TOML Programming exercises

Sergi Palomas

April 25, 2021

Exercise 1

$$\begin{aligned} \min \quad & e^{x_1} \cdot (4x_1^2 + 2x_2^2 + 4x_1 \cdot x_2 + 2x_2 + 1) \\ \text{s.t.} \quad & C1 \ : \ x_1 \cdot x_2 - x_1 - x_2 \leq -1.5 \\ & C2 \ : \ -x_1 \cdot x_2 \leq 10 \\ \text{var} \quad & x_1, x_2 \end{aligned} \tag{1.1}$$

a) Identify whether is convex or not.

The next plot shows the value of the cost function with respect to variables $x_1 and x_2$.

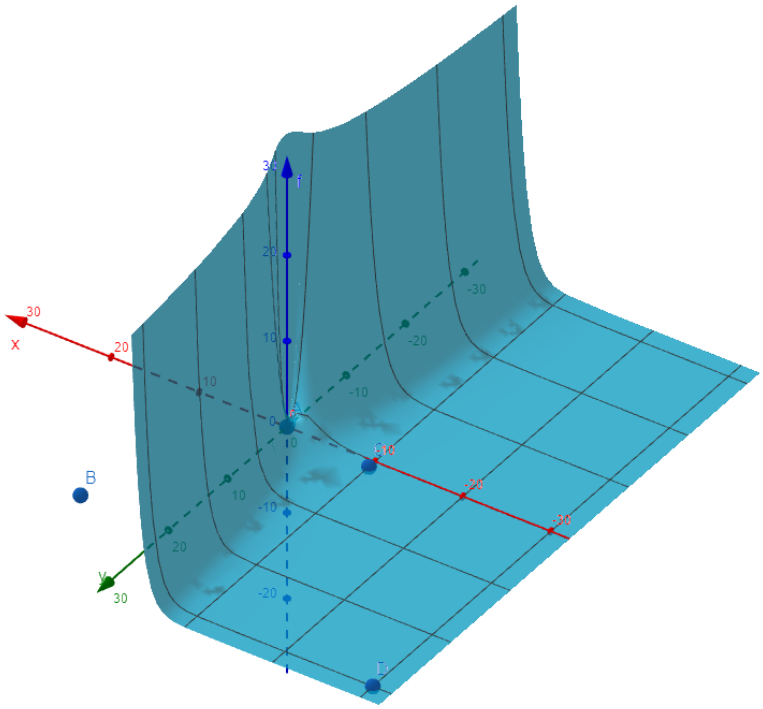


Figure 1: Cost function from exercise 1 equation 1.1

From the plot, we can see that the function is monotonically increasing. So it is convex.

Plot 2 shows the constraints of the problem to optimize. Blue for C1 and red for C2. It also where the different initial guesses fail. None of them starts being a feasible solution (C1 ∩ C2)

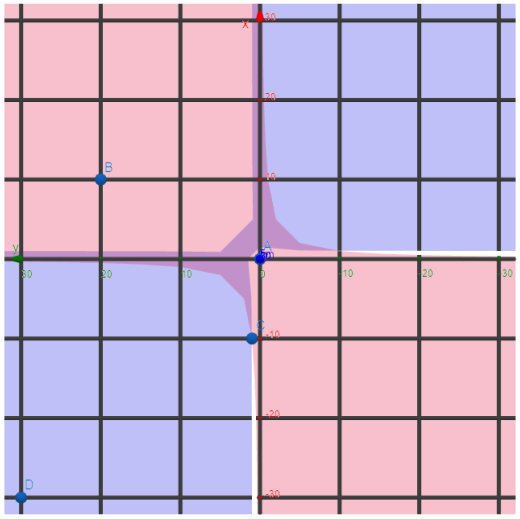


Figure 2: Constraints of ex1

b) Find the minimum. Use as initial points x_0: [0,0], [10,20], [-10,1], [-30,-30] and explain the difference in the solutions.

