College of Engineering
School of Aerospace Engineering

AE 6511: Optimal Guidance and Controls

HW2

Professor:
Panagiotis Tsiotras
Gtech AE Professor

Student:
Tomoki Koike
Gtech MS Student

September 30, 2021

Table of Contents

Problem 1

Solve the following equality minimization problem

$$min \quad f(x) = x$$

subject to

$$g(x, y) = y^2 + x^4 - x^3 = 0.$$

Solution:
The equality constraint shows that

$$y^2 + x^3(x - 1) = 0.$$

Since,

$$y = \sqrt{x^3(1 - x)}$$

looks like the following plot

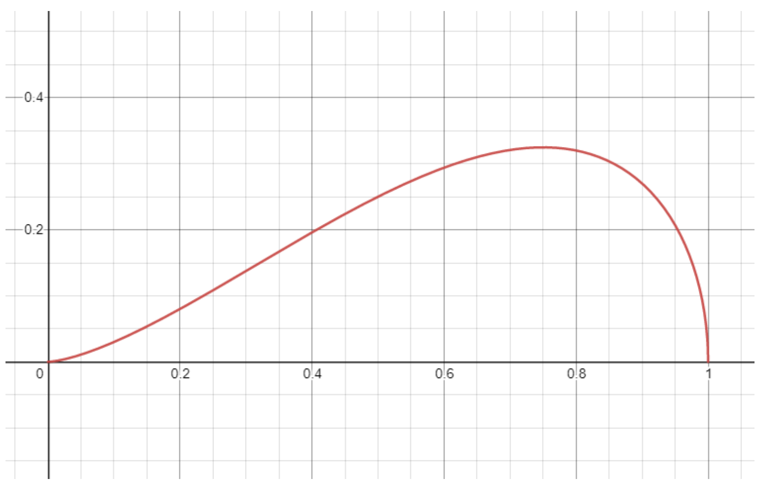

Figure 1: Plot of $y = \sqrt{x^3(1 - x)}$.

Note that the other half in the fourth quadrant is symmetric by the y-axis. This means that the value of x is bounded as

$$x \in [0, 1].$$

Thus, the solution for this minimization problem becomes

$$\boxed{x = 0, \quad y = 0, \quad min\{f(x)\} = 0.}$$

Problem 2

Solve the following optimization problem:

$$min \quad x_1^2 - x_2$$

subject to

$$x_1^2 + x_2^2 \leq 1, \quad x_2 \leq 2, \quad x_1^3 + x_2 = 1.$$

Solution:
The constraints are

$$\begin{cases} h_1(x_1, x_2) = x_1^2 + x_2^2 - 1 \leq 0 \\ h_2(x_2) = x_2 - 2 \leq 0 \\ g_1(x_1, x_2) = x_1^3 + x_2 - 1 \leq 0 \end{cases}$$

The Lagrangian equation becomes

$$\mathscr{L}(x, \mu, \lambda) = \mu(x_1^2 - x_2) + \lambda_1(x_1^2 + x_2^2 - 1) + \lambda_2(x_2 - 2) + \lambda_3(x_1^3 + x_2 - 1)$$

where $m = 2$ and $l = 3$. Taking the first derivative of this, we obtain

$$\mathscr{L}_x(x, \mu, \lambda) = \begin{bmatrix} 2\mu x_1 + \lambda_1(2x_1) + \lambda_3(3x_1^2) \\ -\mu + \lambda_1(2x_2) + \lambda_2 + \lambda_3 \end{bmatrix}^T$$

and even if $\mu = 1$ there is no loss of generality, thus

$$\mathscr{L}_x(x, 1, \lambda) = \begin{bmatrix} 2x_1 + \lambda_1(2x_1) + \lambda_3(3x_1^2) \\ -1 + \lambda_1(2x_2) + \lambda_2 + \lambda_3 \end{bmatrix}^T.$$

If $x_0 = (x_{01}, x_{02})$ is a local minimizer, then the necessary conditions imply the existence of multiplier $(\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3$ such that

$$\begin{cases} 2x_{01} + 2\lambda_1 x_{01} + 3\lambda_3 x_{01}^3 \\ -1 + 2\lambda_1 x_{02} + \lambda_2 + \lambda_3 \\ \lambda_1 \geq 0, \quad \lambda_2 \geq 0 \\ \lambda_1(x_{01}^2 + x_{02}^2 - 1) = 0 \\ \lambda_2(x_2 - 2) = 0 \\ x_1^3 + x_2 - 1 = 0 \end{cases}$$

The possible values are shown below (computed using MATLAB: Problem 2: MATLAB Code)

x_1	x_2	λ_1	λ_2	λ_3
1	0	-2.5000	0	1
-1	2	0	1.6667	-0.6667
0.5437	0.8393	-5.6443	0	10.4744
0	1	0.5000	0	0

The acceptable ones are the second and fourth rows since they suffice the Kuhn-Tucker condition. However, the second row violates constraints h_1 and g_1. Hence, only the fourth row is a candidate local minimizer. Next, we check the 2nd order sufficient condition. The second derivative of the Lagrangian equation is

$$\mathscr{L}_{xx} = \begin{bmatrix} 2 + 2\lambda_1 + 6\lambda_3 x_1 & 0 \\ 0 & 2\lambda_2 \end{bmatrix}$$

and substituting $x_0 = (0, 1)$ and $\lambda^* = (0.5000, 0, 0)$ we get

$$\mathscr{L}_{xx}(x_0, \lambda^*) = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} > 0.$$

