Nguyễn Minh Đức - ITITIU21045

Q1:

(a) $f(x,y,z) = x^2 + y^2 + 2z^2$

$\frac{df}{dx} = 2x$; $\frac{df}{dy} = 2y$; $\frac{df}{dz} = 4z$

$\nabla f(x,y,z) = [2x, 2y, 4z]$

$f_{xx} = 2$ $\quad f_{xy} = 0$ $\quad f_{xz} = 0$

$f_{yx} = 0$ $\quad f_{yy} = 2$ $\quad f_{yz} = 0$

$f_{zx} = 0$ $\quad f_{zy} = 0$ $\quad f_{zz} = 4$

$$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

(b) $f(x_1, x_2) = \ln(e^{x_1} + e^{x_2})$

$\frac{df}{dx_1} = \frac{e^{x_1}}{e^{x_1} + e^{x_2}}$; $\frac{df}{dx_2} = \frac{e^{x_2}}{e^{x_1} + e^{x_2}}$

$\nabla f(x_1, x_2) = \left[\frac{e^{x_1}}{e^{x_1} + e^{x_2}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2}}\right]$

$f_{x_1x_1} = \frac{e^{x_1}(e^{x_1} + e^{x_2}) - e^{2x_1}}{(e^{x_1} + e^{x_2})^2} = \frac{e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2}$

$f_{x_1x_2} = \frac{-e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2}$; $f_{x_2x_1} = \frac{-e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2}$

$f_{x_2x_2} = \frac{e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2}$

$$H = \begin{bmatrix} \frac{e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2} & \frac{-e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2} \\ \frac{-e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2} & \frac{e^{x_1}e^{x_2}}{(e^{x_1} + e^{x_2})^2} \end{bmatrix}$$

$$f(x,y) = -9x + x^2 + 11y + 4y^2 - 2xy$$

Q2. $f(x) = -9x + x^2 + 11y + 4y^2 - 2xy$

$f_x = -9 + 2x - 2y$; $f_y = 11 + 8y - 2x$

$f_{xx} = 2$; $f_{xy} = -2$; $f_{yx} = -2$; $f_{yy} = 8$

$H = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}$ $\nabla f = \begin{bmatrix} -9 + 2x - 2y \\ 11 + 8y - 2x \end{bmatrix}$

The principal mirrors are

$D_1 = 2 > 0$

$D_2 = \begin{vmatrix} 2 & -2 \\ -2 & 8 \end{vmatrix} = 12 > 0$

All ~~principal~~ principal mirrors are positive

Hence it is positive definite

Thus, $f(x)$ is strictly convex

b) $x_0 = 0$, $y_0 = 0$, $h = 0.1$

We use following formula:

$$x_{i+1} = x_i + h\frac{df}{dx}$$

$$y_{i+1} = y_i + h\frac{df}{dy}$$

Iteration 1:

$$x_1 = x_0 + h\frac{df}{dx} = 0 + 0.1 \times (-9 + 2\times 0 - 2\times 0) = -0.9$$

$$y_1 = y_0 + h\frac{df}{dy} = 0 + 0.1 \times (11 + 8\times 0 - 2\times 0) = 1.1$$

Iteration 2.

$$x_2 = x_1 + h\frac{df}{dx} = -2.2$$

$$y_2 = y_1 + h\frac{df}{dy} = 3.26$$

Iteration 3:

$$x_3 = x_2 + h\frac{df}{dx} = -4.192$$

$$f_{min}(x_3, y_3) = 418.411$$

$$y_3 = y_2 + h\frac{df}{dy} = 7.408$$

c) $\nabla f(x,y) = [-9 + 2x - 2y,\ 11 + 8y - 2x]$

$X_1 = (0, 0)$

$S_1 = -\nabla f(X_1) = \cancel{[9, -11]}\ \begin{bmatrix} 9 \\ -11 \end{bmatrix}$