Initial Guess = [0, 0]

The method starts from an unfeasible point (fails on the first constraint in equation 1.1 as $0 \cdot 0 - 0 - 0 \leq -1.5$ is false). As we can see in the plot 3, at the first iteration the algorithm jumps to $x_1 = 1.25$ and $x_2 = 0.25$. Which has a cost of 31.85. After 3 more iterations the cost is reduced to 3.59 (almost 10 times better). Finally, we reach optimality at iteration 17 with a cost of 0.023 and values $x_1 = -9.55$ and $x_2 = 1.05$. The total time to converge is 14 ms:

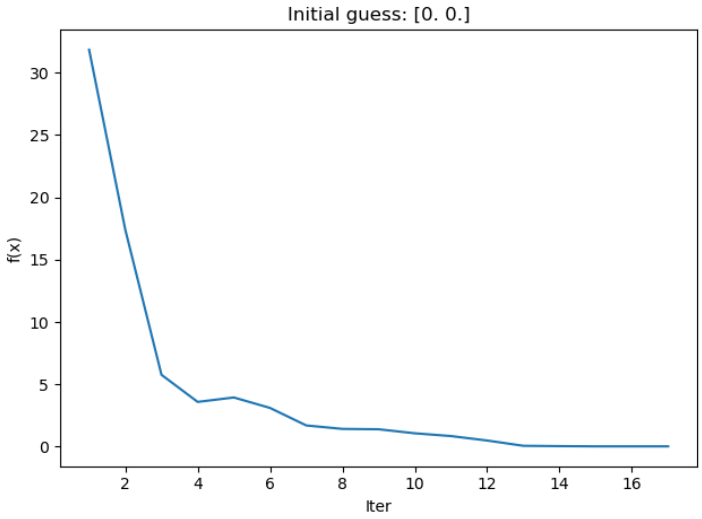


Figure 3: Cost while minimizing from starting point [0, 0]

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.023550379624174556
            Iterations: 17
            Function evaluations: 54
            Gradient evaluations: 17
optimal var: x1 =  -9.547405025104304  x2 =  1.0474050251042841
Time to convergence:  0.014383316040039062
```

Figure 4: Result from starting point [0, 0]

Plot 5 shows which points has the method chosen from the initial guess [0, 0] to the end of the minimization [-9.55, 1.05]. Similarly, plot 6 shows the cost function of each of these points and how we are effectively reducing the value of the cost function at each step..

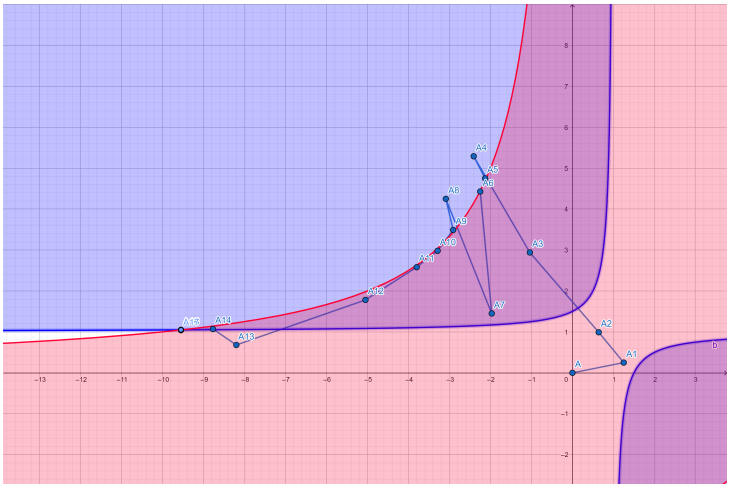


Figure 5: Points used at each iteration of the SLSQP method

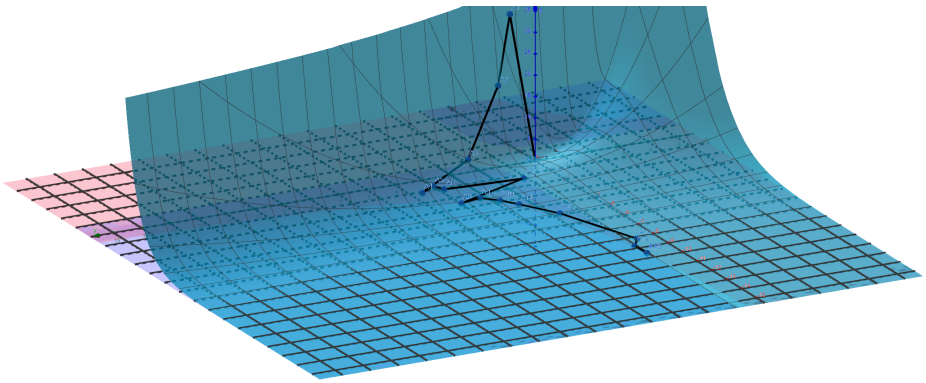


Figure 6: Reducing the cost of the function at each step of the SLSQP method

Initial Guess = [10, 20]

Again, the initial guess is a non feasible solution (the first constraint fails since 170 <= -1.5 is false). As we can see in the plot 7, the first iteration of the SLSQP method gives a cost of 0 (with $x_1 = -18350099$ and $x_2 = 12377862$, which is again, not feasible). In the next step, the cost grows up to 1530.77 and then it begins to minimize the cost function at each iteration. I'm not sure what is causing this behaviour. The optimization finishes successfully (plot 8 after 33ms and 32 iterations with values $x_1 = -9.55$ and $x_2 = 1.05$. Which give a cost of 0.024 but is NOT feasible.

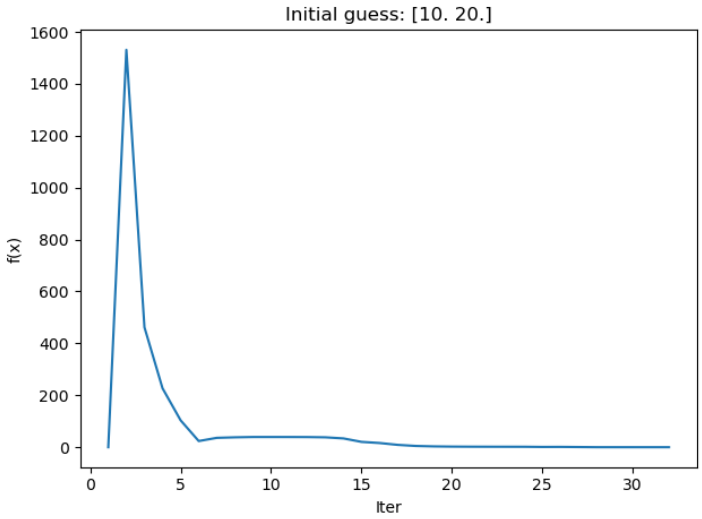


Figure 7: Cost while minimizing from starting point [10, 20]

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.023550379624116897
            Iterations: 32
            Function evaluations: 105
            Gradient evaluations: 32
optimal var: x1 =  -9.54740502510709  x2 =  1.0474050251070897
Time to convergence:  0.04562830924987793
```

