CS164 ASSIGNMENT 3:

CONSTRAINED OPTIMIZATION & LINEAR REGRESSION

Minerva University

CS164 - Optimization Methods

Prof. J. Levitt

December 5, 2022

Table of Contents

Constrained Optimization & Linear Regression

Task 1

Consider the following linear problem:

$$minimize_x \; c^T x$$

$$subject\ to\colon\ Ax <= b$$

We have a feasible region $x = \{x!\ Ax <= b\}$ that satisfies constraints and defined by semi-plans that comes from inequality $Ax <= b$ (Appendix A). KKT conditions stays that $x^\star$ is optimal solution if and only if the following conditions hold true (Appendix B[1]):

Condition		Interpretation	Graphical Meaning
Primal feasibility	$Ax <= b$	$x^\star$ is a feasible solution	$x^\star$ lies in the feasible region satisfying the constraint.
Dual feasibility	$\lambda A + v = c$ $\lambda >= 0$	$x^\star$ is a feasible solution to the dual problem (Appendix C[2]), where λ and v are the Lagrangian multipliers, also known as dual variables	Primal base is associated with a dual base, where non-negativity constraints of initial variables correspond to actual constraints in the dual program. Gradients of constraints and the objective function point in opposite directions.
Complementary slackness	$\lambda(Ax - b) = 0$	At $x^\star$, the i_{th} inequality constraint is binding: either $\lambda_i = 0$ or $A_i x = b_i$	The values of the objective function of a primal base and its associated dual base are the same (Figure 2). If constraints are not active, $\lambda = 0$.

Table 1. *KKT conditions, their interpretations and geometrical meanings for the linear program.*

[1] In Appendix B, we show the notes of how we derived KKT conditions applying Farkas Lemma.
[2] In Appendix C, the Dual KKT Conditions are written for the given LP problem.