Now the sufficient condition requires the following

$$y^T \mathscr{L}_{xx} y > 0 \qquad \forall y \in \mathscr{M}'$$

where

$$\mathscr{M}' = \{y : g_i'(x_0)y = 0, i = 1, ..., p \quad \cap \quad h_i'(x_0)y = 0, i \in \mathscr{T}'(x_0)\}$$

where

$$\mathscr{T}'(x_0) = \{i = 1, ..., q : h_i'(x_0) = 0, \lambda_i > 0\}.$$

Since,

$$\begin{aligned} h_1'(x_0) &= \begin{bmatrix} 0 & 2 \end{bmatrix} \\ h_2'(x_0) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ g_1'(x_0) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \end{aligned}$$

we obtain

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

By checking

$$y^T \mathscr{L}_{xx} y = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = 3 > 0.$$

We can see that the sufficient condition is satisfied, and thus the answer to the constrained minimization problem is

$$\boxed{(x_1, x_2) = (0, 1) \qquad min\{f(x)\} = -1.}$$

Problem 3

Solve the following problem

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

subject to

$$\frac{1}{2}(x_1^2 + x_2^2 + x_3^2) = 1.$$

Solution:
The problem equation and constraint equation are

$$f(x) = x_1 + x_2^2 + x_2 x_3 + 2x_3^2$$
$$g(x) = \frac{1}{2}(x_1^2 + x_2^2 + x_3^2) - 1.$$

The Lagrangian equation becomes

$$\mathscr{L}(x, \mu, \lambda) = \mu(x_1 + x_2^2 + x_2 x_3 + 2x_3^2) + \lambda(\frac{1}{2}(x_1^2 + x_2^2 + x_3^2) - 1).$$

Here there is no loss of generality even if $\mu = 1$ due to strong normality. The first derivative of the Lagrangian equation is

$$\mathscr{L}_x(x, 1, \lambda) = \begin{bmatrix} 1 + \lambda x_1 & (\lambda + 2)x_2 + x_3 & x_2 + (\lambda + 4)x_3 \end{bmatrix}$$

If $x^* = (x_1^*, x_2^*, x_3^*)$ is a local minimizer, then the necessary conditions imply the existence of multiplier $\lambda \in \mathbb{R}$ such that

$$\begin{cases} 1 + \lambda x_1 = 0 \\ (\lambda + 2)x_2 + x_3 = 0 \\ x_2 + (\lambda + 4)x_3 = 0 \\ \frac{1}{2}(x_1^2 + x_2^2 + x_3^2) - 1 = 0 \end{cases}$$

Using MATLAB (refer to Problem 3: MATLAB Code) we can compute the possible values that suffice the above conditions.

x_1	x_2	x_3	λ
0.2265	-0.5342	-1.2897	-4.4142
0.2265	0.5342	1.2897	-4.4142
1.4142	0	0	-0.7071
-1.4142	0	0	0.7071
0.6306	1.1695	-0.4844	-1.5858
0.6306	-1.1695	0.4844	-1.5858

All of them are possible solutions, and therefore, we check each one with the second derivative which is

$$\mathscr{L}_{xx} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda+2 & 1 \\ 0 & 1 & \lambda+4 \end{bmatrix}.$$

Substituting the λ for all possible results we can find the local minimizer. However, we can tell from the shape of the matrix that when $\lambda < 0$ it can not be positive definite or positive semi-definite, and therefore the fourth candidate with $\lambda = 0.7071$ will give us the local minimizer. But we will check this just in case.

When $\lambda = -4.4142$

$$\mathscr{L}_{xx} = \begin{bmatrix} -4.4142 & 0 & 0 \\ 0 & -2.4142 & 1.0000 \\ 0 & 1.0000 & -0.4142 \end{bmatrix} \qquad eig(\mathscr{L}_{xx}) = \begin{bmatrix} -4.4142 \\ -2.8284 \\ 0 \end{bmatrix}$$

When $\lambda = -0.7071$

$$\mathscr{L}_{xx} = \begin{bmatrix} -0.7071 & 0 & 0 \\ 0 & 1.2929 & 1.0000 \\ 0 & 1.0000 & 3.2929 \end{bmatrix} \qquad eig(\mathscr{L}_{xx}) = \begin{bmatrix} -0.7071 \\ 0.8787 \\ 3.7071 \end{bmatrix}$$

When $\lambda = 0.7071$

$$\mathscr{L}_{xx} = \begin{bmatrix} 0.7071 & 0 & 0 \\ 0 & 2.7071 & 1.0000 \\ 0 & 1.0000 & 4.7071 \end{bmatrix} \qquad eig(\mathscr{L}_{xx}) = \begin{bmatrix} 0.7071 \\ 2.2929 \\ 5.1213 \end{bmatrix}$$

When $\lambda = -1.5858$

$$\mathscr{L}_{xx} = \begin{bmatrix} -1.5858 & 0 & 0 \\ 0 & 0.4142 & 1.0000 \\ 0 & 1.0000 & 2.4142 \end{bmatrix} \qquad eig(\mathscr{L}_{xx}) = \begin{bmatrix} -1.5858 \\ 0 \\ 2.8284 \end{bmatrix}$$

Thus, we know that the solution for this minimization problem is given by

$$\boxed{x^* = (-1.4142, 0, 0) \quad \text{and} \quad \lambda = 0.7071.}$$

in which the minmum is

$$\boxed{min\{f(x)\} = -\sqrt{2} \approx -1.4142.}$$

Problem 4

Solve the following problem

$$max \quad x_1$$

subject to

$$\begin{aligned} x_2 - (1 - x_1)^3 &\leq 0 \\ x_2 &\geq 0 \end{aligned}$$

Plot the feasible region for this problem, along with the optimal point. Draw the gradient of the constraints and the gradient of the function to be minimized. What do you observe?