Figure 8: Result from starting point [10, 20]

Initial Guess = [-10, 1]

This initial condition, as seen from the previous runs, is very close to the optima. Therefore, the number of steps before reaching an optima is 3 (the first step is already giving the same cost as the last step from the previous examples). This can be seen in Figure 9. The total time to convergence, as seen in plot is 7ms with a cost of 0.024 and with values $x_1 = -9.55$ and $x_2 = 1.05$. See Figure 10 for more details.

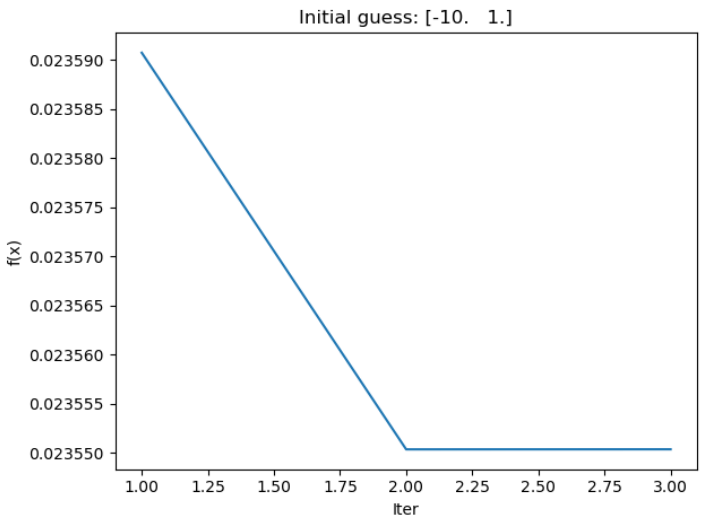


Figure 9: Cost while minimizing from starting point [-10, 1]

```
Inigial guess is: [-10, 1]
Iter    X1          X2         f(X)
   1   -9.545455    1.045455    0.023591
   2   -9.547405    1.047405    0.023550
   3   -9.547405    1.047405    0.023550
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.023550379624174944
            Iterations: 3
            Function evaluations: 10
            Gradient evaluations: 3
optimal var: x1 =  -9.547405025104283  x2 =  1.0474050251042832
Time to convergence:  0.006943702697753906
```

Figure 10: Result from starting point [-10, 1]

c) Give as input the Jacobian.

