GUROBI
OPTIMIZATION

MINLP

Solving Nonlinear Problems with Gurobi

Dan Jeffrey
Senior Technical Account Manager

Agenda

1. Applications for Nonlinear Solvers
2. Quadratic Solvers in Gurobi
3. Nonlinear API's in Gurobi
4. New MINLP Solver
5. Model Walkthrough

4 Gurobi Model Types for Nonlinear Models

… or maybe 8, depends how you count!

Quadratic terms

- Quadratic solvers in Gurobi: QP, QCP, MIQP, MIQCP, non-convex MIQCP

Higher-order Nonlinear terms

- Piece-wise linear constraints
  - Manual - Specify piecewise points manually
  - Automatic Functions
- Mixed Integer Nonlinear Programs (MINLP)
- Nested quadratic

Agenda

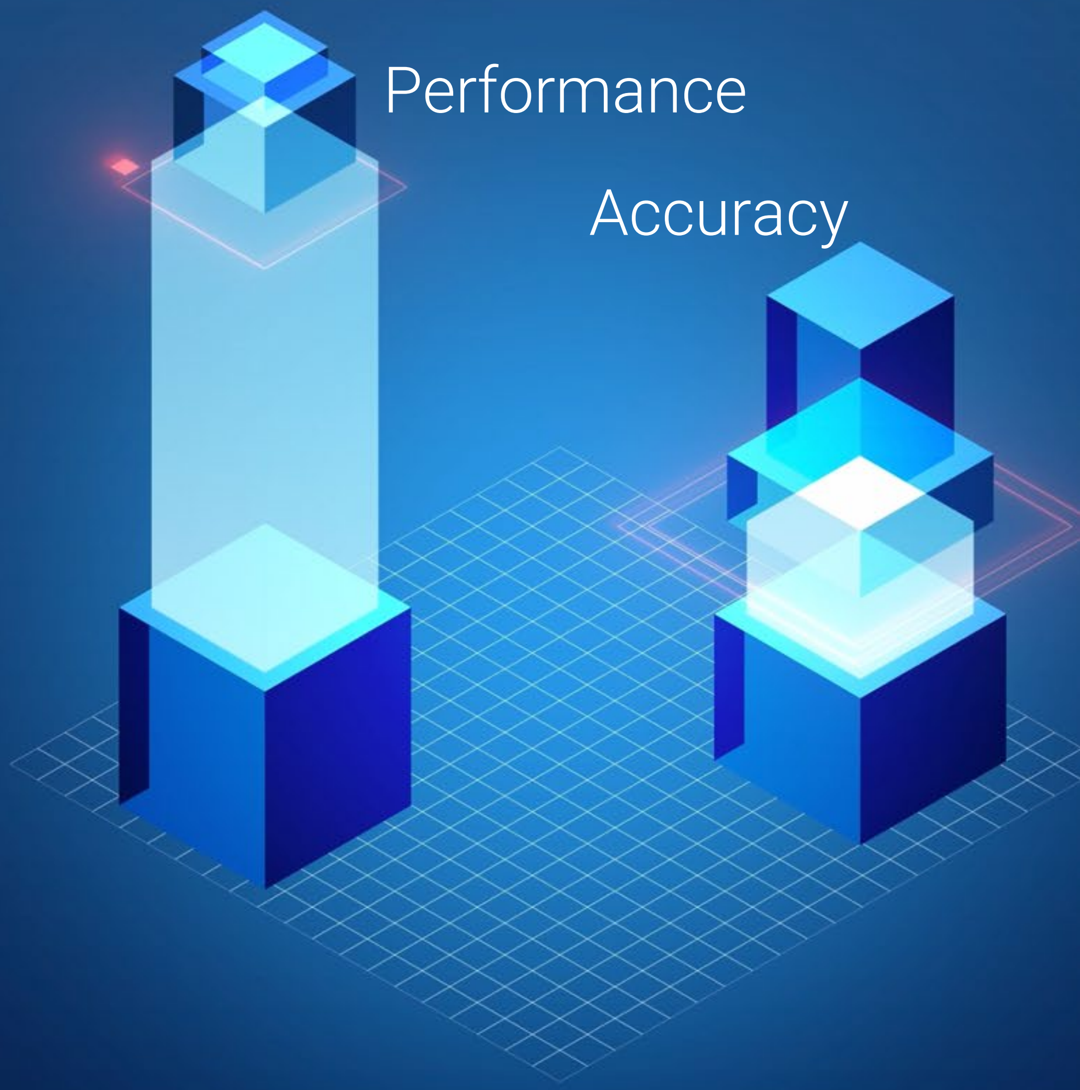
1. **Applications for Nonlinear Solvers**
2. Quadratic Solvers in Gurobi
3. Nonlinear API's in Gurobi
4. New MINLP Solver
5. Model Walkthrough

General Use Cases & Messaging

The world is full of well-known nonlinear relationships across all industries.

- Physical laws (e.g., in energy systems)
- Statistical measures (e.g., in finance, etc.)
- Nonlinear regression (explaining data points with nonlinear functions)

Many optimization software systems today work only with approximations that deliver acceptable performance and accuracy. **Time to try the new alternative – MINLP!**



Finance

Market impact term in programmed trading applications

$y = x^{3/2}$

Portfolio Optimization –
Use nonlinear to identify optimal allocation of assets.

Utilities

AC Optimal Power Flow

Includes sine and cosine functions
to accurately depict power flow.