Solution:
Plotting the feasible region imposed by the inequality constraints we have the following

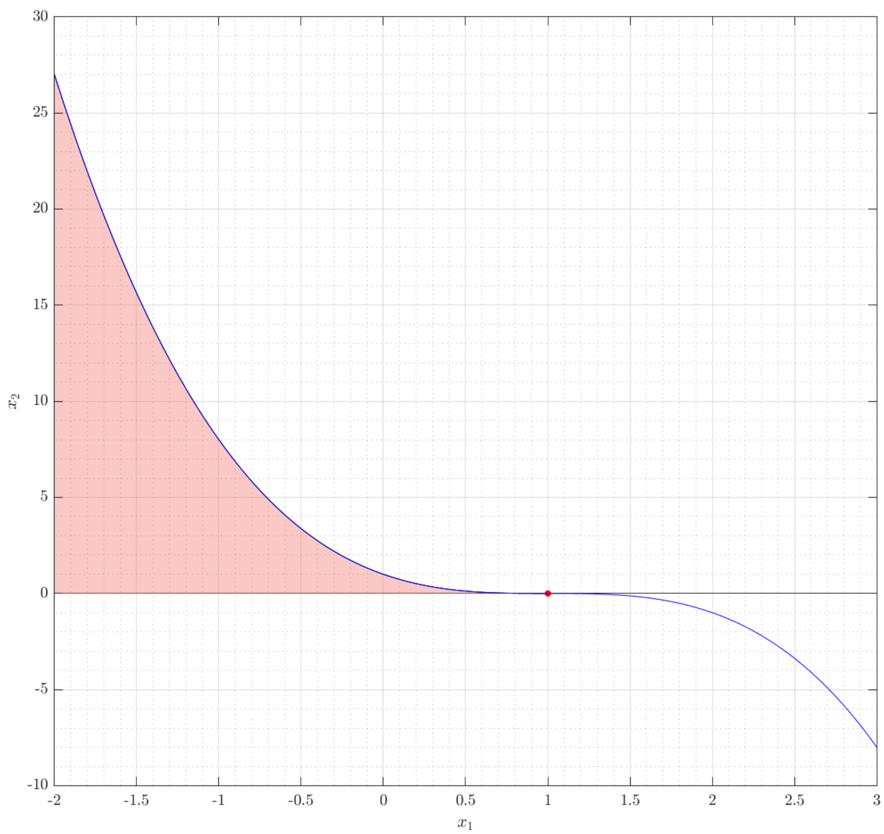

Figure 2: Feasible area of the optimization problem

If the problem function and constraints are

$$\begin{aligned} f(x) &= x_1 \\ h_1(x) &= x_2 - (1 - x_1)^3 \\ h_2(x) &= x_2 \end{aligned}$$

the gradients become

$$\begin{aligned} f'(x) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\ h_1'(x) &= \begin{bmatrix} 3(1 - x_1)^2 & 1 \end{bmatrix} \\ h_2'(x) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \end{aligned}$$

The gradient plot looks as follows.

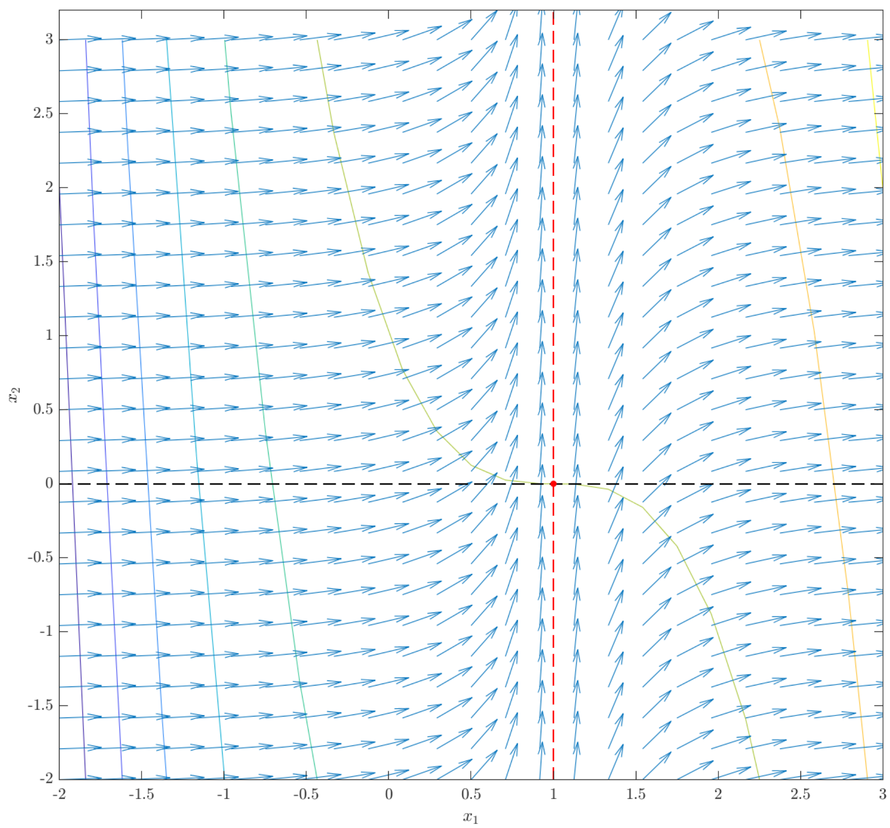

Figure 3: The gradient of the optimization problem

For this gradient plot, the black dotted line is the gradient of $f(x)$ going left to right and the red dotted line indicates the gradient of $h_2(x)$ which is going bottom to top.