Compute λ_1

$$\lambda_1 = \frac{S_1^T S_1}{S_1^T H S_1} = \frac{[9\ \ -11]\begin{bmatrix} 9 \\ -11 \end{bmatrix}}{[9\ -11]\begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix}\begin{bmatrix} 9 \\ -11 \end{bmatrix}} = \frac{202}{[9\ -11]\begin{bmatrix} 40 \\ -106 \end{bmatrix}} = \frac{202}{1526} = \frac{101}{763}$$

Hence the new point is

$$X_2 = X_1 + \lambda_1 S_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{101}{763}\begin{bmatrix} 9 \\ -11 \end{bmatrix} = \begin{bmatrix} 909/763 \\ -1111/763 \end{bmatrix} = \begin{bmatrix} 1.1913 \\ -1.4561 \end{bmatrix}$$

~~Check the~~

Therefore

Adjugate 2×2

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

- $a \leftrightarrow d$
- c and b times with -1

$\rightarrow \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

d) ∇f at $X_1 \Rightarrow \nabla f_{X_1} = \begin{bmatrix} -9 \\ 11 \end{bmatrix}$

adjoint/~~adjugate~~

$$J_i = H = \begin{bmatrix} 2 & -2 \\ -2 & 8 \end{bmatrix} \Rightarrow J_i^{-1} = \frac{1}{\det J_i}\operatorname{adj}(J_i) = \frac{1}{2\times 8 - (-2)\times(-2)}\begin{bmatrix} 8 & 2 \\ 2 & 2 \end{bmatrix}$$

$$= \frac{1}{12}\begin{bmatrix} 8 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 2/3 & 1/6 \\ 1/6 & 1/6 \end{bmatrix}$$

Iter 1: $X_2 = X_1 - J_1^{-1}\nabla f_{X_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 2/3 & 1/6 \\ 1/6 & 1/6 \end{bmatrix}\begin{bmatrix} -9 \\ 11 \end{bmatrix}$

$$= \begin{bmatrix} 25/6 \\ -1/3 \end{bmatrix}$$

Check: $\nabla f_{X_2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow X_2$ is the optimum point

Q3: a)

Resource	Product A	B	Availability
Raw material	20kg/prod	5kg/prod	9500 kg/week
~~Raw material~~			
Production time	0.04hr/prod	0.12hr/prod	40 hrs/week
Storage	a	b	550 kg/week
Profit	$45/prod	$20/prod	

Let a, b be the products of A and B, respectively

Total Profit = 45a + 20b

Therefore, we have:

Maximize $Z = 45a + 20b$

subject to

$20a + 5b \leq 9500$ (1)

$a + b \leq 550$ (2)

$0.04a + 0.12b \leq 40$ (3)

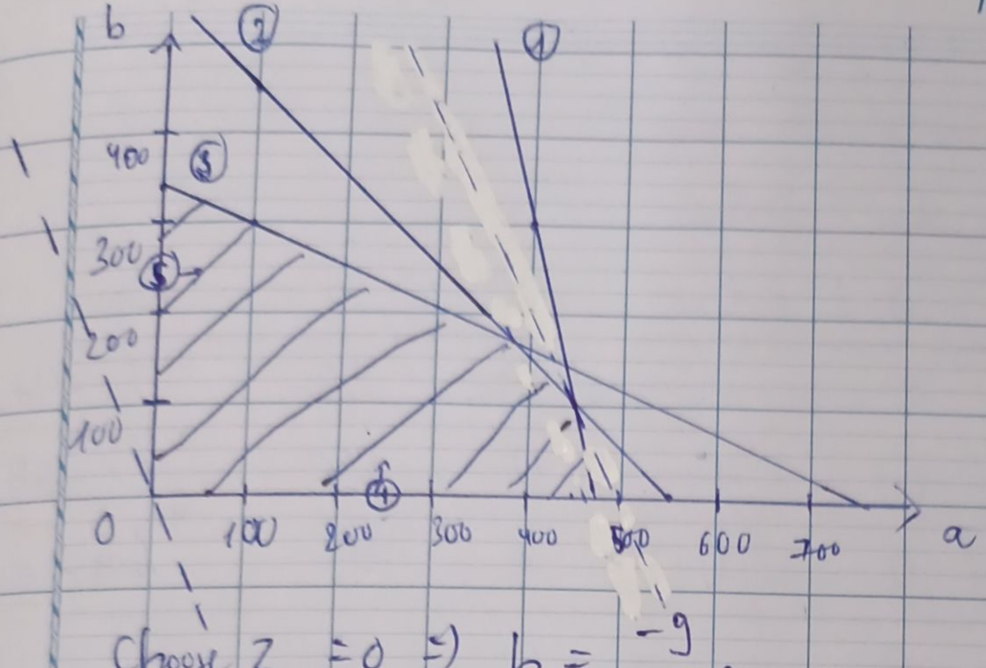
$a, b \geq 0$ (4)(5)