Oil & Gas

Many applications in refineries

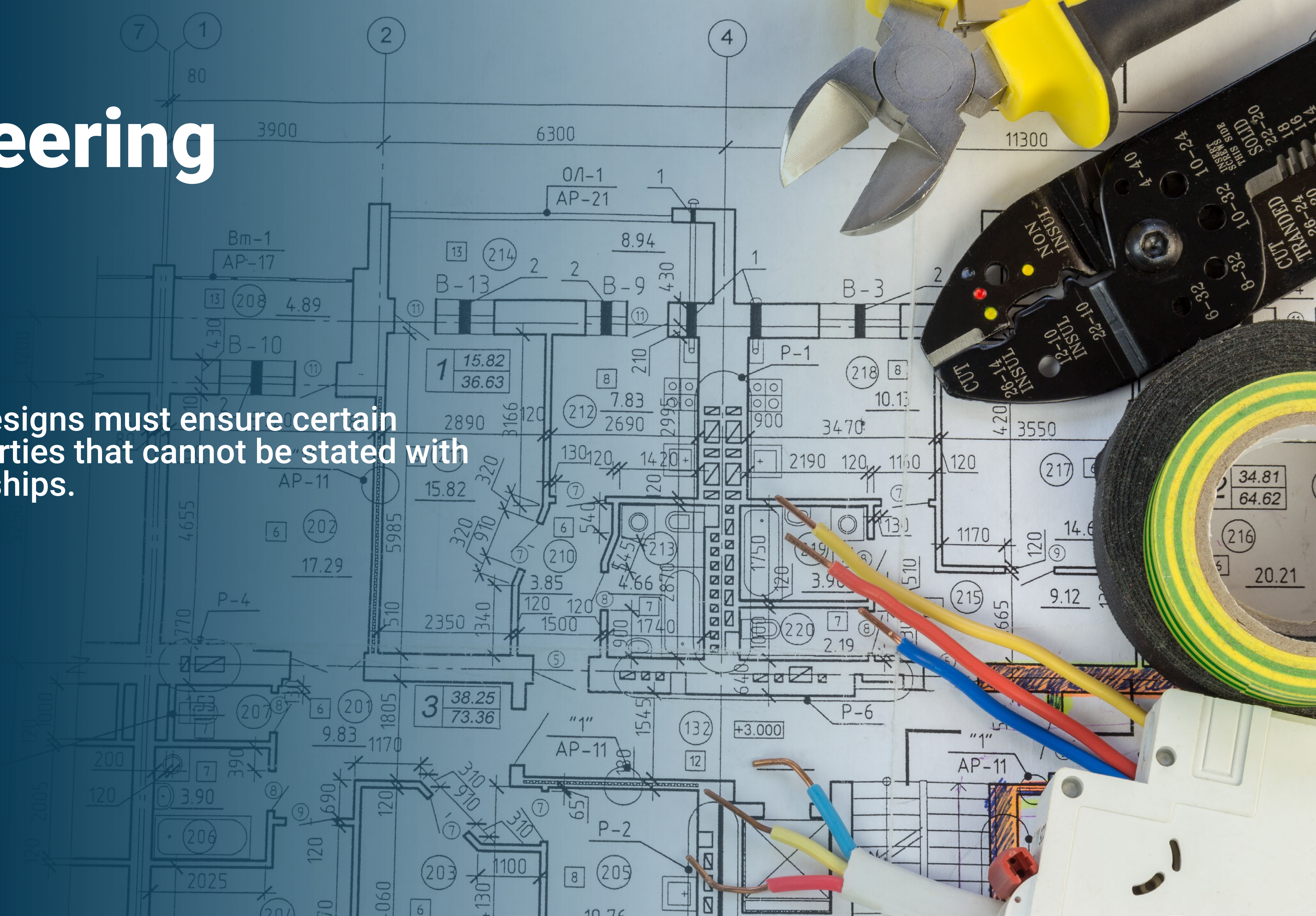
Nonlinear is essential to accurately model complex processes.

Heat exchanger

$(x-y)/(\log x - \log y)$

Engineering

Engineering designs must ensure certain physical properties that cannot be stated with linear relationships.



Agriculture

Crop planning and management to optimize planting schedules, irrigation, and fertilizer

Agenda

1. Applications for Nonlinear Solvers
2. **Quadratic Solvers in Gurobi**
3. Nonlinear API's in Gurobi
4. New MINLP Solver
5. Model Walkthrough

Solver Algorithms Available in Gurobi Optimizer

Gurobi solves a broad variety of problem types

LP	QP	QCP	Non-Convex MIQCP
MILP (including PWL)	MIQP	Convex MIQCP	MINLP

Quadratic Terms

QP – Quadratic Programs

- Objective contains quadratic terms

$$min\ xTQ\ x + pTx$$
$$s.t.\ Ax = b$$
$$x \geq 0$$

QCP

$$x^T Qx + qTx \leq b$$
$$x^T Qx \leq y$$
$$x^T Qx \leq yz$$

MIQP

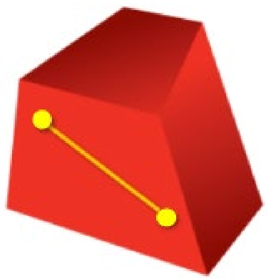
- Same as QP where x is integer

MIQCP

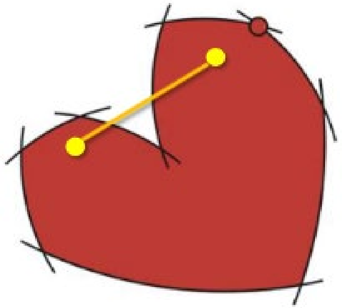
- Same as QCP where x is integer

non-convex MIQCP

- Automatic in Gurobi 11 – no parameter needed



convex



not convex

Confirm the Need for Nonlinear

Harder/slower to solve

Business value of a precise Nonlinear solution?