From figure 2 we can see that the maximum possible value for x_1 is 1 in which $x_2 = 0$. And, at that point, from figure 3, we can observe that the gradient peaks out at that point as well. This tells us that the solution for this maximization problem is

$$\boxed{max\{f(x)\} = 1 \text{ at } x_1 = 1, \quad x_2 = 0}$$

Problem 5

Minimize

$$f(x_1, x_2) = -5x_1 - x_2$$

subject to the inequalities

$$\begin{aligned} g_1(x_1, x_2) &= -x_1 \leq 0 \\ g_2(x_1, x_2) &= 3x_1 + x_2 - 11 \leq 0 \\ g_3(x_1, x_2) &= x_1 - 2x_2 - 2 \leq 0 \end{aligned}$$

Sketch the region of feasible points in x_1, x_2 space. Check the Kuhn-Tucker necessary condition at the point which furnishes the minimum. Verify your answer using the `fmincon` command of MATLAB.

Solution:
Using Python we can plot the feasible region of the inequality constraint (refer to Problem 5: Python Code)

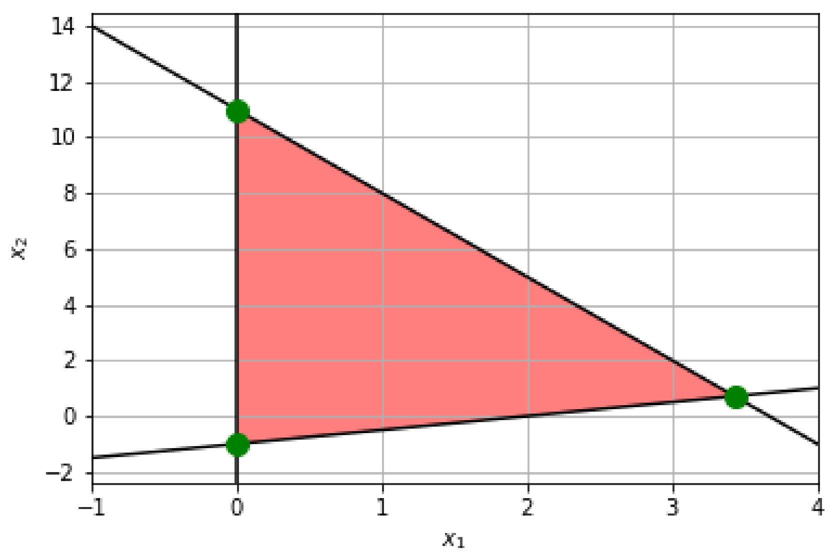

Figure 4: Feasible region of optimization problem

The Lagrangian equation of this problem is

$$\mathscr{L}(x, 1, \lambda) = (-5x_1 - x_2) + \lambda_1(-x_1) + \lambda_2(3x_1 + x_2 - 11) + \lambda_3(x_1 - 2x_2 - 2)$$

Then

$$L_x = \begin{bmatrix} -5 - \lambda_1 + 3\lambda_2 + \lambda_3 & -1 + \lambda_2 - 2\lambda_3 \end{bmatrix}.$$

If $x^* = (x_1^*, x_2^*)$ is a local minimizer, then the necessary conditions imply the existence of multiplier $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}$ such that

$$\begin{cases} -\lambda_1 + 3\lambda_2 + \lambda_3 - 5 = 0 \\ \lambda_2 - 2\lambda_3 - 1 = 0 \\ \lambda_1 \geq 0, \quad \lambda_2 \geq 0, \quad \lambda_3 \geq 0 \\ \lambda_1(-x_1) = 0 \\ \lambda_2(3x_1 + x_2 - 11) = 0 \\ \lambda_3(x_1 - 2x_2 - 2) = 0 \end{cases}$$

The possible values for these conditions can be computed using MATLAB (refer to Problem 5: MATLAB Code) which is tabulate below.

x_1	x_2	λ_1	λ_2	λ_3
0	11	-2	1	0
0	-1	-5.5	0	-0.5
3.4286	0.7143	0	1.5714	0.2857

Furthermore, we know that

$$L_{xx} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \geq 0.$$

Thus, the possible local minimizer should be

$$\boxed{x_1 = 3.4286, \quad x_2 = 0.7143, \quad \lambda_1 = 0, \quad \lambda_2 = 1.5714, \quad \lambda_3 = 0.2857.}$$

This point corresponds to the bottom right vertex of the triangular feasible region shown on figure 4. Since,

$$\begin{gathered} \lambda_1 \geq 0 \\ \lambda_2 \geq 0 \\ \lambda_3 \geq 0 \end{gathered}$$

This solution satisfies the Kuhn-Tucker condition.

Using `fmincon` from the MATLAB optimization toolbox, we are able to verify our results with some code (refer to Problem 5: MATLAB Code). And the result is the following.

