

the minimum value is 0, it occurs at $(0,0)$, $(0,2)$ and maximum value occurs at $f(4,0)$ and $f(4,2)$ and it is 4

2) $L(x, y, \lambda) = 2x^2 + xy + y^2 + 500 + \lambda(100 - x - y)$

$$\frac{\partial L}{\partial x} = 4x + y - \lambda = 0$$

$$\frac{\partial L}{\partial y} = x + 2y - \lambda = 0$$

$$\frac{\partial L}{\partial \lambda} = 200 - x - y = 0$$

Solving these equations we get,

$x = 50$, $y = 150$, $\lambda = 150$

$C(x, y) = 2x^2 + xy + y^2 + 500$

$C(50, 150) = 2(50)^2 + 50(150) + (150)^2 + 500$

$= 2(2500) + 7500 + 22500 + 500$

$= 35000 + 500$

$= 35500$

3)

a) we need to take partial derivatives of L with respect to x and λ

$$\frac{\partial L}{\partial x_i} = 0 \text{ for all } i = 1, 2, \ldots, n$$

$$\frac{\partial L}{\partial \lambda} = 0$$

mathematically that gives us

$$\frac{\partial f}{\partial x_i} - \lambda_0 \frac{\partial g}{\partial x_i} = 0, \text{ for all } i = 1, 2, \ldots, n$$

$$g(x_0) = 0$$

this shows that (x_0, λ_0), the gradient of L with respect to x is 0 in all directions and $g(x_0) = 0$. This is a condition for L to have local extrema so, (x_0, λ_0) is a critical point

b) Using Langrange we have several advantages like,

i) incorporating constraints
(ii) Solving systematically
(iii) Flexibility
(iv) General applicability

4)

a) $\frac{\partial f}{\partial x} = 2x e^{x^2} + 1$

$\frac{\partial f}{\partial y} = 2y$

$\frac{\partial f}{\partial x}(0,0) = 1$

$\frac{\partial f}{\partial y}(0,0) = 0$

$\nabla f(0,0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$\frac{\partial^2 f}{\partial x^2} = 2e^{x^2} + 4x^2 e^{x^2}$

$\frac{\partial^2 f}{\partial y^2} = 2$

$\frac{\partial^2 f}{\partial x \partial y} = 0$

$\frac{\partial^2 f}{\partial x \partial y} = 0$

at origin

$H(0,0) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

b) $f(0) = f(0,0) = e^{0^2} + 0 + 0^2 = 1$

$\nabla f(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$Hf(0) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

$f(x) \approx 1 + x_1 + x_1^2 + x_2^2$

$f(x) \approx 1 + x_1 + x_1^2 + x_2^2$

c) $\nabla a(x) = \begin{bmatrix} 1 + 2x_1 \\ 2x_2 \end{bmatrix}$

$1 + 2x_1 = 0 \Rightarrow x_1 = -\frac{1}{2}$

$2x_2 = 0 \Rightarrow x_2 = 0$

so the critical points are $\left(-\frac{1}{2}, 0\right)$

$a\left(-\frac{1}{2}, 0\right) = 1 + \left(-\frac{1}{2}\right) + \left(\frac{1}{2}\right)^2 + 0^2 = 3/4$

d) $H_{(0,0)} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

This is positive definite, as all the eigen values are positive (2, 2), therefore optimization is strictly convex near origin. The quadratic approximation obtained in the previous step supports the conclusion.

1]:
```python
def n_social(n_tv, budget):
    cost_social = 25
    cost_tv = 250

    # Calculate the number of social campaigns using the rearranged formula
    n_social = (budget - cost_tv * n_tv) / cost_social

    return n_social


n_tv_campaigns = 5
budget = 2500

result = n_social(n_tv_campaigns, budget)
print(f"The number of social campaigns to buy when purchasing {n_tv_campaigns} TV campaigns is: {result}")
```

```
The number of social campaigns to buy when purchasing 5 TV campaigns is: 50.0
```

In [8]:
```python
def n_tv(n_social, budget):
    cost_social = 25
    cost_tv = 250

    # Calculate the number of TV campaigns using the corrected formula
    n_tv = (budget - cost_social * n_social) / cost_tv

    return max(0, n_tv)  # Ensure the result is non-negative


n_social_campaigns = 100  # You can change this value based on your scenario
budget = 2500

result = n_tv(n_social_campaigns, budget)
print(f"The number of TV campaigns to buy when purchasing {n_social_campaigns} social campaigns is: {result}")
```