Examine ways to approximate

- See next slide

Is the Nonlinear curve really an approximation?

- Is truly continuous?
- Machine-learning alternatives
  - Integrate with a Gurobi linear program
  - gurobipy-machinelearning

GUROBI OPTIMIZATION

Confirm the Need for Nonlinear

Nested quadratics

- *example*:

$$maximize\ z = x^3$$

- *can be replaced with*:

$$y == x \cdot x$$
$$z == y \cdot x$$

Multivariate monomial terms

- "How do I model multilinear terms in Gurobi?" -- https://bit.ly/3vjz6fe
- For this constraint:

$$x \cdot y \cdot z == d$$

- Use this instead:

$$x \ \cdot\ y = w$$
$$w \ \cdot\ z = d$$

Agenda

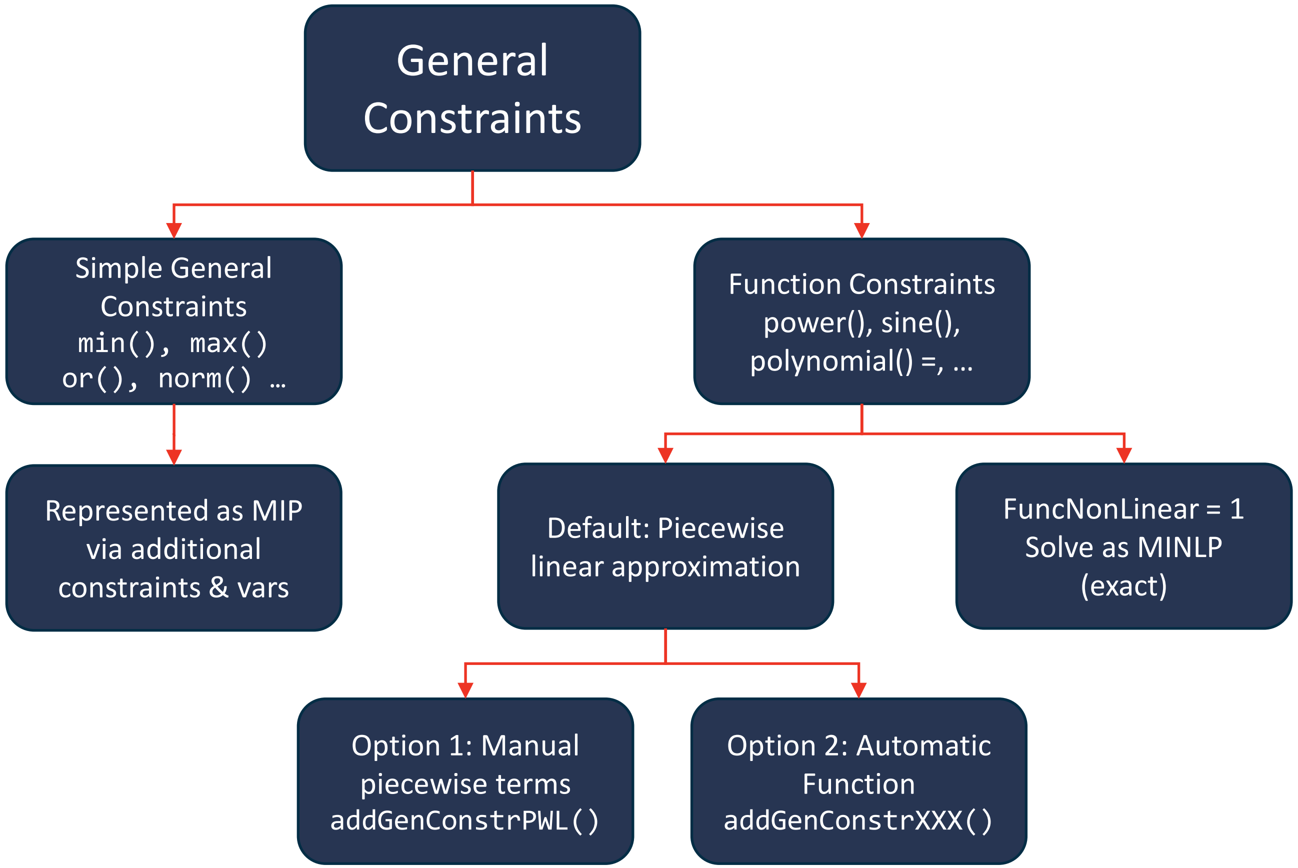
1. Applications for Nonlinear Solvers
2. Quadratic Solvers in Gurobi
3. **Nonlinear API's in Gurobi**
4. New MINLP Solver
5. Model Walkthrough

GUROBI
OPTIMIZATION

General Constraints

Gurobi supports two types of general constraints

Different algorithmic implementations



APIs for Nonlinear Constraints in Gurobi

API's to define nonlinear functions (Gurobi 9.0+)

Functions	API's
e^x, a^x	`addGenConstrExp()` `addGenConstrExpA()`
$\ln(x), \log_a(x)$	`addGenConstrLog()` `addGenConstrLogA()`
$\sin(x), \cos(x), \tan(x)$	`addGenConstrSin()` `addGenConstrCos()` `addGenConstrTan()`
x^a	`addGenConstrPow()`
$ax^3 + bx^2 + cx + d$	`addGenConstrPoly()`

Gurobi 9.0 – 10.0:

- Nonlinear functions always replaced by piecewise-linear approximations

Gurobi 11.0:

- You choose how treat nonlinear constraints
  - Approximation using piecewise linear
  - Exactly using MINLP solver