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
x = 1×2
      3.4286    0.7143
```

Figure 5: Result of `fmincon`

Problem 6

The previous problem is an example of a *Linear Programming* problem. The general linear programming (LP) problem has the form

$$min \;\; c^T x$$

subject to

$$Ax \leq b$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. The MATLAB command `linprog` can be used to solve LP problems.

Consider the following transportation planning problem: A car dealer has purchased 500 cars in Seattle and 400 cars in Chicago. He then sells 200 cars to a customer in Denver, 360 to a customer in Miami and the remaining 340 cars to a customer in New York City. He wishes to determine the shipping schedule to deliver all these vehicles that will incur the minimum cost, given the freight rates (in dollars per vehicle) shown in the table below:

Seattle	\$400	\$550	\$600
Chicago	\$360	\$470	\$500
	Denver	Miami	New York

(a) Formulate the previous transportation problem as an LP problem. (Hint: Let x_1 be the number of cars to be shipped from Seattle to Denver, and let x_2 be the number of cars to be shipped from Seattle to Miami. Then $200 - x_1$ will be the number of cars to be shipped from Chicago to Denver, $360 - x_2$ will be the number of cars to be shipped from Chicago to Miami, etc).

(b) Use the `linprog` command of MATLAB to find the minimum-cost shipping schedule.

Solution:
(a) For convenience we use a shorthand term for the cities: Seattle (S), Denver (D), Chicago (C), Miami (M), New York (N). To formulate this linear programming problem we first define the variables $x = (x_1, x_2, x_3)$ in which each variable defines the number of vehicles transported from S to D, M, or N.

$$\begin{aligned} x_1 &: S \rightarrow D \\ x_2 &: S \rightarrow M \\ x_3 &: S \rightarrow N \\ 200 - x_1 &: C \rightarrow D \\ 360 - x_2 &: C \rightarrow M \\ 340 - x_3 &: C \rightarrow N \end{aligned}$$

Thus, the problem forumalation becomes

$$\boxed{\begin{aligned} f(x_1, x_2, x_3) = 400x_1 + 360(200 - x_1) + 550x_2 + 470(360 - x_2) + 600x_3 + 500(340 - x_3) \\ g_1(x_1, x_2, x_3) = x_1 + x_2 + x_3 - 500 \\ g_2(x_1) = -x_1 \leq 0 \\ g_3(x_2) = -x_2 \leq 0 \\ g_4(x_2) = -x_3 \leq 0 \\ g_5(x_1) = x_1 - 200 \leq 0 \\ g_6(x_2) = x_2 - 360 \leq 0 \\ g_7(x_3) = x_3 - 340 \leq 0 \end{aligned}}$$

(b) Simplifying the equation we get

$$f(x) = 40x_1 + 80x_2 + 100x_3 + 411200.$$

Now using MATLAB's `linprog` command we solve the solution of this linear programming problem to be as follows (refer to the code in Problem 6: MATLAB Code).

```
Solving problem using linprog.

LP preprocessing removed 3 inequalities, 1 equalities,
1 variables, and 5 non-zero elements.

 Iter      Time            Fval  Primal Infeas    Dual Infeas
    0     0.034    5.000000e+04    1.600000e+02   6.324555e+01
    2     0.052    3.080000e+04    6.000000e+01   0.000000e+00
    3     0.058    3.200000e+04    0.000000e+00   0.000000e+00