The Jacobian for the cost function of problem 1.1 is:

$$\begin{matrix} \frac{\partial}{\partial x_1} & e^{x_1} \cdot (4x_1^2 + 4x_1x_2 + 2x_2^2 + 2x_2 + 1)) \\ \frac{\partial}{\partial x_2} & e^{x_1} \cdot (4x_1 + 4x_2 + 2) \end{matrix} \tag{1.2}$$

Which gives, when using the initial guess [0, 0], the same cost function value as in the previous execution but, as we are the Jacobian, fewer function evaluations are needed. The advantage of using exact analytical derivatives rather than finite difference approximations to the derivative is that the latter requires many function evaluations and this slows down the optimization process. Furthermore, the inaccuracy of such approximate derivatives can cause the algorithm to require more steps to converge to an optimal solution and thus, run more slowly.

As shown in the plot 11, the function evaluations has drop from 54 to 20. With the number of iterations and the time to convergence remaining equal.

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.023550379624174556
            Iterations: 17
            Function evaluations: 54
            Gradient evaluations: 17
optimal var: x1 =  -9.547405025104304  x2 =  1.0474050251042841
Time to convergence:  0.022686004638671875
Constraint 1:  -1.5  <= -1.5
Constraint 2. 10.0 <= 10
Result is feasible

----------------------



Using Jacobian
Inigial guess is: [0. 0.]
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.023550379624174583
            Iterations: 17
            Function evaluations: 20
            Gradient evaluations: 17
optimal var: x1 =  -9.5474050251043  x2 =  1.0474050251042981
Time to convergence:  0.022686004638671875