Composite Nonlinear Functions

Gurobi 11.0 can handle selected univariate constraints $f(x) = y$

- Trigonometric, power functions, logarithms, exponentials, etc.
- Use them as building blocks for more elaborate functions

Suppose we want to model this:

$$f(\mathrm{x}) = \sqrt{1+\mathrm{x}^2} + \ln\left(\mathrm{x} + \sqrt{1+\mathrm{x}^2}\right) \leq 2,\ \mathrm{x} \geq 0$$

We introduce auxiliary variables $u, v, w, z \geq 0$ and constraints as follows:

$$\begin{aligned} \mathrm{u} &= 1 + \mathrm{x}^2 \qquad & u &= v^2 \\ w &= \mathrm{x} + v \qquad & z &= \ln w \end{aligned}$$

Then $f(\mathrm{x}) \leq 2$ can be represented as:

$$v + z \leq 2$$

Significant effects on overall tolerances.

More in a coming slide.

Limitations of Univariate API's

Example: $y = \frac{x}{\sin x}$

One solution:

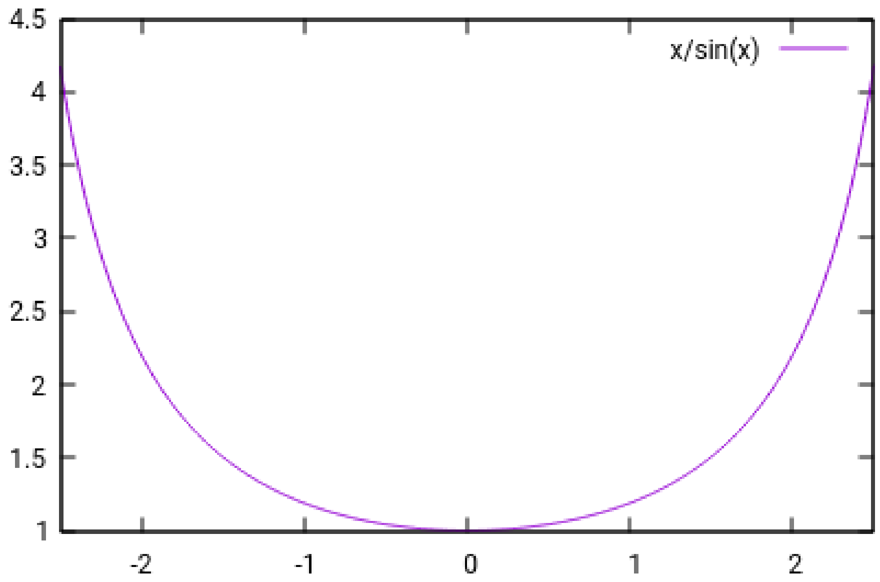
- $x' = 0.0001$
- $y' = 1.0000000016666666$

Gurobi model:

$$u = \sin x$$
$$v = u^{-1}$$
$$y = x \cdot v$$

One solution:

- $x' = 0.0001$
- $u' = 0.000099999999833333343$
- $v' = 10000.000016666666$
- $y' = 1.0000000016666666$



A solution with a violation within a tolerance of 10^{-6}:

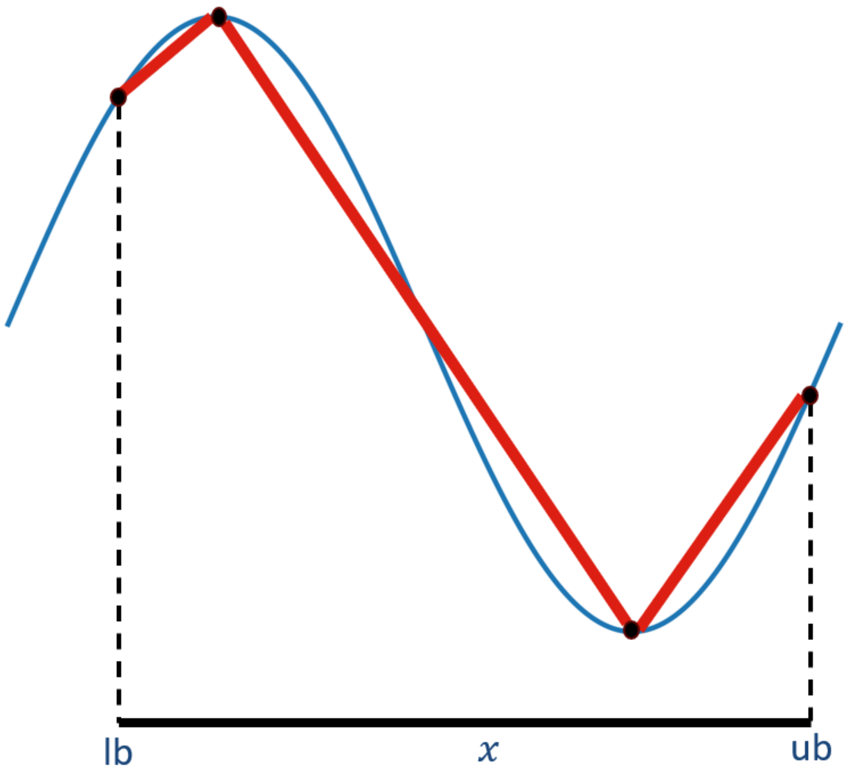
- $x'' = 0.0001$
- $u'' = 0.000098999999833333343$ $u'' = \sin x'' - 10^{-6}$
- $v'' = 10101.010118015167$
- $y'' = 1.0101010118015167$