Optimal solution found.
sol = struct with fields:
    x1: 200
    x2: 300
    x3: 0
fval = 443200
exitflag =
    OptimalSolution
output = struct with fields:
         iterations: 3
    constrviolation: 0
            message: 'Optimal solution found.'
          algorithm: 'dual-simplex'
      firstorderopt: 0
             solver: 'linprog'
```

Figure 6: The displayed result from MATLAB's `linprog` command

Thus, the solution for this linear programming problem is

$$\boxed{x_1 = 200, \quad x_2 = 300, \quad x_3 = 0, \quad min\{f\} = 443200.}$$

Problem 7

The net force on an aircraft maintaining a steady rate of climb must be zero. Choosing force components parallel and normal to the flight path (see figure) one obtains the equations

$$T(V)cos(\alpha+\epsilon) - D(V,\alpha) - mgsin(\gamma) = 0$$
$$T(V)sin(\alpha+\epsilon) + L(V,\alpha) - mgcos(\gamma) = 0$$

where V is the aircraft velocity, γ the flight path angle, α the angle of attack, m the mass of the aircraft, ϵ THE angle between the thrust axis and the zero-lift axis (constant), L the lift, D the drag, and T the engine thrust. We want to find (α, V, γ) to maximize the rate of climb $f(V,\gamma) = Vsin(\gamma)$.

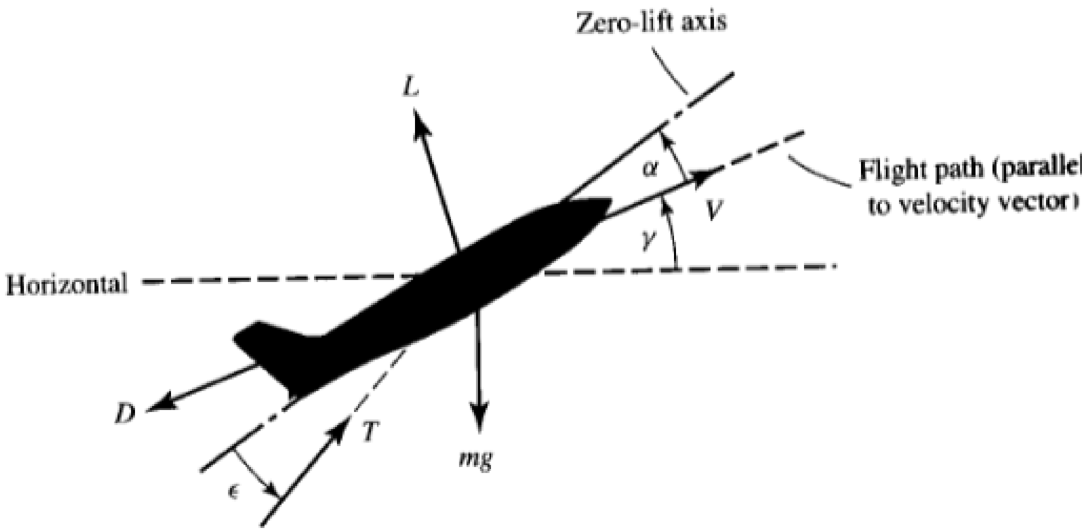

Figure 7: Force digram of aircraft maintaining steady rate of climb

(a) Write down the necessary condition for maximizing the rate of climb for an aircraft while maintaining a steady rate of climb.

(b) Using the MATLAB function `fmincon` show that for a Boeing 727 aircraft the maximum rate of climb is 37.6 ft/s and occurs at $V = 342 ft/s$, $\alpha = 6.39°$, and $\gamma = 6.31°$. The thrust, drag and lift in units of aircraft weight W (=180,000 lb) for a B723 aircraft at take-off (maximum rate of climb) are given approximately by

$$T = 0.2476 - 0.04312V + 0.008392V^2$$
$$D = (0.07351 - 0.0.08617\alpha + 1.996\alpha^2)V^2$$
$$L = (0.1667 + 6.231\alpha - 21.65[max(0, \alpha - 0.2094)]^2)V^2]$$

where V is the velocity in units of $\sqrt{gl}$, $l = 2W/(\rho g S)$, $\rho = 0.002203$ slugs/ft^3 the density of air at sea level, and $S = 1560\text{ft}^2$ the wing span. The angle of attack α is in radians and $\epsilon = 0.0349$ rad.

Solution:
(a) When $x = (x_1, x_2, x_3) = (V, \gamma, \alpha)$ the problem statement of this optimization problem is

$$\begin{gathered} f(x_1, x_2) = x_1 sin(x_2) \\ g_1(x_1, x_2, x_3) = T(x_1)cos(x_3 + \epsilon) - D(x_1, x_3) - mgsin(x_2) = 0 \\ g_2(x_1, x_2, x_3) = T(x_1)sin(x_3 + \epsilon) + L(x_1, x_3) - mgcos(x_2) = 0 \\ \epsilon = \text{const.} \end{gathered}$$

The Lagrangian of this problem becomes

$$\begin{aligned} L(x, \mu, \lambda) = \mu(x_1 sin(x_2)) &+ \lambda_1(T(x_1)cos(x_3 + \epsilon) - D(x_1, x_3) - mgsin(x_2)) \\ &+ \lambda_2(T(x_1)sin(x_3 + \epsilon) + L(x_1, x_3) - mgcos(x_2)). \end{aligned}$$

Then we take the first derivative of this

$$\boxed{L_x = \begin{bmatrix} \mu sin(x_2) + \lambda_1\big(T_{x_1}cos(x_3 + \epsilon) - D_{x_1}\big) + \lambda_2\big(T_{x_1}sin(x_3 + \epsilon) + L_{x_1}\big) \\ \mu x_1 cos(x_2) + \lambda_1\big(-mgcos(x_2)\big) + \lambda_2\big(mgsin(x_2)\big) \\ \lambda_1\big(-Tsin(x_3 + \epsilon) - D_{x_3}\big) + \lambda_2\big(Tcos(x_3 + \epsilon) + L_{x_3}\big) \end{bmatrix}^T}$$

From this, we know that the first order necessary condition is that the 3 elements of L_x should equal 0 while $\lambda = (\lambda_1, \lambda_2) \in \mathbb{R}^2$ and $\mu > 0$.

(b) For this part, we will have to first clarify the values that we are going to use. We are given that $\epsilon := 0.0349$ [rad] and from the problem statement we know that the thrust, drag, and lift equations are given as dimensionless values. Therefore, for the objective function f and constraint functions g, we get a dimensionless value and need not mg in the equation. For the final output of the velocity we will have to multiply that by

$$\sqrt{gl} = \Big(\frac{2W}{\rho S}\Big)^{1/2} = 323.65 \quad ft/s$$

to get the proper velocity.

Now let us compute the maximum rate of climb using `fmincon` with nonlinear constraints (refer to the code in Problem 7: MATLAB Code). The output are as follows.

```
x = 1x3
        1         2         3
   1    1.0575    75.5084   0.1115

fval = -0.1162
exitflag = 1
output = struct with fields:
         iterations: 15
          funcCount: 67
    constrviolation: 1.3323e-15
           stepsize: 1.9018e-08
          algorithm: 'interior-point'
      firstorderopt: 2.9392e-09
       cgiterations: 0
            message: '↵Local minimum found that satisfies the constraints.↵↵Optimization comp
       bestfeasible: [1x1 struct]

lambda = struct with fields:
         eqlin: [0x1 double]
      eqnonlin: [2x1 double]
       ineqlin: [0x1 double]
         lower: [3x1 double]
         upper: [3x1 double]
    ineqnonlin: [0x1 double]
grad = 3x1
     -0.1099
     -1.0511
           0

hessian = 3x3
      0.1677   -1.2269    0.4391
     -1.2269    9.0896   -4.0783
      0.4391   -4.0783    8.1814
```

Figure 8: `fmincon` output results

Hence, we can confirm the results as follows.

$$\boxed{\begin{aligned} \lambda_1 &= -1.0646, \qquad \lambda_2 = -0.0641 \\ V^* &= 1.0575 \times \sqrt{gl} = 1.0575 \times 323.65 = 342.2599 \quad [ft/s] \\ \gamma^* &= mod(75.5084, 2\pi) \times \frac{180}{\pi} = 6.3126 \quad [deg] \\ \alpha^* &= 0.1115 \times \frac{180}{\pi} = 6.3885 \quad [deg] \\ max\{f\} &= 0.1162 \times \sqrt{gl} = 0.1162 \times 323.65 = 37.6081 \quad [ft/s] \end{aligned}}$$

