Session 23: Introduction to non-linear programming (NLP) Solutions

A generic non-linear program has the following form (x is a n-dimensional vector.)

$$\begin{aligned} \text{Minimize}: \quad & f(x) \\ g_1(x) &\leq 0 \\ &\cdots \\ g_m(x) &\leq 0 \\ x &\in X \end{aligned}$$

In a linear program, all of the functions $f, g_1, \cdots, g_m$ are linear, and the constraint set X is n-dimensional real numbers. In a MIP, the constraint set X also requires certain variables to be integers.

Unlike with LP and MIPs, there are no generic algorithms that can solve arbitrary non-linear programs with large n or m. The strategy in practice is to find special structure in the functions f and g_j.

1. Sum of Squares

Let x, y and z be decision variables. Gurobi allows the following types of quadratic constraints:

- **Minimizing** a sum of squares plus a linear term, or an expression that can be expressed as a sum of squares plus a linear term.
- A sum of squares plus a linear term **less than** equal to a linear term.

Examples of what is allowed:

$$\begin{aligned} \text{Minimize:} \quad & x^2 + (x+2y)^2 + 3(y+z)^2 - 5y \\ x^2 + 2y^2 + 3z^2 &\leq 5 \\ (x+2y)^2 &\leq 5z - 2x \\ x^2 + 4y^2 + 4xy &\leq 5z - 2x \\ 6x + y^2 + 2z &\leq 10 \end{aligned}$$

However, in the code, you cannot use the power ** operator or ^2, but must multiply out using *:

```
[1]: from gurobipy import Model, GRB
     mod=Model()
     x=mod.addVar(lb=-GRB.INFINITY)
     y=mod.addVar(lb=-GRB.INFINITY)
     z=mod.addVar(lb=-GRB.INFINITY)

     mod.setObjective(x*x+(x+2*y)*(x+2*y)+3*(y+z)*(y+z)-5*y)
     mod.addConstr(x*x+2*y*y+3*z*z <= 5)
     mod.addConstr((x+2*y)*(x+2*y)<=5*z-2*x)
     mod.addConstr(x*x+4*y*y+4*x*y<=5*z-2*x)
     lhs4=6*x+y*y+2*z
     mod.addConstr(lhs4<=10)
     mod.optimize()
```

```
Gurobi Optimizer version 9.0.1 build v9.0.1rc0 (win64)
Optimize a model with 0 rows, 3 columns and 0 nonzeros
Model fingerprint: 0xd99a98ef
Model has 5 quadratic objective terms
Model has 4 quadratic constraints
...
Barrier solved model in 8 iterations and 0.05 seconds
Optimal objective -2.04110515e+00
```

[2]:
```python
print(f'Optimal solution: obj={mod.objval}')
print(f'\t x={x.x}')
print(f'\t y={y.x}')
print(f'\t z={z.x}')
print(f'\t lhs4={lhs4.getValue()}')
```

```
Optimal solution: obj=-2.0411051519358554
         x=-1.1342962897247952
         y=0.79686555380835
         z=-0.41150244505172434
         lhs4=-6.993787917605932
```

The following are **NOT allowed**:

- Subtracting a square instead of adding:

$$\text{Minimize: } x^2 + (x+2y)^2 - 3(y+z)^2 - 5y$$

- Maximizing a sum of squares:

$$\text{Maximize: } x^2 + (x+2y)^2 + 3(y+z)^2 - 5y$$

- A sum of squares larger than a linear expression:

$$x^2 + y^2 - 2xy \geq 5$$

Q1 (DMD Example 8.1)

Solve the following non-linear optimization formulation problem using Gurobi. The formulation maximizes expected returns of a portfolio subject to not exceeding a certain level of risk.
Decision variables: Let A, G, D denote the fraction of total investment to put in the assets Advent, GSS, and Digital.
Objective and constraints:

$$\begin{aligned}
\text{Maximize:} \quad & 11A + 14G + 7D \\
\text{subject to:} \quad & \\
\text{(Fractions)} \quad & A + G + D = 1 \\
\text{(Target risk)} \quad & \sqrt{16A^2 + 22G^2 + 10D^2 + 6AG + 2GD - 10AD} \leq 3.1 \\
\text{(Nonnegativity)} \quad & A, G, D \geq 0
\end{aligned}$$

```
[3]: from gurobipy import Model, GRB
     mod=Model()
     A=mod.addVar()
     G=mod.addVar()
     D=mod.addVar()
     mod.setObjective(11*A+14*G+7*D, sense=GRB.MAXIMIZE)
     mod.addConstr(A+G+D == 1)
     riskSquared=16*A*A+22*G*G+10*D*D+6*A*G+2*G*D-10*A*D
     mod.addConstr(riskSquared <= 3.1**2)
     mod.setParam('outputflag',False)
     mod.optimize()
```

```
[4]: import numpy as np
     print('Optimal annual return:', mod.objval)
     print('\t A:',A.x)
     print('\t G:',G.x)
     print('\t D:',D.x)
     print('\t risk:',np.sqrt(riskSquared.getValue()))
```

```
Optimal annual return: 12.250136110681215
         A: 0.37801476764694214
         G: 0.534011005727641
         D: 0.08797422662541113
         risk: 3.0999991921462304
```

2. Linearizing using Auxiliary Decision Variables

2.1 Max and Min

The non-linear objective

$$\text{Minimize } \max(x, y)$$

is equivalent to

$$\begin{aligned} \text{Minimize} \quad & z \\ \text{subject to} \quad & \\ & \max(x, y) \leq z \end{aligned}$$

which is equivalent to the linear formulation:

$$\begin{aligned} \text{Minimize} \quad & z \\ \text{subject to} \quad & \\ & x \leq z \\ & y \leq z \end{aligned}$$

Similarly,

$$\text{Minimize } \max(x, y) - \min(x, y)$$

is equivalent to

$$\begin{aligned} \text{Minimize} \quad & U - L \\ \text{subject to:} \quad & \\ & L \le x \le U \\ & L \le y \le U \end{aligned}$$

2.2 Absolute Values

Similarly, the non-linear objective

$$\text{Minimize } |x_1 - y_1| + |x_2 - y_2|$$

is equivalent

$$\begin{aligned} \text{Minimize} \quad & z_1 + z_2 \\ \text{subject to} \quad & \\ \text{(New constraint 1)} \quad & |x_1 - y_1| \le z_1 \\ \text{(New constraint 2)} \quad & |x_2 - y_2| \le z_2 \end{aligned}$$

Now, because $|x| = \max(x, -x)$, the above is equivalent to the linear formulation:

$$\begin{aligned} \text{Minimize} \quad & z_1 + z_2 \\ \text{subject to} \quad & \\ & x_1 - y_1 \le z_1 \\ & y_1 - x_1 \le z_1 \\ & x_2 - y_2 \le z_2 \\ & y_2 - x_2 \le z_2 \end{aligned}$$

2.3 Big-M Method

Suppose we want to either turn a continuous variable X on or off, we can do

$$0 \le X \le MZ,$$

where Z is a binary decision variable and M is a sufficiently large number that is guaranteed to be larger than the maximum possible value of X in any optimal soluiton.

2.4 Either/Or Constraint

Similar to the above, suppose that we want to force a continuous variable X to be either between A_1 and A_2 or between B_1 and B_2, we can do

$$ZA_1 + (1 - Z)B_1 \le X \le ZA_2 + (1 - Z)B_2.$$

Q2 (Portfolio Optimization with Complex Constraints)

Consider the following formulation of a portfolio optimization problem, **use auxiliary decision variables to linearize the last four constraints.**
Data:

- S: the set of stocks.

- w_i: the old weight of stock $i \in S$ before optimization. (The "weight" of a stock is % of total funds invested in the stock; weights of all stocks should add to one.)
- R_i: the expected annual return of stock $i \in S$.
- C_{ij}: the estimated covariance between stocks $i, j \in S$.
- σ_{target}: the maximum volatility of the final portfolio.
- Δ: the total movement allowed between the old weights and the new weights.
- k: the maximum # of stocks allowed in the portfolio.
- ϵ: the minimum non-zero weight allowed.
- λ: penalty for different weights for stocks 1, 2 and 3.