Violation of 10^{-6} in auxiliary constraint leads to violation of 10^{-2} in composite constraint

Nonlinear: Automatic Function Constraints

Automatic Functions

- Default for nonlinear constraints
- Gurobi generates piecewise terms
- Pass Nonlinear term
- Optionally pass precision information:
  - `FuncPieces, FuncPieceError, FuncPieceLength, FuncPieceRatio`



PWL approximation

Advantages

- Solver works on a linear representation
- Faster solving

Disadvantages

- Problems can get quite large

Agenda

1. Applications for Nonlinear Solvers
2. Quadratic Solvers in Gurobi
3. Nonlinear API's in Gurobi
4. **New MINLP Solver**
5. Model Walkthrough

Nonlinear: New MINLP Solver in Gurobi

Solves Nonlinear, integer problems

- General, Nonlinear terms
- Finds global optimum
- Not using an approximation

Advantages

- Solves exact representation of the problem
- Smaller problem size

Disadvantages

- Branching takes time

Using MINLP in Gurobi 11:

Set `FuncNonlinear` model attribute:

`FuncNonlinear = 1`

means: "Use MINLP solver"

GUROBI OPTIMIZATION

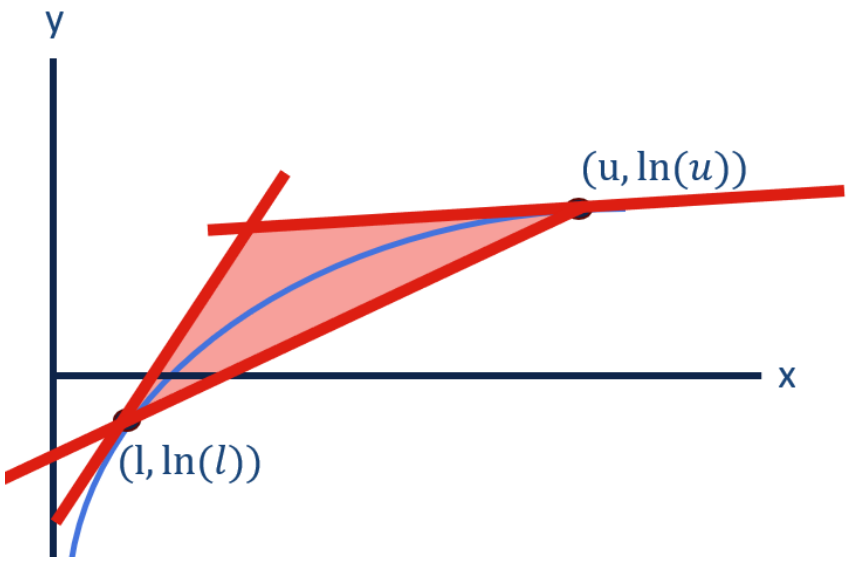
MINLP Solver – How It Works

Uses Branching

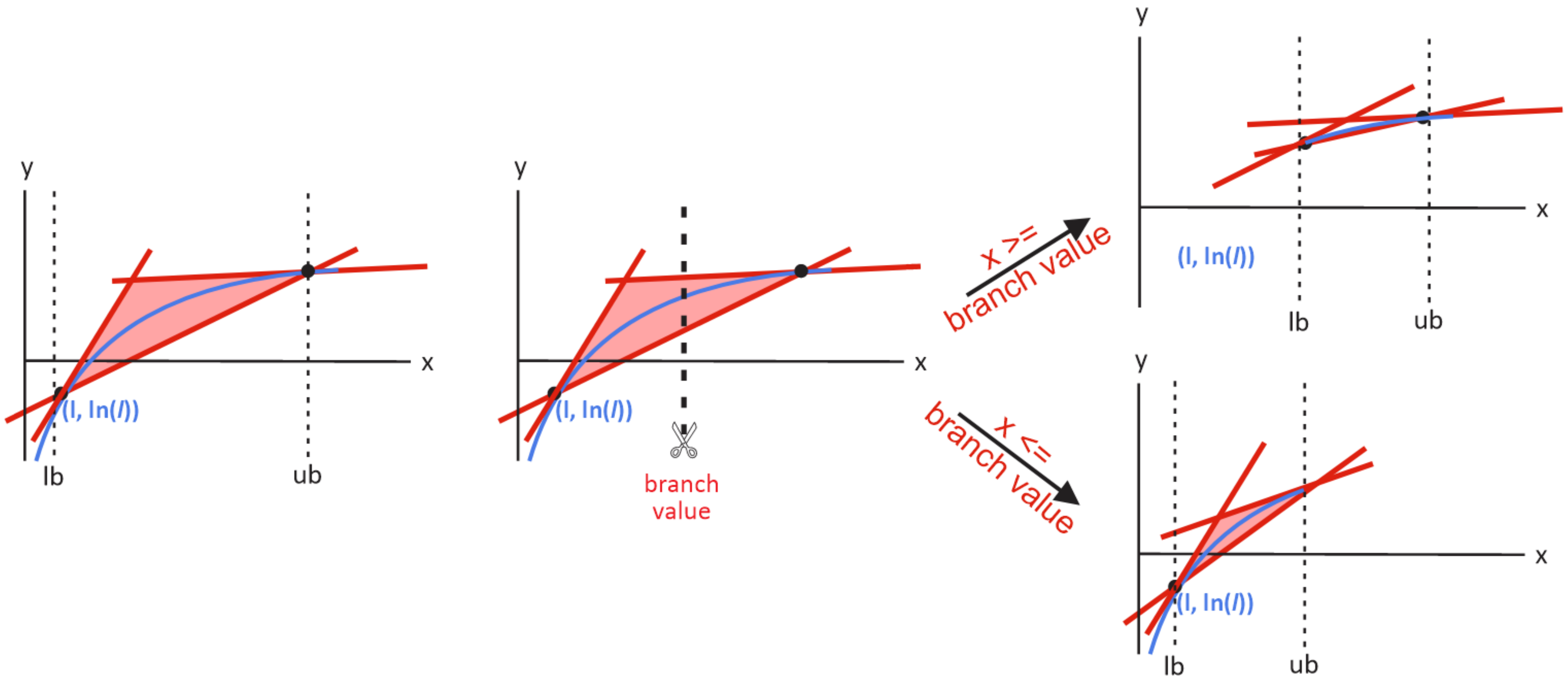
- Similar to branching in MIP
- Solving linear relaxations
- Create sub-problems
- Solve them all

