Ασκηση 1

α) $\min\limits_{x_1,x_2\in\mathbb{R}} 2x_1^2+2x_2^2-6x_1x_2+7x_1$

Αρχίζουμε από $\vec{x_0}=(0,0)$

Βήμα 1

$\vec{s}=-\nabla f(0,0)=\left(4x_1-6x_2+7\Big|_{x_1=x_2=0},\ 4x_2-6x_1\Big|_{x_1=x_2=0}\right)=(7,0)$

Ο περιορισμός της f στη γραμμή $\vec{x_0}+t\vec{s}=(0,0)+t(7,0)=(7t,0)$

Είναι η συνάρτηση $g(t)=f(7t,0)=2(7t)^2+2\cdot 0^2-6\cdot 7t\cdot 0+7\cdot 7t=$

$=98t^2+0+49t=\underset{\text{Κυρτή}}{\underline{98t^2+49t}}$

$g'(t)=196t+49$

Η παράγωγος μηδενίζεται στο $t^*=-\frac{1}{4}$

Αρα η ελάχιστη τιμή της f λαμβάνεται στο σημείο $\left(-\frac{7}{4},0\right)=\vec{x_1}$

Για να διαπιστώσουμε αν αυτή είναι η βέλτιστη λύση, ελέγχουμε την παράγωγο

της f στο $\vec{x_1}$: $-\nabla f\left(-\frac{7}{4},0\right)=\left(4x_1-6x_2+7\Big|_{\substack{x_1=-\frac{7}{4}\\x_2=0}},\ 4x_2-6x_1\Big|_{\substack{x_1=-\frac{7}{4}\\x_2=0}}\right)=$

$=\left(-7+7,\ -\frac{42}{4}\right)=\left(0,-\frac{21}{2}\right)=(0,10{,}5)$ Δεν είναι 0, αρα δεν είναι βέλτιστη λύση.

Αρα απαιτείται περαιτέρω επανάληψη της μεθόδου για να βρεθεί βέλτιστη λύση β) $\vec{X_1} = \underbrace{\vec{X_0}}_{(0,0)} - H(f, \vec{X_0})^{-1} \underbrace{\nabla F(\vec{X_0})}_{(7,0)}$

$H(f, \vec{X_0})^{-1}$

- $\dfrac{\partial^2 F(\vec{X_0})}{\partial X_1^2} = 4$

$\dfrac{\partial^2 F(\vec{X_0})}{\partial X_2^2} = 4$

- $\dfrac{\partial^2 F(\vec{X_0})}{\partial X_1 \partial X_2} = -6$

$HF(X_1, X_2) = \begin{bmatrix} 4 & -6 \\ -6 & 4 \end{bmatrix}$

$\det(H) = 4 \cdot 4 - (-6 \cdot (-6)) = 16 - 36 = -20$

$H^{-1} = \dfrac{1}{\det(H)} \cdot \begin{bmatrix} 4 & 6 \\ 6 & 4 \end{bmatrix} = \dfrac{1}{-20} \begin{bmatrix} 4 & 6 \\ 6 & 4 \end{bmatrix} = \begin{bmatrix} -\frac{1}{5} & -\frac{3}{10} \\ -\frac{3}{10} & -\frac{1}{5} \end{bmatrix}$

Άρα :

$\vec{X_1} = (0,0) - \begin{bmatrix} -\frac{1}{5} & -\frac{3}{10} \\ -\frac{3}{10} & -\frac{1}{5} \end{bmatrix} \cdot (7,0) = -\left[-\frac{7}{5} + 0, -\frac{21}{10} + 0 \right] =$

$\boxed{\vec{X_1} = \left(-\frac{7}{5}, -\frac{21}{10} \right)}$

Άσκηση 3

Συνάρτηση επιπέδου $x+3\psi+2z=5 \Rightarrow x+3\psi+2z-5=0$

Η απόσταση ανάμεσα σε 2 σημεία στο τρισδιάστατο επίπεδο εκφράζεται ως εξής:

$$d=\sqrt{(x_2-x_1)^2+(\psi_2-\psi_1)^2+(z_2-z_1)^2}$$

Για απόσταση από την αρχή των αξόνων ως το επίπεδο μπορούμε να εφαρμόσουμε τον παρακάτω τύπο:

$$d_2=\frac{|1\cdot 0+3\cdot 0+2\cdot 0-5|}{\sqrt{1+3+2}}=\boxed{\frac{5}{\sqrt{6}} \text{ min απόσταση}}$$

$$\min d=\frac{|ax_0+b\psi_0+cz_0+d|}{\sqrt{a^2+b^2+c^2}}$$

$(x_0,\psi_0,z_0)=(0,0,0)$

$a=1,\ b=3,\ c=2,\ d=-5$

Άσκηση 4

Το πρόβλημα εκφράζεται ως :