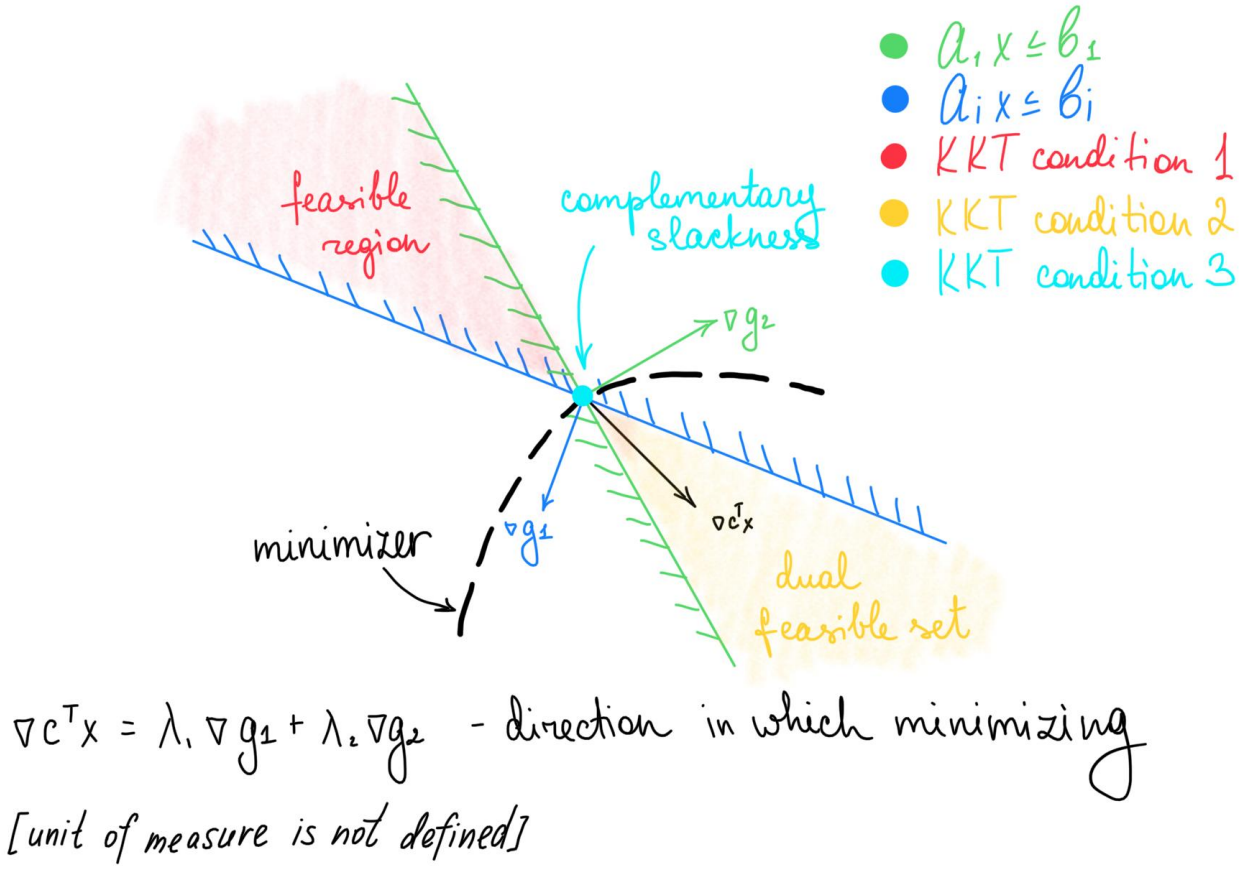

Figure 2. *The geometric insights into KKT conditions for the given linear program.*[3]

The third condition is exactly the reason why the optimal solution will never be found in the interior of the feasible region and will always be on a vertex, edge or facet. Since the primal program is a minimization problem, the value of its objective function is always more than the optimal value. Meanwhile, this is the opposite for the dual program (importantly, this reasoning works only when 1st condition is satisfied and variables are in the feasible set). Therefore, "the optimization problem is equivalent to finding a primal feasible base with an associated dual feasible base" (Hoang, 2016).

[3] **#dataviz:** Analyzing comments on my previous application of this HC, I paid great attention to careful labelling axes and showing the situation with units. Throughout the paper, I was using data visualization to help me and the reader understand challenging concepts. I also ensured that I have a great colour differentiation to make the message conveyed more straightforward.

We can prove it with calculus, constructing the Lagrangian and proving that, for any primal and dual variables satisfying the equality constraints, the difference of the primal by the dual objective functions is equal to the complementary slackness value.

Task 2

Since the least-squares estimator can be influenced by outlying data points, we consider l_1 and l_∞ norms (two other measures of a discrepancy) to provide a better fit. l_1 norm cost $min_\Theta ||Y - X\Theta||_1$ that is equal to $min_\Theta \sum_{i=1}^{i} s_i$, where s_i is a slack varible that is $>= \left|Y_i - X_i\Theta_i\right|$ and $>= 0$. This is an optimization problem that we can express as a linear program by opening the module and rewriting in epigraph form:

$$minimize_{s,\Theta}\ 1_i^T s$$

$$subject\ to:\ Y - X\Theta <= s$$

$$Y - X\Theta >= -s$$

$$s_1,\ldots, s_i >= 0$$

Then, let's perform the same operations for l_∞ norm ($min_\Theta ||Y - X\Theta||_\infty$) with a slack variable t:

$$minimize_{t,\Theta}\ t$$

$$subject\ to:\ Y - X\Theta <= 1_n t$$

$$Y - X\Theta >= -1_n t$$

$$t_1,\ldots, t_i >= 0$$

Task 3

Using a synthetic dataset and the CVXPY package (Appendix D[4]), we solve both optimization problems, and compute the parameters that define the line of best fit for each (Figure 3, Appendix E[5]).

```
Solution for  l1-norm intro optimization problem
4654.158202580338
Computed parameter θ that defines the line of best fit: [ 2.08882545 -0.0878932 ]
Solution for  l∞-norm intro optimization problem
73.69539077782807
Computed parameter θ that defines the line of best fit: [ 2.16279773 -7.70026163]
```

Figure 3. *The output of the code listed in Appendix D and E that calculates parameters* θ.

Then, we plot both lines in different colours over the scatter plot (Figure 4, Appendix F).