Process:

1. Define lines that bound a curve
2. If lines are within tolerance, stop
3. If not, split the curve into two curves with updated variable bounds and try again
4. Keep going until all lines are within tolerance.



Nonlinear: MINLP Branching



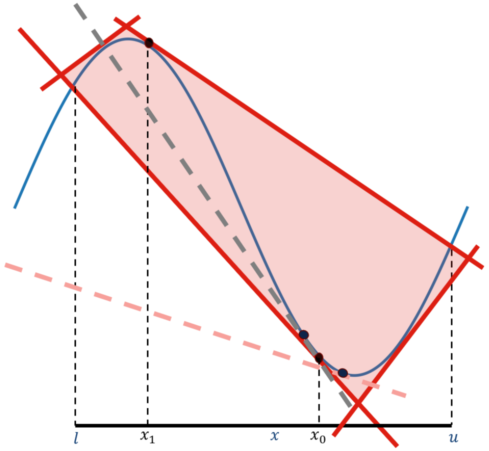
Sub-
problem 1

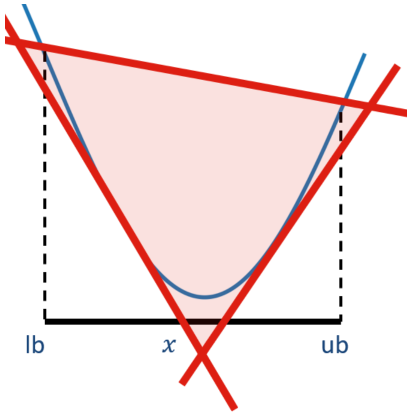
Sub-
problem 2

More Complex Curves

Derives hyperplane cuts

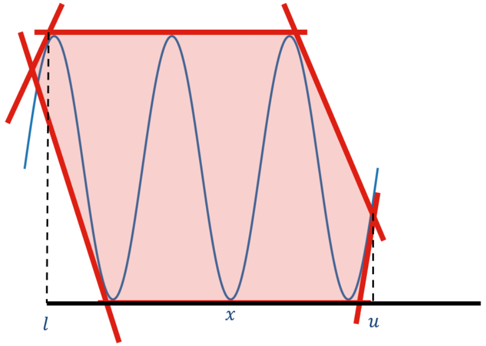
- To add to LP relaxation
- Adding more tangents at various points improves the relaxation.
- Bound the solution space
- Iterative branching
  - Refine the approximation
  - Until difference < tolerance





Options

- Enable Nonlinear constraint:
- FuncNonlinear = 1
- Default: piecewise-linear approximation
- FuncNonlinear = -1

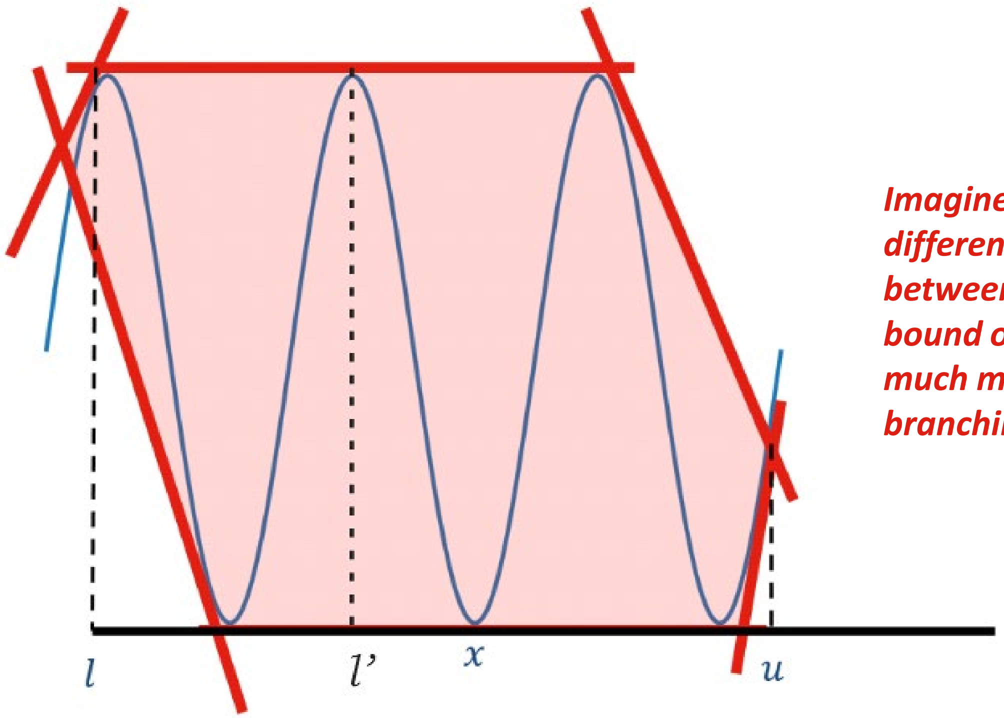


Tighter initial bounds will speed up performance!