----------------------
```

Figure 11: Improvement when using the Jacobian matrix with IG [0, 0]

Exercise 2

$$
\begin{aligned}
\min \quad & x_1^2 + x_2^2 \\
\text{s.t.} \quad & C1 : -x_1 - x_2 + 1 \leq 0 \\
& C2 : -x_1^2 - x_2^2 + 1 \leq 0 \\
& C3 : -9x_1^2 - x_2^2 + 9 \leq 0 \\
& C4 : -x_1^2 + x_2 \leq 0 \\
& C5 : x_1 - x_2^2 \leq 0 \\
& bnd : 0.5 \leq x_1 \\
\text{var} \quad & x_1, x_2
\end{aligned}
\tag{2.1}
$$

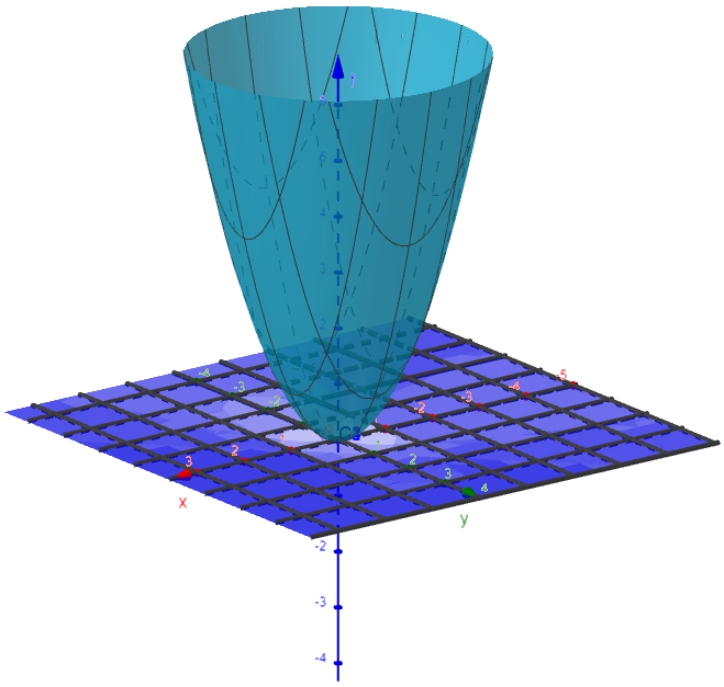
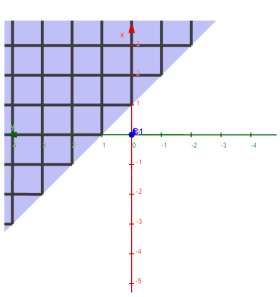
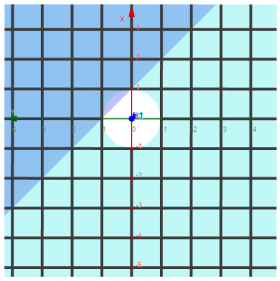


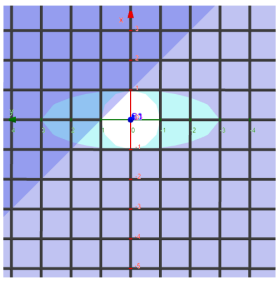
Figure 12: Exercise 2 cost function



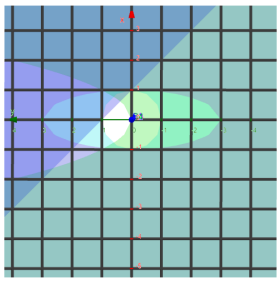
(a) Constraint 1



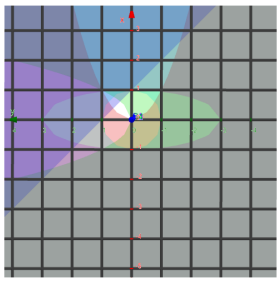
(b) Constraint 1-2



(c) Constraint 1-3



(d) Constraint 1-4



(e) Constraint 1-5

Figure 13: Constraints of Exercise 2. Where a) shows only constraint C1. b) shows constraints C1 and C2. c) shows constraints C1, C2 and C3...

a) Identify weather is convex or not. The Hessian matrix is positive-definite:

$$H = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \tag{2.2}$$

Therefore, we know that the function is strictly convex and has only one minimum. We can also see this int image 12

b) Propose an initial point that is not feasible and an initial point that is feasible and check what happens with SLSQP as method.

I have chosen [2, 2] and [0, 0] for the feasible and infeasible Initial guesses as both are equidistant from the optimal solution values [1, 1].

Feasible Initial Guess = [2, 2] The optimization finishes after only 6 Iterations. Giving a function value of 2 ($x_1 = 1, x_2 = 1$) and with 18 function evaluations and 6 gradient evaluations. The total time to convergence is 8.98ms.

Infeasible Initial Guess = [0, 0] This point breaks C1, C2 and C3. The method failed to converge (Positive directional derivative for linesearch). Which makes complete sense as this is the minimum of the cost function. The optimizer got into a position where it did not manage to find a direction where the value of the objective function decreases (fast enough), but could also not verify that the current position is a minimum (as it is not feasible). So we give another try with: **Infeasible Initial Guess [0, 2]** The optimizer finished successfully with a cost function of 2, 20 iterations, 79 function evaluations, 18 gradient evaluations and 20.82 ms.

c) Repeat giving as input the Jacobian. Check the convergence time (or number of steps). For the feasible IG, the difference is minimal. Only the Function evaluations has decreased to one third. The function value is and the time to reach convergence is the same as without the Jacobian.

For the infeasible IG, the difference is more significative. The number of iterations has gone from 20 to 6, the function evaluations from 79 to 15 and the Gradient from 18 to 6. The result is very slightly different (function value is 1.99 in $[x_1 = 0.99 and x_2 = 0.99]$. The method is now 2x faster (time to converge has dropped from 20.82 to 8.94 ms)

Exercise 4

$$\begin{aligned} \min \quad & x^2 + 1 \\ \text{s.t.} \quad & C1 : (x-2)(x-4) \leq 0 \\ \text{var} \quad & x \end{aligned} \tag{4.1}$$

We have to minimize the convex function 4.1. The problem is geometrically showed in plot **??**: The cost function is convex in all its domain and the constraint C1 (blue region in the image) restricts the values of x such that 2 <= x <= 4. Since we are minimizing, we already see that the optimal value will be $f(2) = 2^2 + 1 = 5$.

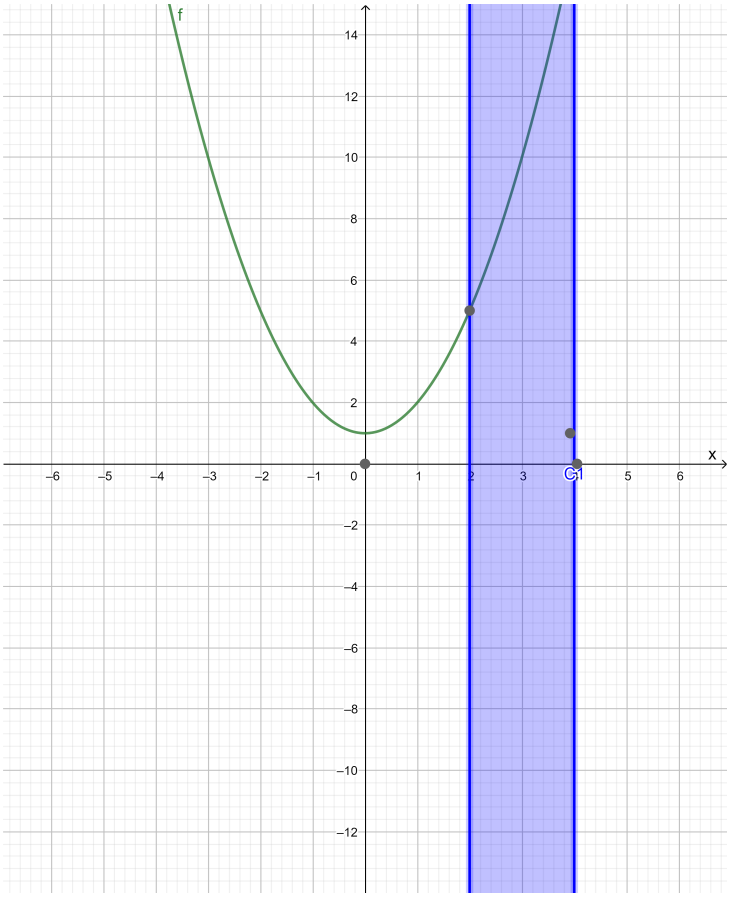


Figure 14: Cost function from exercise 4 equation 4.1

The problem was solved with the CVX toolbox. The solution is shown in plot 15:

```
 Solving exercise 4

time to converge =  0.022335052490234375
status: optimal
optimal value p* =  4.999999987444469
optimal var: x1 =  1.9999999968611173
optimal dual variables lanbda1 =  2.0000322081789013
```

Figure 15: Result of exercise 4

Exercise 5

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & C1 \,:\, (x_1 - 1)^2 + (x_2 - 1)^2 \leq 1 \\ & C2 \,:\, (x_1 - 1)^2 + (x_2 + 1)^2 \leq 1 \\ \text{var} \quad & x_1, x_2 \end{aligned} \tag{5.1}$$

As shown in Exercise 2, this cost function is convex. We showed it graphically in plot 12. Plot 16 shows the feasible solution space to satisfy C1 (green), C2 (red) and the cost function (blue).

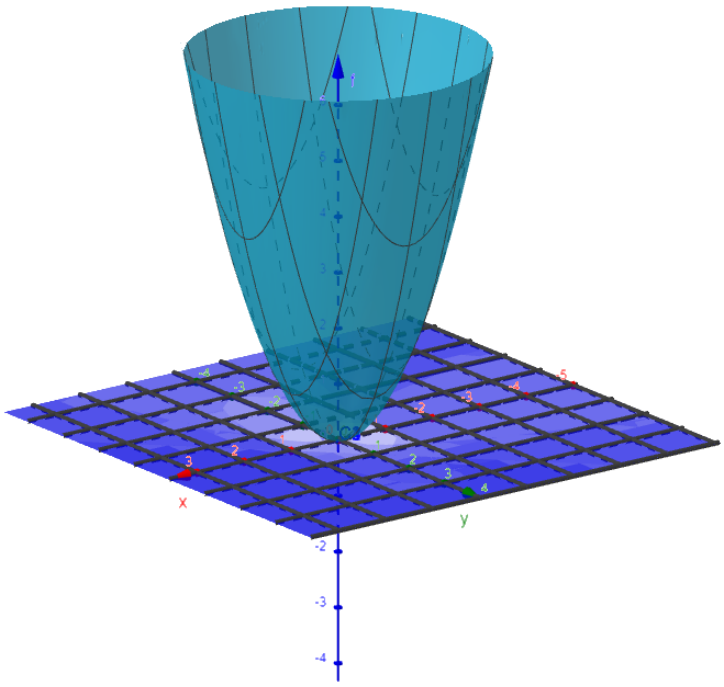


Figure 16: Constraints and cost function from exercise 5 equation 5.1

There is only one value of (x_1, x_2) which satisfies both constraints. That is $(1, 0)$. Which gives and optimal value of $f(1, 0) = 1$ The problem was solved using the CVX toolbox and producing the result shown in plot 17