b) (1) $\Leftrightarrow$ $b = 1900 - 4a$ (1)

(2) $\Leftrightarrow$ $b = 550 - a$ (2)

(3) $\Leftrightarrow$ $b = \frac{1000}{3} - \frac{1}{3}a$ (3)

b



Choose Z = 0 => $b = \frac{-9}{4} a$

Maximum value of Z that lies in feasible region is ~~19125~~ 22250

=> a = 450 ; b = 100 (satisfied all conditions)

~~b)~~ c) We ~~ha~~ use simplex method and have:

Max $45a + 20b + 0S_1 + 0S_2 + 0S_3$

subject to $20a + 5b + S_1 = 9500$

$a + b + S_2 = 550$

$0.04a + 0.12b + S_3 = 40$

$a, b, S_1, S_2, S_3 \geq 0$

Set $a = S_1 = 0$, therefore we can solve the equations

$$\Rightarrow \begin{cases} b = 1900 \\ S_2 = -1350 \\ S_3 = -188 \end{cases} \rightarrow \text{Not feasible}$$

Set $a = b = 0$ $\Rightarrow \begin{cases} S_1 = 9500 \\ S_2 = 550 \\ S_3 = 40 \end{cases}$

Basis	c_B	a	b	S_1	S_2	S_3	b	Ratio
		45	20	0	0	0		
S_1	0	20	5	1	0	0	9500	$9500/20 = 475$ → key (minimum)
S_2	0	1	1	0	1	0	550	$550/1 = 550$
S_3	0	0.04	0.12	0	0	1	40	$40/0.04 = 1000$
z_j		0	0	0	0	0	0	
$c_j - z_j$		45	20	0	0	0		

*Note: $R_1 \to R_1/20$

$R_2 \to R_2 - R_1$

$R_3 \to R_3 - 0.04R_1$

• Iteration 1 (Use key row to be pivot)

Basis	c_B	a	b	S_1	S_2	S_3	b	Ratio
		45	20	0	0	0		
a	45	1	0.25	0.05	0	0	475	$475/0.25 = 1900$
S_2	0	0	0.75	-0.05	1	0	75	$75/0.75 = 100$ → Pivot (minimum)
S_3	0	0	0.11	-0.002	0	1	21	$21/0.11 = 190.91$
z_j		45	11.25	2.25	0	0	21375 → Profit	
$c_j - z_j$		0	8.75	-2.25	0	0		

Positive means that profit can still be improved

Pivot because most positive net evaluation row

• Iteration 2

$R_3 \to R_3 - 0.11R_2$

$R_1 \to R_1 - 0.25R_2$

Basis	c_B	a	b	S_1	S_2	S_3	b
		45	20	0	0	0	
a	45	1	0	$1/15$	$-1/3$	0	450
b	20	0	1	$\frac{-1}{15}$	$\frac{4}{3}$	0	100
S_3	0	0	0	$\frac{2}{375}$	$\frac{-11}{75}$	1	10
z_j		45	20	$5/3$	$35/3$	0	22250
$c_j - z_j$		0	0	$-5/3$	$-35/3$	0	