Importance of Variable Bounds in MINLP

Tighter bounds = less branching

Tighten your variable bounds



Imagine the difference between lower bound of l vs l' – much more branching with l

GUROBI OPTIMIZATION

Agenda

1. Applications for Nonlinear Solvers
2. Quadratic Solvers in Gurobi
3. Automatic Piecewise Constraints
4. New MINLP Solver
5. **Model Walkthrough**

Sample Model Part 1

$$\text{minimize: } \sin x + \cos 2x$$
$$s.t.: \ 0.25\, e^{x} - x \leq 0$$
$$-1 <= x <= 4$$

```
m = gp.Model()
# Create variables
x = m.addVar(lb=-1, ub=4, vtype=GRB.INTEGER, name="x")
twox = m.addVar(lb=-2, ub=8, name="2x")
sinx = m.addVar(lb=-1, ub=1, name="sinx")
cos2x = m.addVar(lb=-1, ub=1, name="cos2x")
expx = m.addVar(name="expx")
# Set objective
m.setObjective(sinx + cos2x, GRB.MINIMIZE)
# Add constraints
lc1 = m.addConstr(0.25 * expx - x <= 0)
lc2 = m.addConstr(2.0 * x - twox == 0)
# Add general function constraints
# sinx = sin(x)
gc1 = m.addGenConstrSin(x, sinx, "gc1")
# cos2x = cos(twox)
gc2 = m.addGenConstrCos(twox, cos2x, "gc2")
# expx = exp(x)
gc3 = m.addGenConstrExp(x, expx, "gc3")
```

Sample Model Part 2

```python
print("### Use Automatic PWL ...:")
m_pwl, x_pwl = build_model()
m_pwl.params.FuncNonlinear = 0 # (default)
m_pwl.write("automatic_pwl.lp")
m_pwl.optimize()
printsol(m_pwl, x_pwl)
m_pwl.dispose()
print("### MINLP - Set FuncNonlinear=1:")
m_minlp_1, x_minlp_1 = build_model()
m_minlp_1.params.FuncNonlinear = 1
m_minlp_1.write("minlp.lp")
m_minlp_1.optimize()
printsol(m_minlp_1, x_minlp_1)
m_minlp_1.dispose()
```

Sample Models – Log Output

Using Automatic Piecewise Linear:

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 …
Optimize a model with 2 rows, 5 columns and 4 nonzeros
Model fingerprint: 0x40e6a86e
Model has 3 general constraints
Variable types: 4 continuous, 1 integer (0 binary)
…
Presolve added 40 rows and 117 columns
Presolve time: 0.00s
Presolved: 42 rows, 122 columns, 1209 nonzeros
Variable types: 105 continuous, 17 integer (5 binary)
…
Optimal solution found (tolerance 1.00e-04)
x = 2.0
Obj = 1.256238552403841
```

Using MINLP Solver:

```
Set parameter FuncNonlinear to value 1
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 …
Optimize a model with 2 rows, 5 columns and 4 nonzeros
Model fingerprint: 0x40e6a86e
Model has 3 general constraints
Variable types: 4 continuous, 1 integer (0 binary)
Presolve time: 0.00s
Presolved: 17 rows, 6 columns, 34 nonzeros
Presolved model has 3 nonlinear constraint(s)
Solving non-convex MINLP
Variable types: 4 continuous, 2 integer (0 binary)
…
x = 2.0
Obj = 1.2556538059620697
```

GUROBI OPTIMIZATION

Sample Models – LP Saved From Presolve

LP files are identical

Write the presolved models to an LP file