$\min 2x_1^2 + x_2^2 + 3x_3^2$

ε.ω. $x_1 + x_2 + x_3 = 10$, $x_1, x_2, x_3 \geq 0$

Λύνουμε με μέθοδο Lagrange :

$$L(x_1, x_2, x_3; \lambda_0) = 2x_1^2 + x_2^2 + 3x_3^2 + \lambda_0(x_1 + x_2 + x_3 - 10)$$
$$= (2x_1^2 + \lambda_0 x_1) + (x_2^2 + \lambda_0 x_2) + (3x_3^2 + \lambda_0 x_3) - 10\lambda_0$$

Παρατηρούμε $\forall \lambda_0 \in \mathbb{R}$:

- $\min\limits_{x_1 \in \mathbb{R}} 2x_1^2 + \lambda_0 x_1 > -\infty$
- $\min\limits_{x_2 \in \mathbb{R}} x_2^2 + \lambda_0 x_2 > -\infty$
- $\min\limits_{x_3 \in \mathbb{R}} 3x_3^2 + \lambda_0 x_3 > -\infty$

Άρα $\min\limits_{x_1, x_2, x_3 \in \mathbb{R}} L(x_1, x_2, x_3; \lambda_0) > -\infty \quad \forall \lambda_0 \in \mathbb{R}$

Για κάθε $\lambda_0 \in \mathbb{R}$ βρίσκουμε τη βέλτιστη λύση του $\min\limits_{x_1, x_2, x_3 \in \mathbb{R}} L(x_1, x_2, x_3; \lambda_0)$

$$\nabla L(x_1, x_2, x_3; \lambda_0) = 0 \Rightarrow \begin{cases} \frac{\partial L}{\partial x_1} = 0 \\ \frac{\partial L}{\partial x_2} = 0 \\ \frac{\partial L}{\partial x_3} = 0 \end{cases} \Rightarrow \begin{cases} 4x_1 + \lambda_0 = 0 \\ 2x_2 + \lambda_0 = 0 \\ 6x_3 + \lambda_0 = 0 \end{cases}$$

Άρα :

$$x_1^* = x_1(\lambda_0) = -\frac{\lambda_0}{4}, \quad x_2^* = x_2(\lambda_0) = -\frac{\lambda_0}{2}, \quad x_3^* = x_3(\lambda_0) = -\frac{\lambda_0}{6}$$

λ_0 πρέπει να ικανοποιεί το εξής :

$$x_1(\lambda_0) + x_2(\lambda_0) + x_3(\lambda_0) = 10 \Rightarrow -\frac{\lambda_0}{4} - \frac{\lambda_0}{2} - \frac{\lambda_0}{6} = 10 \Rightarrow$$

$$\Rightarrow -\frac{3\lambda_0}{12} - \frac{6\lambda_0}{12} - \frac{2\lambda_0}{12} = 10 \Rightarrow -\frac{11\lambda_0}{12} = 10 \Rightarrow \lambda_0 = \frac{-120}{11}$$

Βέλτιστη λύση :

$$x_1\left(-\frac{120}{11}\right) = \frac{30}{11}$$

$$x_2\left(-\frac{120}{11}\right) = \frac{60}{11}$$

$$x_3\left(-\frac{120}{11}\right) = \frac{20}{11}$$

Άσκηση 2

α) Πεδίο Ορισμού Συνάρτησης $\longrightarrow \{(x_1,x_2): -1-2x_1+x_2>0\}$

Είναι Κυρτό Σύνολο

- $F_1 = x_1^2$, $f_1' = 2x_1$, $f_1'' = 2 > 0$ $\rightarrow$ κυρτή

- $f_2 = x_2^2$, $f_2' = 2x_2$, $f_2'' = 2 > 0$ $\rightarrow$ Κυρτή

- $F_3 = -\log(-1-2x_1+x_2)$

  Είναι σύνθεση της $f(\psi) = -\log\psi$ που είναι κυρτή και της γραμμικής συνάρτησης $-1-2x_1+x_2$