Decision variables:

- x_i: the new weight of stock i. (Continuous)
- y: variation of weights among stocks 1, 2 and 3. (Continuous)

Formulation: All summations are over the set S of stocks.

$$
\begin{array}{rcl}
\text{Maximize:} & \sum_i R_i x_i - \lambda y & \text{(Average Return)} \\
\text{subject to:} & & \\
\text{(Valid weights)} & \sum_i x_i = 1 & \\
\text{(Risk tolerance)} & \sum_{i,j} C_{ij} x_i x_j \leq \sigma_{target}^2 & \\
\text{(Change in weights)} & \frac{1}{2} \sum_i |x_i - w_i| \leq \Delta & \\
\text{(Non-negligible weights)} & \text{If } x_i > 0 \text{ then } x_i \geq \epsilon & \text{for each stock } i. \\
\text{(Simplicity)} & (\#\text{ of stock } i \text{ with } x_i > 0) \leq k & \\
\text{(Similar weights for stocks 1-3)} & \max(x_1, x_2, x_3) - \min(x_1, x_2, x_3) \leq y & \\
 & x_i \geq 0 &
\end{array}
$$

Solution to Q2.

Auxiliary decision variables:

- δ_i: the absolute difference between x_i and w_i. (Continuous)
- z_i: whether to have a non-zero weight on stock i. (Binary)
- u: upper bound of x_1, x_2 and x_3.
- l: lower bound of x_1, x_2 and x_3.

Linearized constraints:

$$
\begin{array}{rrl}
\text{(Change in weights 1)} & x_i - w_i \leq \delta_i & \text{for each stock } i. \\
\text{(Change in weights 2)} & -(x_i - w_i) \leq \delta_i & \text{for each stock } i. \\
\text{(Change in weights 3)} & \frac{1}{2} \sum_i \delta_i \leq \Delta & \\
\text{(Non-negligible weights)} & \epsilon z_i \leq x_i \leq z_i & \text{for each stock } i. \\
\text{(Simplicity)} & \sum_i z_i \leq k & \\
\text{(Similar weights 1)} & l \leq x_i \leq u & \text{for } i \in \{1, 2, 3\}. \\
\text{(Similar weights 2)} & u - l \leq y &
\end{array}
$$

Note that the simplicity constraints can be expressed simply because we already have $x_i \le z_i$ from the previous constraint, which is equivalent to the Big M method linking the continuous variable x_i with the binary variable z_i.

3. (Optional) Special Non-Linear Constraints Supported in Gurobi

The following information will not be tested in any exam or quiz, but may be helpful for the final project.

In the following table, x, y, z, and w are decision variables. Moreover, the code assume that you have imported all of the functions.

```
from gurobipy import Model, GRB, max_, min_, abs_, and_, or_
mod=Model()
```

Non-linear relationship	Sample Constraint as Math Expression	Gurobi Command
Maximum of arbitrary variables	$x = max(y, z, 5)$	`mod.addConstr(x==max_(y,z,5))`
Minimum of arbitrary variables	$x = min(y, z, 5)$	`mod.addConstr(x==min_(y,z,5))`
Absolute value of arbitrary variables	$x = max(y, -y)$	`mod.addConstr(x==abs_(y))`
AND of binary variables	$x = min(y, z, w)$	`mod.addConstr(x==and_(y,z,w))`
OR of binary variables	$x = max(y, z, w)$	`mod.addConstr(x==or_(y,z,w))`
At most one (arbitrary variable) non-zero	$\mathbb{1}(x \neq 0) + \mathbb{1}(y \neq 0) + \mathbb{1}(z \neq 0) \le 1$	`mod.addSOS(GRB.SOS_TYPE1,[x,y,z])`

Example:

```
[5]: from gurobipy import Model, GRB, max_, min_, and_, abs_
     mod=Model()
     x=mod.addVar(lb=-GRB.INFINITY)
     y=mod.addVar(lb=-GRB.INFINITY)
     z=mod.addVar()
     l=mod.addVar(lb=-GRB.INFINITY)
     a=mod.addVar()
     mod.addConstr(z==max_(x,5,y))
     mod.addConstr(l==min_(x,y))
     mod.addConstr(z<=l+5)
     mod.addConstr(a==abs_(y))
     mod.addSOS(GRB.SOS_TYPE1, [x,y])
     mod.setObjective(z-l-a,sense=GRB.MAXIMIZE)
     mod.setParam('OutputFlag',False)
     mod.optimize()
     print('\nObjective',mod.objval)
     print(f'x={x.x} y={y.x} z={z.x} l={l.x} a={a.x}')
```

```
Objective 5.0
x=5.0 y=0.0 z=5.0 l=0.0 a=0.0
```