```
 Solving exercise 4

time to converge =  0.022335052490234375
status: optimal
optimal value p* =  4.999999987444469
optimal var: x1 =  1.9999999968611173
optimal dual variables lanbda1 =  2.0000322081789013
```

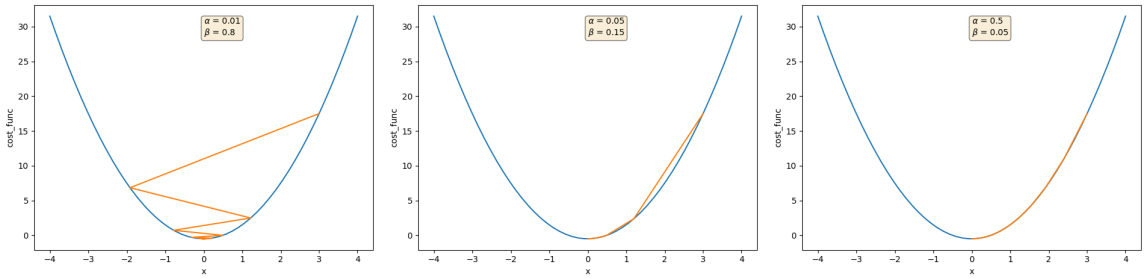
Figure 17: Result of exercise 5

Exercise 6

The first function that we use for the Gradient Descent Method using Backtracking Line Search is:

$$f(x) : \quad 2x^2 - 0.5 \tag{6.1}$$

We will try different values for the parameters of α (percentage we want to decrease of the prediction based on linear interpolation) and β (how fast do we decrease the step size in the Backtracking). So we can see the effect they have on the Gradient Descent Method:



(a) (b) (c)

Figure 18: Gradient Descend Method using Backtracking Line Search for unconstrained problem 6.1 with different values of α and β

Then, we do the same but for the non-convex function 6.2 where there is a local and a global minimum:

$$f(x) : \quad 2x^4 - 4x^2 + x - 0.5 \tag{6.2}$$

And the results are shown in plots 19. As we see, if the jump we make is not big enough (small value for β) we can easily fall in a local minimum. However, it is always a trade-off with the performance. Nevertheless, high values for β do not guarantee finding always the global minimum either.

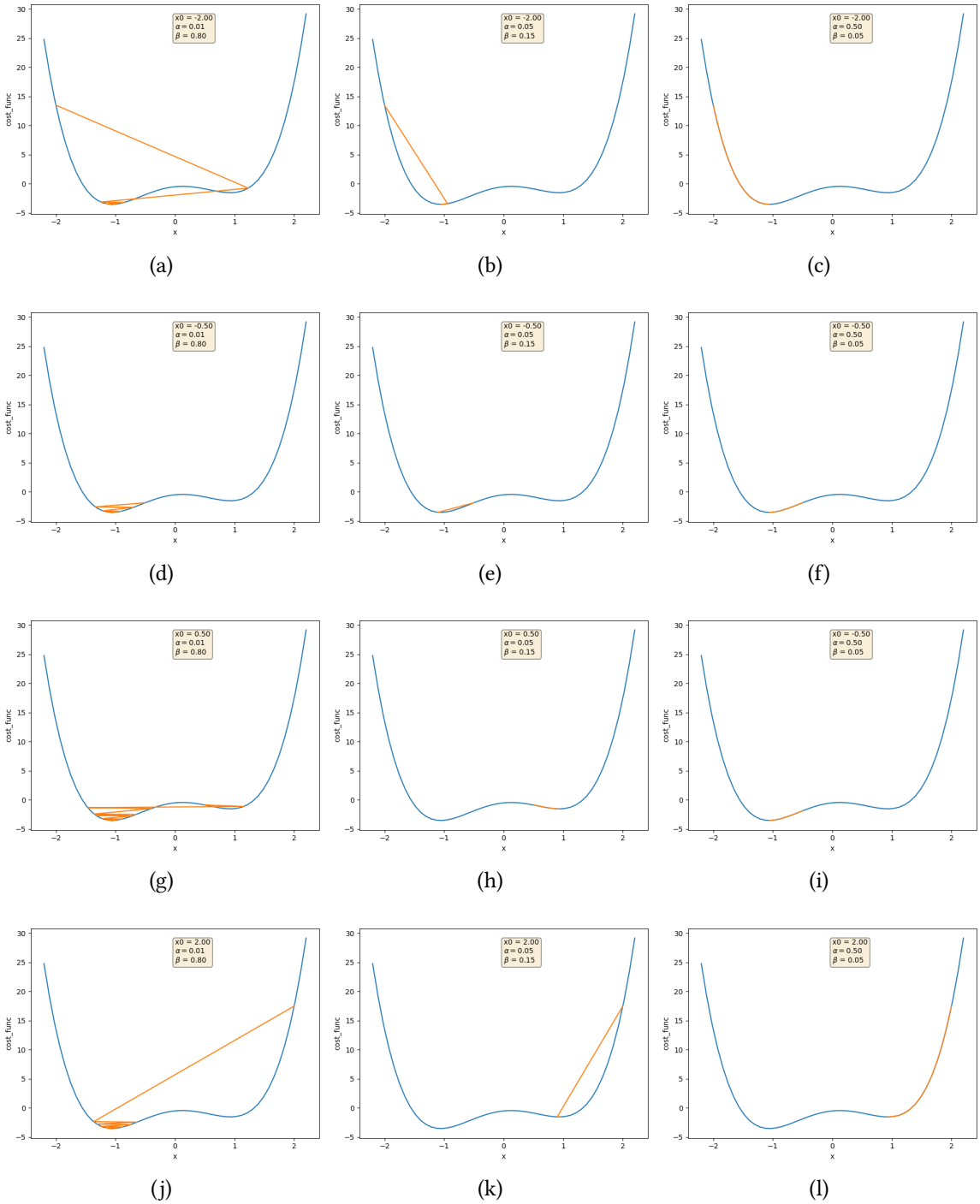


Figure 19: Gradient Descend Method using Backtracking Line Search for unconstrained problem 6.2 with different values of α, β and starting points (x0)

Exercise 7

Here we solve a Network Utility problem with 3 sources and 5 links with capacity (C1..C5) = (1,2,1,2,1). Source 1 traverses link 1 and 2. Source 2 traverses link 2 and source 3 traverses link 1 and 5. Better seen in figure 20:

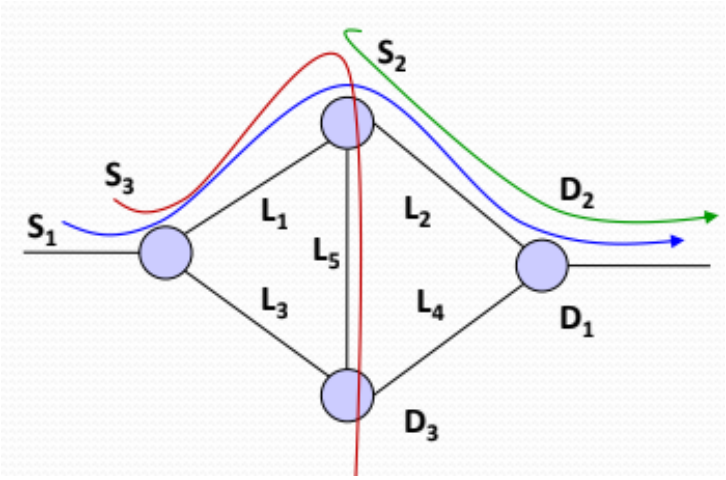


Figure 20: Statement Exercise 7

The problem can be formally stated as:

$$\begin{aligned} \max \quad & \sum_{s \in S} log(x_s) \\ \text{s.t.} \quad & Ax \leq C^T \\ & x \geq 0 \end{aligned} \tag{7.1}$$

With A matrix with coefficients:
a_ij = 1 if source j lies on link $i \in L_i$
a_ij = 0 otherwise

And C a vector with the values of the capacity of each link.

The problem was solved with the CVX toolbox achieving an optimal value of -0.955 in 10 ms as shown in figure 21