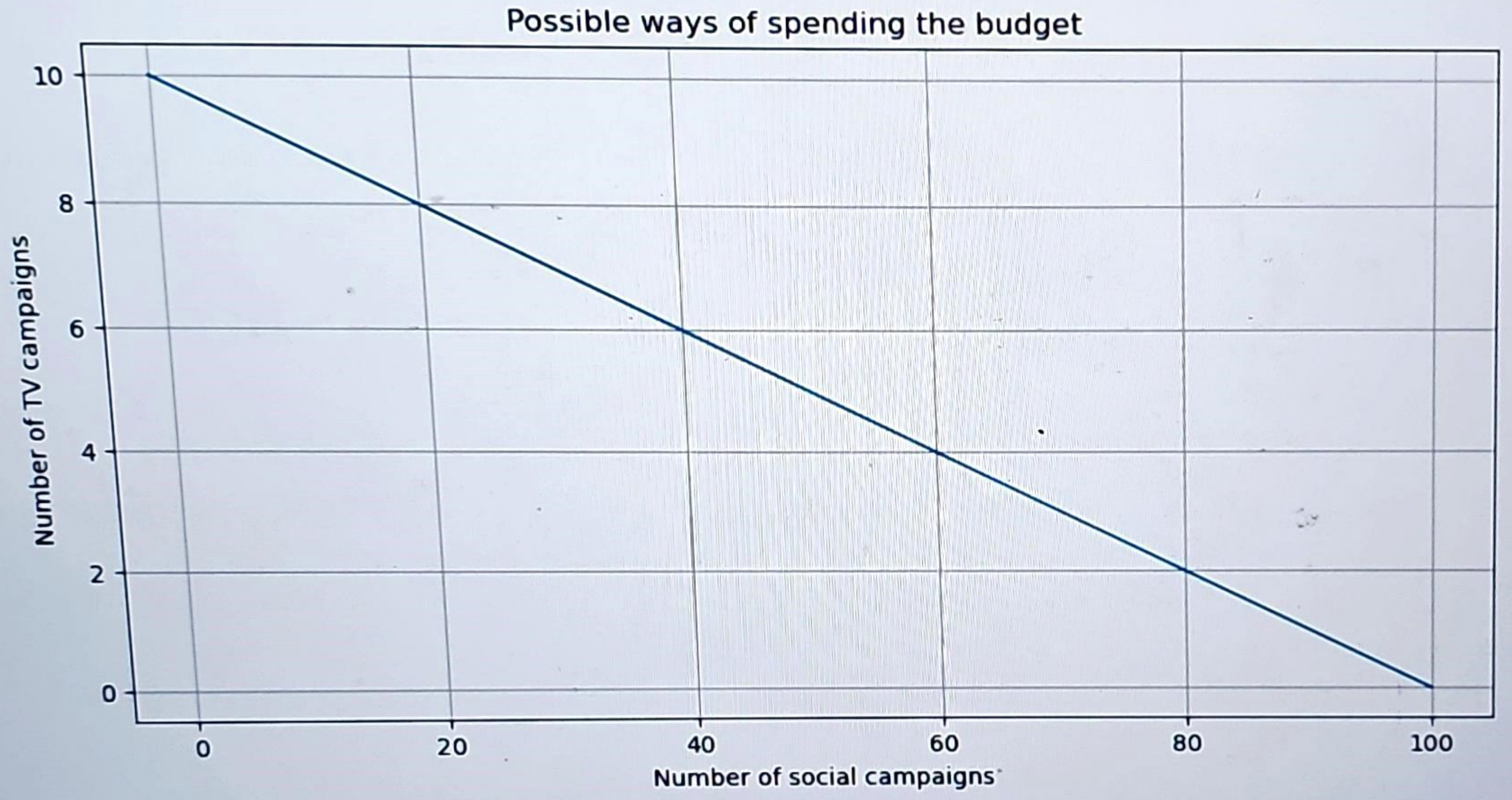
```
budget = 2500

result = n_tv(n_social_campaigns, budget)
print(f"The number of TV campaigns to buy when purchasing {n_social_campaigns} social campaigns is: {result}")
```

```
The number of TV campaigns to buy when purchasing 100 social campaigns is: 0
```