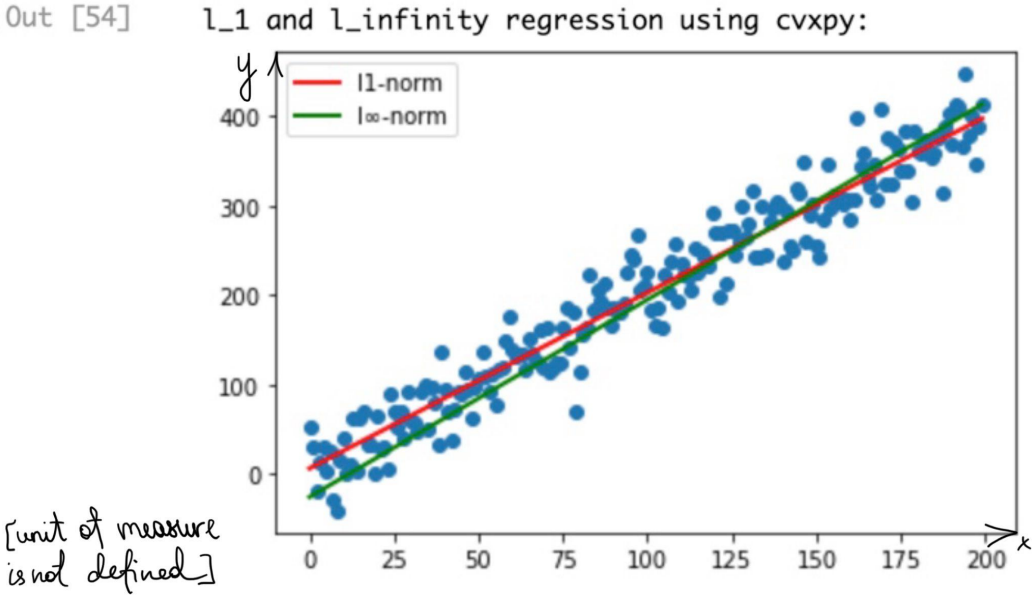

Figure 4. *The output of the code listed in Appendix F that shows the dataset and lines of best fit.*

Task 4

Following the structure above, we solve the Wine Problem, listing all of our steps in Appendix G.

[4] In Appendix D, we attached the code that we used to generate a synthetic dataset.
[5] In Appendix E, we declared a variable for the parameters of the line and a vector of slack variables.