  Άρα F_3 κυρτή

Συνεπώς το άθροισμα $f_1+f_2+F_3$ είναι Κυρτή συνάρτηση

ΑΣΚΗΣΗ 2

ΕΡΩΤΗΜΑΤΑ Β,Γ,Δ

ΚΩΔΙΚΑΣ PYTHON:

```python
import math


# Υπολογισμός της αντικειμενικής συνάρτησης
def objective_function(x, y):
   return x**2 + y**2 - math.log(-1 - 2*x + y)


# Υπολογισμός της κλίσης της αντικειμενικής συνάρτησης
def gradient(x, y):
   df_dx = 2*x + (2/(1 + 2*x - y))
   df_dy = 2*y - (1/(1 + 2*x - y))
   return df_dx, df_dy


# Αναζήτηση γραμμής με ακριβή αναζήτηση
def line_search(x, y, direction):
   step_size = 0.001  # Αρχικό μήκος βήματος
   alpha = 0.5  # Συντελεστής μείωσης μήκους βήματος

   while True:
      x_new = x + step_size * direction[0]
      y_new = y + step_size * direction[1]
      if objective_function(x_new, y_new) < objective_function(x, y):
         return x_new, y_new
      step_size *= alpha
```

```
# Πιο απότομη κατάβαση με ακριβή αναζήτηση
def gradient_descent_exact():
    x0, y0 = 0, 0  # Αρχική εκτίμηση
    max_iterations = 1000
    epsilon = 1e-6  # Κατώφλι σύγκλισης

    for i in range(max_iterations):
        df_dx, df_dy = gradient(x0, y0)
        direction = (-df_dx, -df_dy)
        x1, y1 = line_search(x0, y0, direction)

        if math.sqrt((x1 - x0)**2 + (y1 - y0)**2) < epsilon:
            return x1, y1

        x0, y0 = x1, y1

# Πιο απότομη κατάβαση με συντελεστή οπισθοδρόμησης 0.5
def gradient_descent_backtracking():
    x0, y0 = 0, 0  # Αρχική εκτίμηση
    max_iterations = 1000
    epsilon = 1e-6  # Κατώφλι σύγκλισης
    c = 0.5  # Συντελεστής οπισθοδρόμησης

    for i in range(max_iterations):
        df_dx, df_dy = gradient(x0, y0)
        direction = (-df_dx, -df_dy)

        step_size = 1.0
        while objective_function(x0 + step_size * direction[0], y0 + step_size * direction[1]) >=
objective_function(x0, y0):
            step_size *= c

        x1 = x0 + step_size * direction[0]
```

```
        y1 = y0 + step_size * direction[1]

        if math.sqrt((x1 - x0)**2 + (y1 - y0)**2) < epsilon:
            return x1, y1

        x0, y0 = x1, y1

# Μέθοδος Newton με συντελεστή οπισθοδρόμησης 0.5
def newton_method_backtracking():
    x0, y0 = 0, 0  # Αρχική εκτίμηση
    max_iterations = 1000
    epsilon = 1e-6  # Κατώφλι σύγκλισης
    c = 0.5  # Συντελεστής οπισθοδρόμησης

    for i in range(max_iterations):
        df_dx, df_dy = gradient(x0, y0)
        d2f_dx2 = 2 + (2/((1 + 2*x0 - y0)**2))
        d2f_dy2 = 2 + (1/((1 + 2*x0 - y0)**2))
        d2f_dxdy = -2/((1 + 2*x0 - y0)**2)
        hessian = [[d2f_dx2, d2f_dxdy], [d2f_dxdy, d2f_dy2]]
        inv_hessian = [[hessian[1][1], -hessian[0][1]], [-hessian[1][0], hessian[0][0]]]
        direction = (-inv_hessian[0][0] * df_dx + -inv_hessian[0][1] * df_dy, -inv_hessian[1][0] * df_dx + -
inv_hessian[1][1] * df_dy)

        step_size = 1.0
        while objective_function(x0 + step_size * direction[0], y0 + step_size * direction[1]) >=
objective_function(x0, y0):
            step_size *= c

        x1 = x0 + step_size * direction[0]
        y1 = y0 + step_size * direction[1]

        if math.sqrt((x1 - x0)**2 + (y1 - y0)**2) < epsilon:
```

```python
        return x1, y1

    x0, y0 = x1, y1

# Κλήση των συναρτήσεων για την προσέγγιση της βέλτιστης λύσης
print("Πιο απότομη κατάβαση με ακριβή αναζήτηση:")
x, y = gradient_descent_exact()
print("x =", x)
print("y =", y)
print("f(x, y) =", objective_function(x, y))

print("\nΠιο απότομη κατάβαση με συντελεστή οπισθοδρόμησης 0.5:")
x, y = gradient_descent_backtracking()
print("x =", x)
print("y =", y)
print("f(x, y) =", objective_function(x, y))

print("\nΜέθοδος Newton με συντελεστή οπισθοδρόμησης 0.5:")
x, y = newton_method_backtracking()
print("x =", x)
print("y =", y)
print("f(x, y) =", objective_function(x, y))
```