```
                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       4   -9.983342e+00    7.908e+03    9.773e+01
    1       8    3.952802e+00    1.391e+03    1.503e+03    1.105e+02
    2      12   -5.660664e+00    1.319e+02    4.596e+02    3.109e+01
    3      16   -1.964019e-01    2.732e+01    1.234e+02    2.842e+00
    4      20   -2.556686e+00    4.759e+00    1.531e+01    5.027e+00
    5      24    3.040139e-01    6.662e-01    6.387e+00    1.995e+00
    6      28   -1.071559e-01    1.135e-01    1.921e+00    3.482e-01
    7      32   -1.072534e-01    3.477e-03    4.168e-01    6.083e-02
    8      36   -1.077727e-01    4.955e-05    5.532e-01    7.007e-03
    9      40   -1.102374e-01    1.131e-03    4.479e-01    2.551e-02
   10      45   -1.157896e-01    1.430e-03    1.125e-01    9.987e-02
   11      50   -1.162226e-01    5.358e-05    2.750e-02    3.486e-02
   12      55   -1.162486e-01    1.845e-06    2.020e-03    9.887e-03
   13      59   -1.162488e-01    1.890e-06    3.834e-05    9.116e-04
   14      63   -1.162486e-01    5.796e-10    2.432e-06    1.528e-05
   15      67   -1.162486e-01    1.332e-15    2.939e-09    1.902e-08

Feasible point with lower objective function value found.


Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

Figure 9: `fmincon` output of iterations

Appendix

8.1 Problem 2: MATLAB Code

```
% Problem 2 MATLAB code
% Tomoki Koike
clear all; close all; clc;  % housekeeping commands
%%
% Declare and solve necessary conditions
syms x_1 x_2 lambda_1 lambda_2 lambda_3;
assume(lambda_1 >= 0); assume(lambda_1, 'real');
assume(lambda_2 >= 0); assume(lambda_2, 'real');
assume(x_1, 'real');
assume(x_2, 'real');
assume(lambda_3, 'real');
eqn1 = 2*x_1 + 2*lambda_1*x_1 + 3*lambda_3*x_1^3 == 0;
eqn2 = -1 + 2*lambda_1*x_2 + lambda_2 + lambda_3 == 0;
eqn3 = lambda_1 * (x_1^2 + x_2^2 - 1) == 0;
eqn4 = lambda_2 * (x_2 - 2) == 0;
eqn5 = x_1^3 + x_2 - 1 == 0;
%%
eqn = [eqn1, eqn2, eqn3, eqn4, eqn5];
sol = solve(eqn, [x_1, x_2, lambda_1, lambda_2, lambda_3]);
%%
sol.x_1 = double(sol.x_1);
sol.x_2 = double(sol.x_2);
sol.lambda_1 = double(sol.lambda_1);
sol.lambda_2 = double(sol.lambda_2);
sol.lambda_3 = double(sol.lambda_3);
%%
sol_table = struct2table(sol)
sol_array = table2array(sol_table);
%%
% Check 2nd order sufficient condition
x1 = sol_array(2, 1);
l1 = sol_array(2, 3);
l3 = sol_array(2, 5);
Lxx = [2 + 2*l1 + 6*l3*x1, 0; 0, 2*l1]
eig(Lxx)
%%
% Check 2nd order sufficient condition
x1 = sol_array(4, 1);
l1 = sol_array(4, 3);
l3 = sol_array(4, 5);
```

```
Lxx = [2 + 2*l1 + 6*l3*x1, 0; 0, 2*l1]
eig(Lxx)
```

8.2 Problem 3: MATLAB Code

```
% Problem 3 MATLAB code
% Tomoki Koike
clear all; close all; clc;  % housekeeping commands
%%
% Declare and solve necessary conditions
syms x_1 x_2 x_3 lambda
assume(lambda, 'real');
assume(x_1, 'real');
assume(x_2, 'real');
assume(x_3, 'real');
eqn1 = 1 + lambda*x_1 == 0;
eqn2 = (lambda + 2)*x_2 + x_3 == 0;
eqn3 = x_2 + (lambda + 4)*x_3 == 0;
eqn4 = 0.5*(x_1^2 + x_2^2 + x_3^2) - 1 == 0;
%%
eqn = [eqn1, eqn2, eqn3, eqn4];
sol = solve(eqn, [x_1, x_2, x_3, lambda]);
%%
sol.x_1 = double(sol.x_1);
sol.x_2 = double(sol.x_2);
sol.x_3 = double(sol.x_3);
sol.lambda = double(sol.lambda);
%%
sol_table = struct2table(sol)
sol_array = table2array(sol_table);
%%
% Check 2nd order sufficient condition
for i = 1:6
    fprintf("Iteration %i", i);
    l = sol_array(i, 4);
    Lxx = [l, 0, 0; 0, l+2, 1; 0, 1, l+4]
    eig(Lxx)
end
%%
% The answer to the minimization problem is
fx = x_1 + x_2^2 + x_2 * x_3 + 2*x_3^2;
subs(fx, {x_1, x_2, x_3}, {sol_array(4,1), ...
    sol_array(4,2), sol_array(4,3)})
```

8.3 Problem 5: MATLAB Code

```
% Problem 5 MATLAB code
% Tomoki Koike
clear all; close all; clc;  % housekeeping commands
%%
% Declare and solve necessary conditions
syms x_1 x_2 lambda_1 lambda_2 lambda_3
assume(lambda_1, 'real');
assume(lambda_2, 'real');
assume(lambda_3, 'real');
assume(x_1, 'real');
assume(x_2, 'real');
eqn1 = -lambda_1 + 3*lambda_2 + lambda_3 - 5 == 0;
eqn2 = lambda_2 - 2*lambda_3 - 1 == 0;
eqn3 = lambda_1 * -x_1 == 0;
eqn4 = lambda_2 * (3*x_1 + x_2 - 11) == 0;
eqn5 = lambda_3 * (x_1 - 2*x_2 - 2) == 0;
%%
eqn = [eqn1, eqn2, eqn3, eqn4, eqn5];
sol = solve(eqn, [x_1, x_2, lambda_1, lambda_2, lambda_3]);
sol.x_1 = double(sol.x_1);
sol.x_2 = double(sol.x_2);
sol.lambda_1 = double(sol.lambda_1);
sol.lambda_2 = double(sol.lambda_2);
sol.lambda_3 = double(sol.lambda_3);
sol_table = struct2table(sol)
sol_array = table2array(sol_table);
%%
% Solving using fmincon
f = @(x) -5*x(1) -x(2);
x0 = [1, 1];
A = [-1, 0; 3, 1; 1, -2];
b = [0; 11; 2];
x = fmincon(f, x0, A, b)
```