→ Optimal solution

a = 450

b = 100

Z = 22250

→ Cannot be improved because of ~~negative~~ non-positive

e) The storage will give the maximum profit

According to the graph, we see that if we want to obtain the maximum profit, the shadow price should be high to get the optimal point

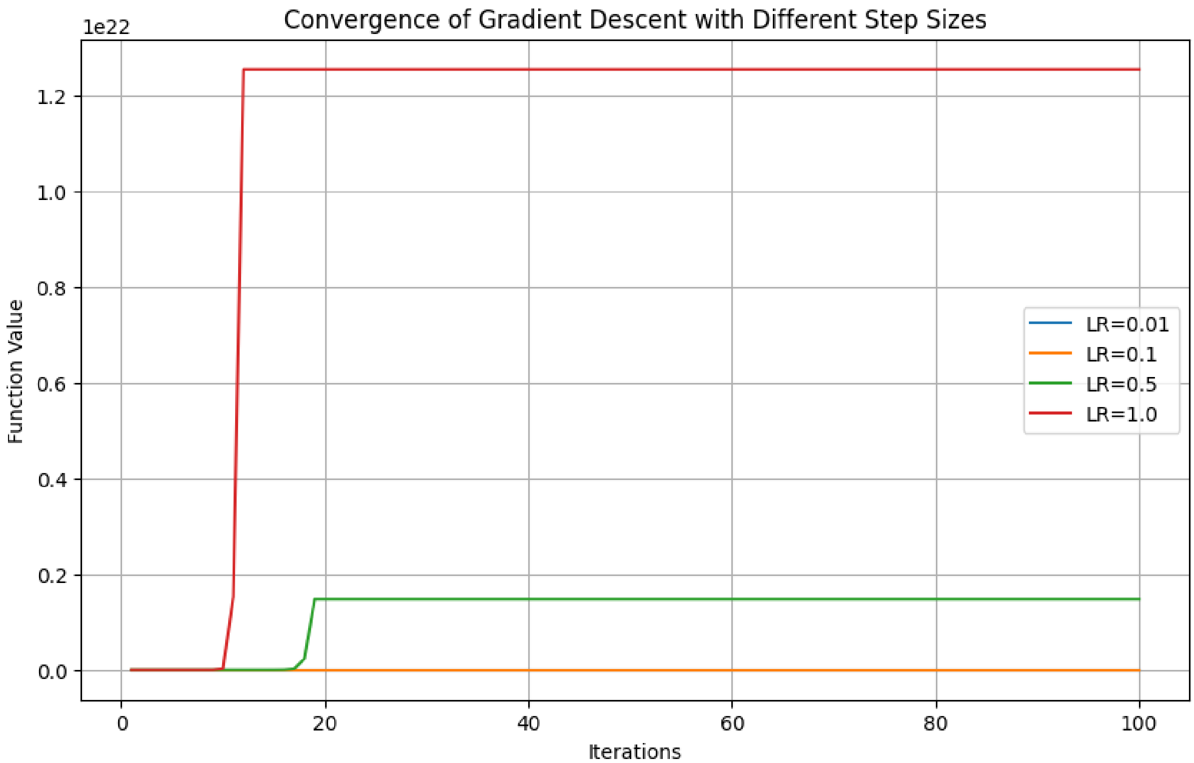
Therefore the 2 equation is the most likely line to increase the profit to gain the maximum profit