Appendix

Appendix A

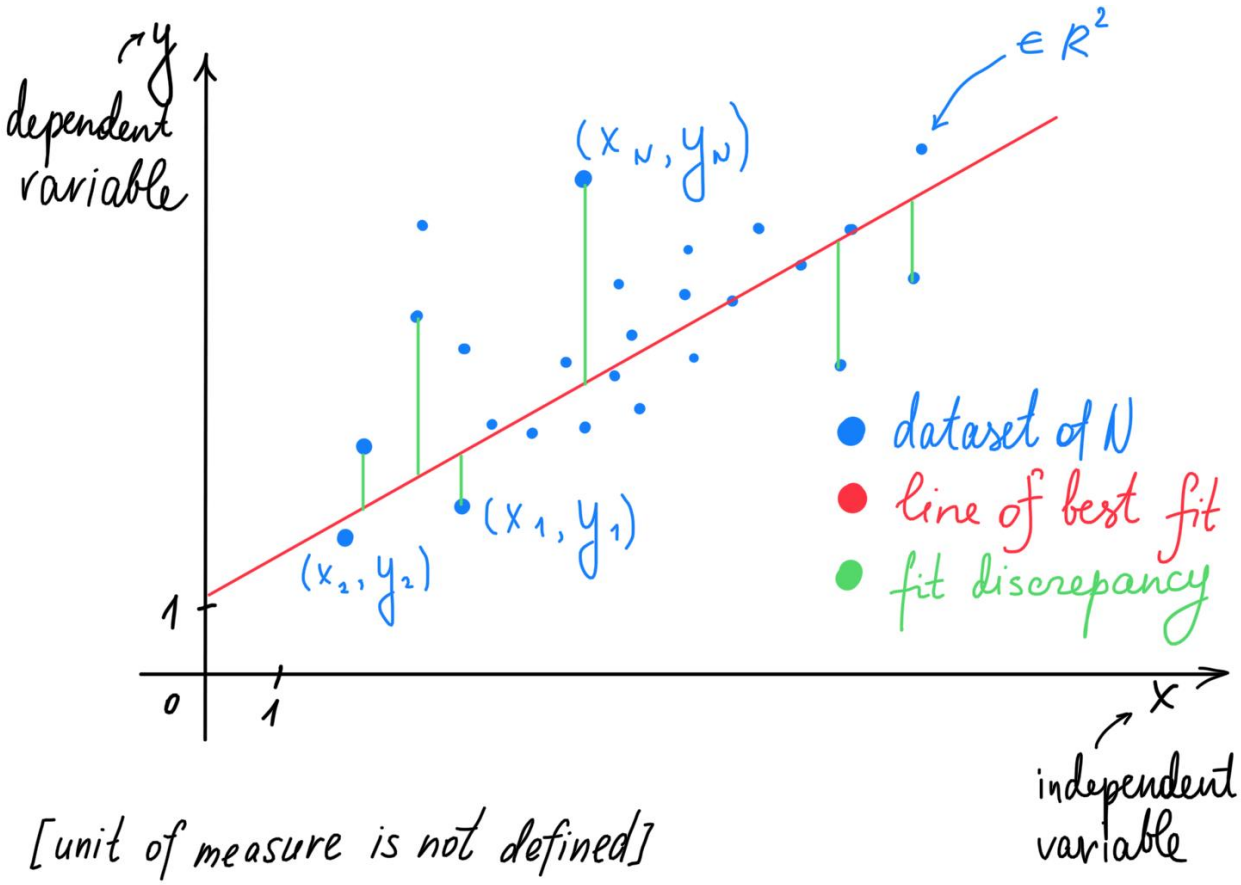

Figure 1. *The sketch created for a better understanding of the information given in the assignment's description. This helps to see what to expect after going through the three following tasks.*

Appendix B

$\min_x c^T x$

$Ax \le b$

(i.e. $a_i^T x \le b_i$)

assume x^* is optimal solution

If symbolize $Ax \le b$ binding constraints as G, then

$Gx^* = g$

$Gd \le 0$, where d is direction

↑ for the direction to be feasible

$G(x^* + \alpha d) = g + \alpha Gd \le 0$

↑ step size when we move along the direction d

if $Gd \ge 0$ then constraints are not satisfied,

the direction d won't be feasible at the point x^*.

$Gd \le 0$ ← feasible directions will be no solution

$c^T d \ge 0$ ← objective 'll increase along the direction

Applying Farkas Lemma, we know $G^T y = c$ has a solution

So we denote set I that sat. constr.:

$I = \{i : a_i^T x^* = b\}$

Rows of G: a_i^T, $i \in I$

Appendix C

Let's write the dual problem for the given LP:

$minimize_w\ wb$

$subject\ to$: $wA >= c^T$, where w is a row vector $w = [\lambda_1\ \lambda_2]$ if we consider the following example:

$A = \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix}$, c = $\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$, b = $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

Then we can write the dual problem as $minimize\ [\lambda_1\ \lambda_2]b$ subject to $[\lambda_1\ \lambda_2]\begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix} >= \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}^T$.

This can be rewritten more concretely as separate inequalities (i.e., $minimize$ $b_1 * \lambda_1 + b_2 * \lambda_2$ and the same for the constraints). Then, the constraints of this problem remind the dual feasibility constraints, but we need to add surplus variables $k_1, k_2 \geq 0$:

$minimize_\lambda\ b_1\lambda_1 + b_2\lambda_2$

$subject\ to$: $a_1\lambda_1 + a_3\lambda_2 - k_1 >= c_1$

$a_2\lambda_1 + a_4\lambda_2 - k_2 >= c_2$

To show the relationship between initial and dual problems, we write dowm KKT conditions for the dual problem:

- Primal Feasibility: $a_1\lambda_1 + a_3\lambda_2 - k_1 >= c_1$, $a_2\lambda_1 + a_4\lambda_2 - k_2 >= c_2$
- Dual Feasibility: $\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - x_1 * \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - x_2 * \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - s_1\begin{bmatrix} -1 \\ 0 \end{bmatrix} - s_2\begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- Complementary Slackness: $x_1(-a_1\lambda_1 - a_3\lambda_2 + c_1) = 0$,

  $x_2(-a_2\lambda_1 - a_4\lambda_2 + c_2) = 0$.

The dual feasibility conditions for the initial LP are the same with to the feasibility

conditions for the dual problem and vice-versa. “Two linear programming problems are dual to each other if and only if they share KKT conditions with the primal and dual feasibility conditions swapped” (Griffin, 2016).