```
presolved_model = model.presolve()
presolved_model.write(“presolved_model.lp”)
```

Compare presolved LP files

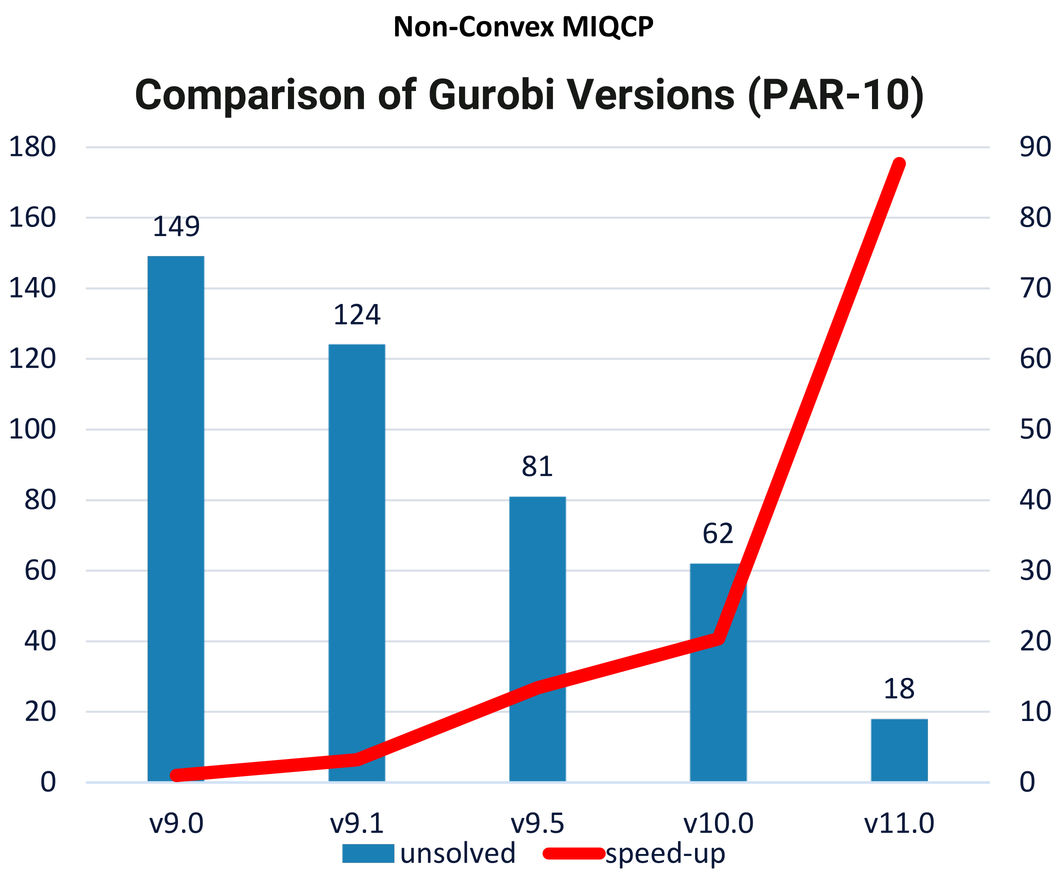
- Piecewise terms are linear
  - Faster to solve
- Piecewise model is bigger
  - Slower to solve

Conclusion

- Some models faster with piecewise
- Others faster with MINLP
- Please try both!

LIVE DEMO

The Future for Gurobi MINLP



Gurobi 11 = first release of MINLP

Optimizations in future releases

- Retest with each new release!

API improvements too

Compare to Non-Convex MIQCP history

- Started with Gurobi 9.0
- Gurobi 11 > 80x faster

Questions?

GUROBI
OPTIMIZATION

Thank you.

GUROBI
OPTIMIZATION

During the Q&A session there were two important questions that were not answered live. Those are documented here:

1. How does IIS work with nonlinear? Is there a way of displaying the model that is equivalent to .lp format?

   Gurobi's computeIIS() API works with a MINLP model to create an Irreducible Inconsistent Subsystem of constraints for an infeasible model.

   You can write a MINLP model to an LP file – as demonstrated at the end of the webinar.

2. What happens if the variable bounds are unknown? Does Gurobi approximate them in the presolve?

   If you have a variable in a MINLP with an infinite bound such as the default: lower bound of zero and upper bound of positive infinity, Gurobi will find the global optimum.

   The model is bounded by a combination of constraints. So, the feasible region is bounded. If it is not, Gurobi returns a result of "unbounded" – and there is no objective value.

Source Code Example:

```
#!/usr/bin/env python3.11
# Copyright 2024, Gurobi Optimization, LLC
# This example considers the following nonconvex nonlinear problem:
#    minimize   sin(x) + cos(2*x) + 1
#    subject to  0.25*exp(x) - x <= 0
#                -1 <= x <= 4
#
#  We show you two approaches to solve it as a nonlinear model:
#
#  Set the paramter FuncNonlinear = 0 to handle all general function
#     constraints as pwl approximations. This is the default in v11.
#
#  Set the paramter FuncNonlinear = 1 to handle all general function
#     constraints as true nonlinear functions.
#
import gurobipy as gp
from gurobipy import GRB

def printsol(m, x):
    print(f"x = {x.X}")
    print(f"Obj = {m.ObjVal}")
def build_model():
    # Create a new model
    m = gp.Model()
    # Create variables
    x = m.addVar(lb=-1, ub=4, vtype=GRB.INTEGER, name="x")
    twox = m.addVar(lb=-2, ub=8, name="2x")
    sinx = m.addVar(lb=-1, ub=1, name="sinx")
    cos2x = m.addVar(lb=-1, ub=1, name="cos2x")
    expx = m.addVar(name="expx")
    # Set objective
    m.setObjective(sinx + cos2x + 1, GRB.MINIMIZE)
    # Add linear constraints
    lc1 = m.addConstr(0.25 * expx - x <= 0)
    lc2 = m.addConstr(2.0 * x - twox == 0)
    # Add general function constraints
    # sinx = sin(x)
    gc1 = m.addGenConstrSin(x, sinx, "gc1")
    # cos2x = cos(twox)
    gc2 = m.addGenConstrCos(twox, cos2x, "gc2")
    # expx = exp(x)
    gc3 = m.addGenConstrExp(x, expx, "gc3")
    return m, x
```

```python
try:
    print("####################################################")
    print("### Use Automatic Piecewise linear approximation:")
    m_pwl, x_pwl = build_model()
    m_pwl.params.FuncNonlinear = 0
    m_pwl.write("automatic_pwl.lp")
    m_pwl_presolve = m_pwl.presolve()
    m_pwl_presolve.write("m_pwl_presolve.lp")
    m_pwl.optimize()
    printsol(m_pwl, x_pwl)
    m_pwl.dispose()
    print("####################################################")
    print("### MINLP - Set FuncNonlinear parameter on Model:")
    m_minlp_1, x_minlp_1 = build_model()
    m_minlp_1.params.FuncNonlinear = 1
    m_minlp_1.write("minlp.lp")
    m_minlp_presolve = m_minlp_1.presolve()
    m_minlp_presolve.write("m_minlp_presolve.lp")
    m_minlp_1.optimize()
    printsol(m_minlp_1, x_minlp_1)
    m_minlp_1.dispose()
    print("####################################################")

except gp.GurobiError as e:
    print(f"Error code {e.errno}: {e}")

except AttributeError:
    print("Encountered an attribute error")
```