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt

# Define the n_tv function
def n_tv(n_social, budget):
    cost_social = 25
    cost_tv = 250
    n_tv = (budget - cost_social * n_social) / cost_tv
    return max(0, n_tv)

# Define the budget and social campaign range
budget = 2500
social_min = 0
social_max = budget / 25  # Assuming cost_social is 25

# Task 3a: Set the social_x array using the linspace function
social_x = np.linspace(social_min, social_max, 100)

# Task 3b: Set the value 'tv_y' by calling the 'n_tv' function
tv_y = np.vectorize(lambda x: n_tv(x, budget))(social_x)

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(social_x, tv_y)
plt.xlabel('Number of social campaigns')
plt.ylabel('Number of TV campaigns')
plt.title('Possible ways of spending the budget')
plt.grid(True)
plt.show()
```

```
plt.show()
```


Possible ways of spending the budget
Number of TV campaigns
0
2
4
6
8
10
Number of social campaigns
0
20
40
60
80
100

In [11]:

```python
def revenues(social, tv):
    return 7 * (social ** (3/4)) * (tv ** (1/4))

social_campaigns = 10
tv_campaigns = 5

result = revenues(social_campaigns, tv_campaigns)
print(f"The revenue for {social_campaigns} social campaigns and {tv_campaigns} TV campaigns is: {result}")
```

```
The revenue for 10 social campaigns and 5 TV campaigns is: 58.86274906776002
```

In [13]:
```python
import sympy as sp

# Define the variables
social, tv, lmbda = sp.symbols('social tv lmbda')

# Define the Revenue and Constraint functions
revenue = 7 * social**(3/4) * tv**(1/4)
constraint = 25 * social + 250 * tv - 2500

# Define the Lagrangian
lagrangian = revenue - lmbda * constraint

# Calculate the partial derivatives
partial_derivative_social = sp.diff(lagrangian, social)
partial_derivative_tv = sp.diff(lagrangian, tv)

# Solve the system of equations
solution = sp.solve([partial_derivative_social, partial_derivative_tv, constraint], (social, tv, lmbda))

# Extract the social, tv, and lambda values
optimal_social, optimal_tv, optimal_lambda = solution[0][0], solution[0][1], solution[0][2]

print("Optimal Values:")
print(f"Social Campaigns: {optimal_social}")
print(f"TV Campaigns: {optimal_tv}")
print(f"Lagrange Multiplier: {optimal_lambda}")
```

```
Optimal Values:
Social Campaigns: 75.0000000000000
TV Campaigns: 2.50000000000000
Lagrange Multiplier: 0.0897302713432092
```

In [16]:
```python
from sympy import symbols, Eq, solve
```

[16]:
```python
from sympy import symbols, Eq, solve

s, t, l = symbols('s t l')

# Define the equations
equations = [
    Eq((21/4)*((t**(1/4))/s**(1/4)) - 25*l, 0),
    Eq((7/4)*(s**(3/4)/t**(3/4)) - 250*l, 0),
    Eq(25*s + 250*t - 2500, 0)
]

# Solve the system of equations
solutions = solve(equations, (s, t, l), simplify=False)

# Display the solutions
for sol in solutions:
    print("Solution:")
    print(f"Social Campaigns (s): {sol[0]}")
    print(f"TV Campaigns (t): {sol[1]}")
    print(f"Lagrange Multiplier (l): {sol[2]}\n")
```

```
Solution:
Social Campaigns (s): 75.0000000000000
TV Campaigns (t): 2.50000000000000
Lagrange Multiplier (l): 0.0897302713432092
```

[17]:

```
# Extract the first and second values from the first solution
social_value = solutions[0][0]
tv_value = solutions[0][1]

# Call the revenues function with the extracted values
revenue_result = revenues(social_value, tv_value)

# Print the result
print(f"Revenue for Social Campaigns = {social_value}, TV Campaigns = {tv_value}: {revenue_result}")
```

```
Revenue for Social Campaigns = 75.0000000000000, TV Campaigns = 2.50000000000000: 224.325678358023
```