Thus, increasing storage will raise profits the most

```
#Code by Nguyen Minh Duc_ITITIU21045_Lab05TMC_EX1a
import sympy as sp
import numpy as np
x,y,z = sp.symbols('x y z')
f = x**2 + y**2 + 2*z**2
gradient = [sp.diff(f,var) for var in (x, y, z)]
print("Gradient vector: ")
print(gradient)
hessian = sp.hessian(f,(x,y,z))
print("\nHessian matrix: ")
print(hessian)
```

```
Gradient vector:
[2*x, 2*y, 4*z]

Hessian matrix:
Matrix([[2, 0, 0], [0, 2, 0], [0, 0, 4]])
```

```
#Code by Nguyen Minh Duc_ITITIU21045_Lab05TMC_EX2b
import numpy as np
import matplotlib.pyplot as plt

# Define the function you want to optimize
def f(x, y):
    return -9*x + x**2 + 11*y + 4*y**2 - 2*x*y

# Define the numerical gradient calculation using the central difference method
def numerical_gradient(f, x, y, epsi=1e-6):
    grad_x = (f(x + epsi, y) - f(x - epsi, y)) / (2 * epsi)
    grad_y = (f(x, y + epsi) - f(x, y - epsi)) / (2 * epsi)
    return grad_x, grad_y

# Gradient Descent function
def gradient_descent(f, init_x, init_y, learning_rate, num_iter):
    x = init_x
    y = init_y
    history = []
    for _ in range(num_iter):
        gradient_x, gradient_y = numerical_gradient(f, x, y)
        x = x - learning_rate * gradient_x
        y = y - learning_rate * gradient_y
        history.append(f(x, y))
    return x, y, history

# Set the initial points, step sizes, and number of iterations
init_x = 2.0
init_y = 3.0
learning_rates = [0.01, 0.1, 0.5, 1.0]
num_iter = 100

# Initialize the figure for plotting
plt.figure(figsize=(10, 6))
for learning_rate in learning_rates:
    optimal_x, optimal_y, history = gradient_descent(f, init_x, init_y, learning_rate, num_iter)
    iter = range(1, num_iter + 1)
    plt.plot(iter, history, label=f'LR={learning_rate}')
plt.title("Convergence of Gradient Descent with Different Step Sizes")
plt.xlabel("Iterations")
plt.ylabel("Function Value")
plt.legend()
plt.grid(True)
plt.show()
```