```
Solving exercise 4

Time to converge =  0.009927034378051758
status: optimal
optimal value p* =  -0.9547712584084567
optimal var: x1 =  0.4226496768137118  x2 =  1.577350318148986  x3 =  0.5773503215854673
optimal dual variables λ1 1.7320508173896876
optimal dual variables λ2 =  0.6339745957524683
optimal dual variables λ3 =  5.2143189508284225e-09
```

Figure 21: Result Exercise 7

Exercise 8

Here we solve a Resource Allocation problem formally stated as:

$$\begin{aligned} \max \quad & \sum_{i}^{N} log(x_i) \\ \text{s.t.} \quad & x_1 + x_2 \leq R_1 2 \\ & x_1 \leq R_2 3 \\ & x_3 \leq R_3 2 \\ & x_1 + x_2 + x_3 \leq 1 \\ \text{Var} \quad & x_i, R_j k \end{aligned} \tag{8.1}$$

Where x_i is thee traffic allocated to each user and R_{jk} is the percentage of time each link is used.

The problem has been solved with CVX toolbox, giving an optimal value of -3.99 after 12 ms as shown in figure 22

```
Solving exercise 8

Time to converge =  0.01438450813293457
status: optimal
optimal value p* =  -3.988984047216252
optimal var: x1 =  0.16666664923447433  x2 =  0.3333333506576221  x3 =  0.3333333506561061
optimal var: R12 =  0.4999999997865294  R23 =  0.16666664912911194  R32 =  0.33333335055074376
optimal dual variables λ1 2.9999997481224927
optimal dual variables λ2 =  2.9999997480909575
optimal dual variables λ3 =  2.999999748106721
optimal dual variables λ4 =  2.999999748075187
```

Figure 22: Result Exercise 8