Appendix D

```
#import needed libraries
import numpy as np
import cvxpy as cvx
import matplotlib.pyplot as plt

# generate a synthetic dataset

# actual parameter values
theta1_act = 2
theta2_act = 5

# Number of points in dataset
N = 200

# Noise magnitude
mag = 30

# datapoints
x = np.arange(0,N)
y = theta1_act * x + theta2_act *np.ones([1,N]) + np.random.normal(0,mag,N)

# Scatter plot of data
plt.figure()
plt.scatter(x,y)
plt.show()
```

Appendix E

```
def lnorm(intro,x,y,N):
    print("Solution for ", intro, "intro optimization problem")
    theta = cvx.Variable(2)
    if  intro == "l1-norm":
        s = cvx.Variable(N)
        objective = cvx.Minimize(np.ones([N,1]).T@s) #alternatively cvx.Minimize(sum(s))
    elif intro == "l∞-norm":
        s = cvx.Variable()
        objective = cvx.Minimize(s)
    else:
        print("Defined function is applicable only for l1 and l∞-norm")
    constraints = []
    X = np.ones((N,2)) #a matrix X: X^T*theta = x*theta_1 + theta_2
    for i in range(N):
        X[i,0] = x[i]
        if  intro == "l1-norm":
            a = s[i]
        elif intro == "l∞-norm":
            a = s
        constraints.append(a >= 0)
        constraints.append(y[0,i] - X[i,]@theta <= a)
        constraints.append(y[0,i] - X[i,]@theta >= -a)
    prob = cvx.Problem(objective, constraints)
    print(prob.solve())
    return print("Computed parameter θ that defines the line of best fit:", theta.value)

theta1 = lnorm("l1-norm",x,y,N)
thetainf = lnorm("l∞-norm",x,y,N)
print(theta1,thetainf)
```

Appendix F

```
    prob.solve()
    return theta.value

theta1 = lnorm("l1-norm",x,y,N)
thetainf = lnorm("l∞-norm",x,y,N)
y1 = theta1[0]*x + theta1[1]
yinf = thetainf[0]*x + thetainf[1]
plt.plot(x, y1, '-r', label='l1-norm',linewidth=2)
plt.plot(x, yinf, '-g', label='l∞-norm',linewidth=2)
plt.legend(loc='upper left')
plt.show()
```

Appendix G

Representing the table data by the matrix $X \in R^{3\times 6}$, we are mixing the three wines together in a proportion that will most closely match the target flavour profile in the six

dimensions given by $y = [3, 2, 2, 2, 3, 2]$. We first transfer the matrix to the equations:

	Sugar	Acid	Alcohol	Fruity	Flowery	Intensity
W1	1	1	2	3	2	5
W2	2	1	2	3	2	5
W3	1	3	5	1	3	1

$$\Rightarrow \begin{cases} x + 2y + z = 3 \\ x + y + 3z = 2 \\ 2x + 2y + 5z = 2 \\ 3x + 3y + z = 2 \\ 2x + 2y + 3z = 3 \\ 5x + 5y + z = 2 \end{cases}$$

This can be transformed into $f(x, y, z) = (x + 2y + z - 3)^2 +$

$+ (x + y + 3z - 2)^2 + (2x + 2y + 5z - 2)^2 + (3x + 3y + z - 2)^2 +$

$+ (2x + 2y + 3z - 3)^2 + (5x + 5y + z - 2)^2$.

Then, we denote the proportion of each wine by the vector θ that has the following constraints:

$\theta >= 0$, since we cannot have a negative amount of wine.

$\sum_{i=1}^{i} \theta_i = 1$ to have the complete mixture, where the vector's numbers describe proportions.

These constraints are part of the optimization problem that is also known as a Quadratic Program.

$minimize_{x,y,z} f$

$subject\ to: x, y, z >= 0$

$h(x, y, z) = 0$, where $h(x, y, z) = x + y + z - 1$

$R(x, y, z) = f(x, y, z) - \mu_1 x - \mu_2 y - \mu_3 z + L(h(x, y, z))$, where a vector of multipliers μ shows the inequality constraints. Non-zero elements in μ tell us that the constraint must be an equality constraint. If we find an optimum with a non-zero value of μ, it means that

the line of best fit is not accessible when the constraint is active.