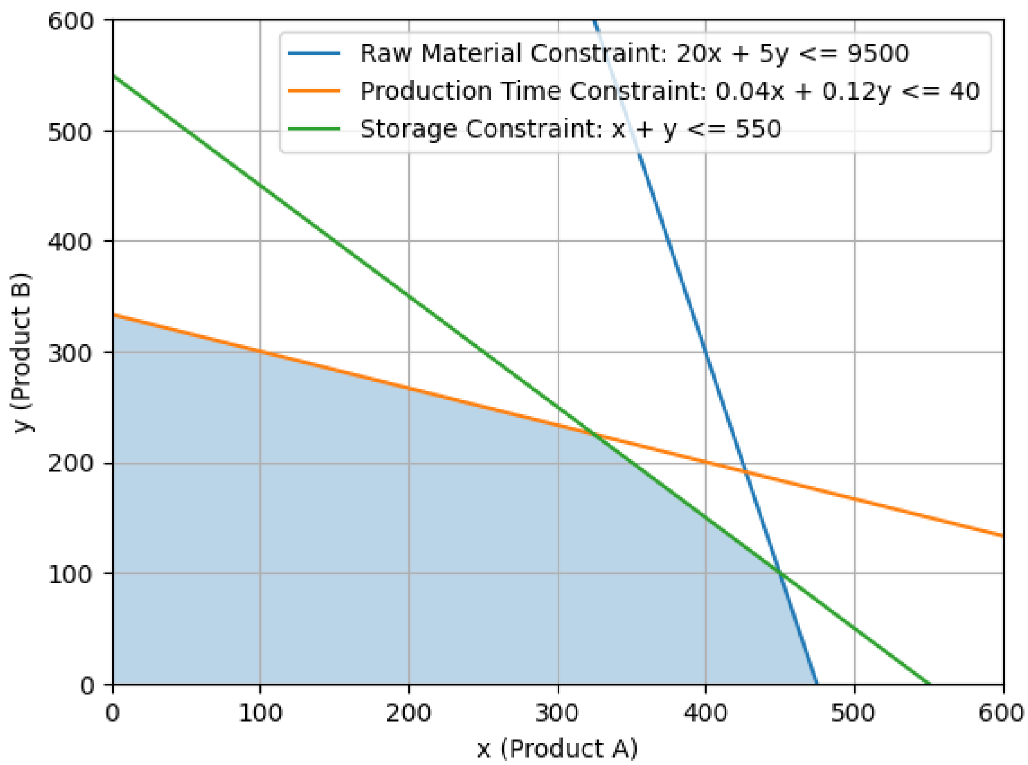


Convergence of Gradient Descent with Different Step Sizes
1e22
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Function Value
0
20
40
60
80
100
Iterations
LR=0.01
LR=0.1
LR=0.5
LR=1.0

```
from os import X_OK
#Code by Nguyen Minh Duc_ITITIU21045_Lab05TMC_EX2c

# Define the function you want to optimize
def f(x, y):
    return -9*x + x**2 + 11*y + 4*y**2 - 2*x*y

# Define the gradient of the function
def gradient(x, y):
    grad_x = -9 + 2*x - 2*y
    grad_y = 11 + 8*y - 2*x
    return grad_x, grad_y

# Initial guesses
x_o = 0
y_o = 0

# Learning rate (step size)
alpha = 0.1

# Perform one iteration of steepest descent
grad_x, grad_y = gradient(x_o, y_o)
x_new = x_o - alpha * grad_x
y_new = y_o - alpha * grad_y

# Print the updated values
print("Updated x:", x_new)
print("Updated y:", y_new)

# Evaluate the function value at the updated point
min_value = f(x_new, y_new)
print("Minimum value of the function:", min_value)
```

```
Updated x: 0.9
Updated y: -1.1
Minimum value of the function: -12.57
```

```
#Code by Nguyen Minh Duc_ITITIU21045_Lab05TMC_EX3b
import matplotlib.pyplot as plt
import numpy as np

# Define the constraints
x = np.linspace(0, 600, 400)  # Generate x values
y1 = (9500 - 20 * x) / 5
y2 = (40 - 0.04 * x) / 0.12
y3 = 550 - x

# Plot the constraints
plt.plot(x, y1, label="Raw Material Constraint: 20x + 5y <= 9500")
plt.plot(x, y2, label="Production Time Constraint: 0.04x + 0.12y <= 40")
plt.plot(x, y3, label="Storage Constraint: x + y <= 550")

# Fill the feasible region
plt.fill_between(x, np.minimum(np.minimum(y1, y2), y3), 0, where=(x >= 0) & (x <= 550) & (y1 >= 0) & (y2


# Add labels and legend
plt.xlabel("x (Product A)")
plt.ylabel("y (Product B)")
plt.legend(loc="upper right")
plt.grid(True)
plt.xlim(0, 600)
plt.ylim(0, 600)
# Show the plot
plt.show()
```



```python
#Code by Nguyen Minh Duc_ITITIU21045_Lab05TMC_EX3d
import cvxpy as cp

# Define the variables
x = cp.Variable()
y = cp.Variable()

# Define the objective function to maximize profit
objective = cp.Maximize(45 * x + 20 * y)

# Define the constraints
constraints = [
    20 * x + 5 * y <= 9500,
    0.04 * x + 0.12 * y <= 40,
    x + y <= 550,
    x >= 0,
    y >= 0
]

# Create the problem
problem = cp.Problem(objective, constraints)

# Solve the problem
problem.solve()

# Print the optimal solution and profit
optimal_x = x.value
optimal_y = y.value
optimal_profit = objective.value
print("Optimal solution:")
print(f"x = {optimal_x}")
print(f"y = {optimal_y}")
print(f"Optimal profit = ${optimal_profit:.2f}")
```

```
Optimal solution:
x = 449.9999999910152
y = 99.9999998279479
Optimal profit = $22250.00
```