8.4 Problem 5: Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from sympy.solvers import solve
```

```
from sympy import Symbol

def g2(x):
    return -2 * x + 11

def g3(x):
    return 0.5 * x - 1

G2 = np.vectorize(g2)
G3 = np.vectorize(g3)

x = Symbol('x')
x1 = 0
x2 = solve(g2(x) - g3(x))

y1 = G2(0)
y2 = G3(0)
y3 = G2(x2)

xr = np.linspace(-1, 4, 100)
y2r = G2(xr)
y3r = G3(xr)

plt.plot(xr, y2r, '-k')
plt.plot(xr, y3r, '-k')
temp = np.linspace(-3, 15)
plt.plot(np.zeros(temp.shape), temp, '-k')

plt.plot(x1, y1, 'go', markersize=10)
plt.plot(x1, y2, 'go', markersize=10)
plt.plot(x2, y3, 'go', markersize=10)

plt.fill([x1, x1, *x2], [y1, y2, y3], 'red', alpha=0.5)
plt.xlim(-1, 4)
plt.ylim(-2.4, 14.5)
plt.grid(True)
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.savefig('p5_region.png')
plt.show()
```

8.5 Problem 6: MATLAB Code

```
% Problem 6 MATLAB code
% Tomoki Koike
clear all; close all; clc;  % housekeeping commands
%%
syms x_1 x_2 x_3
f = (400*x_1 + 360*(200 - x_1) + 550*x_2 ...
    + 470*(360 - x_2) + 600*x_3 + 500*(340 - x_3));
f_s = simplify(f)
%%
% Initialize LP problem
x1 = optimvar('x1');
x2 = optimvar('x2');
x3 = optimvar('x3');
prob = optimproblem('Objective', 40*x1 + 80*x2 + 100*x3 + 411200, ...
    'ObjectiveSense', 'minimize');
prob.Constraints.c1 = x1 + x2 + x3 == 500;
prob.Constraints.c2 = x1 >= 0;
prob.Constraints.c3 = x2 >= 0;
prob.Constraints.c4 = x3 >= 0;
prob.Constraints.c5 = 200 - x1 >= 0;
prob.Constraints.c6 = 360 - x2 >= 0;
prob.Constraints.c7 = 340 - x3 >= 0;
options = optimoptions(prob);
options.Display = 'iter';
%%
% Solve
[sol, fval, exitflag, output] = solve(prob, 'Options', options, ...
    'Solver','linprog')
```

8.6 Problem 7: MATLAB Code

```
% Problem 7 MATLAB code
% Tomoki Koike
clear all; close all; clc;  % housekeeping commands
%%
% Define symbolic values
syms x_1 x_2 x_3 m g epsilon
assume(m, {'real', 'positive'});
assume(g, {'real', 'positive'});
assume(epsilon, {'real', 'positive'});

```

```matlab
% Define Lagrangian parameters
syms lambda_1 lambda_2 mu
assume(lambda_1, 'real');
assume(lambda_2, 'real');
assume(mu, {'real', 'positive'});

% Define symbolic functions
syms f(x_1, x_2) g_1(x_1,x_2,x_3) g_2(x_1,x_2,x_3)
syms T(x_1) D(x_1,x_3) L(x_1,x_3)
f(x_1,x_2) = x_1 * sin(x_2);
g_1(x_1,x_2,x_3) = T(x_1)*cos(x_3 + epsilon) - D(x_1,x_3) - m*g*sin(x_2);
g_2(x_1,x_2,x_3) = T(x_1)*sin(x_3 + epsilon) + L(x_1,x_3) - m*g*cos(x_2);
%%
syms l(x_1, x_2, x_3, mu, lambda_1, lambda_2)
l(x_1, x_2, x_3, mu, lambda_1, lambda_2) = (mu*f(x_1,x_2) + ...
    lambda_1*g_1(x_1,x_2,x_3) + lambda_2*g_2(x_1,x_2,x_3));

% 1st order derivative of the Lagrangian
l_x = [diff(L, x_1); diff(L, x_2); diff(L, x_3)]
%%
% Solving optimization problem
options = optimoptions('fmincon','Display','iter', ...
    'Algorithm','interior-point');
problem.options = options;
problem.solver = 'fmincon';
problem.objective = @(x) -x(1)*sin(x(2));
problem.x0 = [100,0.1,0.1];
problem.nonlcon = @nl_constraints;
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(problem);
%%
function [c, ceq] = nl_constraints(x)
    epsilon = 0.0349;

    % dimensionless equations of thrust, drag, and lift
    T = @(x) 0.2476 - 0.04312*x(1) + 0.008392*x(1)^2;
    D = @(x) (0.07351 - 0.08617*x(3) + 1.996*x(3)^2)*x(1)^2;
    L = @(x) (0.1667 + 6.231*x(3) - 21.65*max(0, x(3) - 0.2094)^2)*x(1)^2;

    c = [];
    ceq = [T(x)*cos(x(3) + epsilon) - D(x) - sin(x(2));
           T(x)*sin(x(3) + epsilon) + L(x) - cos(x(2))];
end
```