Then, we write down the KKT conditions for optimality of the problem that we have derived above, using a multiplier λ for the equality constraint:

- Primal feasibility: $-x <= 0,\ -y <= 0,\ -z <= 0,\ h = 0.$
- Dual feasibility: $\mu_i >= 0.$
- Complementary slackness: $-\mu_1 x = 0,\ -\mu_2 y = 0,\ -\mu_3 z = 0.$
- Stationarity: $0\ is\ in\ R'(x, y, z).$

Then, we solve the KKT system to find the fractions that allow the flavour profile to be most closely matched. To do this, we try eight different combinations of active constraints for the three elements of θ just as we did in the Break-out Group in Session 14 (Appendix G). After running this code, we can see that all the constraints are satisfied only when μ_1 is an active constraint (Appendix H). Then, $x = 0.0,\ y = 0.54,\ z = 0.46;\ x, y, z >= 0\ ; y + z = 1.$

Appendix H

```
A = matrix(RR, [
    [88, 90, 56, 1, -1, 0, 0],
    [90, 94, 58, 1, 0, -1, 0],
    [56, 58, 92, 1, 0, 0, -1],
    [1, 1, 1, 0, 0, 0, 0],
    [-1, 0, 0, 0, 0, 0, 0],
    [0, -1, 0, 0, 0, 0, 0],
    [0, 0, -1, 0, 0, 0, 0]
])
v = vector(RR, [62, 68, 64, 1, 0, 0, 0])

for i in range(0, 4):
    combs = c([4, 5, 6], i)
    for rem in combs:
        active_constraints = set([4,5,6]) - set(rem)
        active_constraints = list(map(lambda x: "u" + str(x-3), active_constraints))
        active_constraints = ', '.join(active_constraints)
        print(f"Active constraints: {active_constraints if active_constraints else 'none'}")

        mask = np.ones(7, dtype=bool)
        for index in rem:
            mask[index] = 0

        A_new = matrix([np.array(A[k])[mask] for k in range(A.nrows()) if k not in rem])
        v_new = vector(np.array(v)[mask])
        B = A_new.augment(v_new)
        B = B.rref()
        result = B.columns()[-1]
        x, y, z, L, *u = result
        print(f"x={x}, y={y}, z={z}, L={L}, u={u}")
        if x < 0 or y < 0 or z < 0:
            print("INVALID: One of (x, y, z) is negative")
        if x + y + z > 1:
            print("INVALID: Equality constraint is broken")
        for u_i in u:
            if u_i < 0:
                print('INVALID: u_i is negative')
                break
        print()
```

Appendix I

```
Active constraints: u1, u2, u3
x=0.026315789473684157, y=0.0, z=0.0, L=-1.5170019797458472e+16,
u=[-1.5170019797458446e+16, -1.517001979745845e+16, -1.5170019797458482e+16]
INVALID: u_i is negative

Active constraints: u2, u3
x=1.0, y=0.0, z=0.0, L=-25.999999999999993, u=[-3.999999999999996, -33.99999999999999]
INVALID: u_i is negative

Active constraints: u1, u3
```

```
x=5.551115123125782e-17, y=1.0000000000000004, z=0.0, L=-25.999999999999993,
u=[1.999999999999997, -31.999999999999996]
INVALID: Equality constraint is broken
INVALID: u_i is negative

Active constraints: u1, u2
x=0.0, y=0.0, z=0.999999999999998, L=-27.99999999999993, u=[-33.99999999999987,
-37.99999999999987]
INVALID: u_i is negative

Active constraints: u3
x=-0.9999999999999992, y=1.9999999999999993, z=0.0, L=-29.999999999999986,
u=[-33.99999999999999]
INVALID: One of (x, y, z) is negative
INVALID: u_i is negative

Active constraints: u2
x=0.5000000000000002, y=0.0, z=0.4999999999999999, L=-9.999999999999998,
u=[-3.999999999999996]
INVALID: u_i is negative

Active constraints: u1
x=0.0, y=0.5428571428571434, z=0.4571428571428571, L=-9.542857142857143,
u=[2.91428571428571]
INVALID: Equality constraint is broken

Active constraints: none
x=-1.4999999999999991, y=1.9999999999999993, z=0.4999999999999999,
L=-13.999999999999993, u=[]
INVALID: One of (x, y, z) is negative
```