Week 6 Reading: Bounds, Relaxations, and Solver Results

A sequence for reviewing the reactor calculation

After computing reactor conditions that meet product specifications, separately assess optimization progress and process prediction accuracy.

Derive the MILP and LP solutions of the two-neuron example, calculate bounds and scaling, then check the OMLT connections and CSTR results.

The lab ZIP contains the network coefficients and recorded results. All LPs and MILPs use open-source HiGHS.

Deriving the triangle relaxation from Big-M

Assume $L < 0 < U$ and $z \in [0, 1]$. Rearrange $h \leq Uz$ and $h \leq a - L(1 - z)$ in terms of z.

$$\frac{h}{U} \leq z \leq \frac{h - a + L}{L}$$

$$h\,(U - L) \leq U\,(a - L)$$

Dividing by negative L reverses the inequality. Retaining $h \geq 0$ and $h \geq a$ gives the lecture triangle relaxation. Keep the valid interval for a as well.

Deriving the integer solution of the toy example

$$0 \le x_1, x_2 \le 1, x_1 + x_2 = 1$$
$$h_1 = \max\left(0, x_1 + x_2 - 1\right), h_2 = \max\left(0, x_1 - x_2\right)$$
$$\min_{x_1, x_2} x_1 \text{subject to} h_1 + h_2 \ge .4$$

The equality gives $x_2 = 1 - x_1$, hence $h_1 = 0$ and $h_2 = \max(0, 2x_1 - 1)$. Satisfying $h_2 \ge 0.4$ requires the active branch with $2x_1 - 1 \ge 0.4$.

The optimum is $(x_1, x_2) = (0.7, 0.3)$, with outputs $(h_1, h_2) = (0, 0.4)$. Compare the objective 0.7 with both MILP implementations.

Calculating the two LP relaxation values

With box bounds [−1, 1], the first triangle permits $h_1 \leq 0.5$ even when $a_1 = 0$. Thus $x_1 = 0$, $x_2 = 1$, $h_1 = 0.4$, $h_2 = 0$ is feasible for the original LP. Since x_1 is nonnegative, its minimum is zero.

Incorporating $a_1 = 0$ into the bounds forces $h_1 = 0$. The second triangle has upper bound $h_2 \leq x_1$, which requires $x_1 \geq 0.4$.

The point $x_1 = 0.4$, $x_2 = 0.6$, $h_2 = 0.4$ is feasible for the tightened LP, proving its optimum is 0.4. Its actual second ReLU output is zero, so it is not an integer-graph solution.

Relaxed solutions versus the actual network

Each formulation minimizes x_1 with $x_2 = 1 - x_1$ and required output sum 0.4.

Formulation	x_1, x_2	Allowed h_1, h_2	Actual ReLU output	Interpretation
Box-bound LP	0,1	0.4,0	0,0	Inflated h_1 meets the limit
Equality-aware LP	0.4,0.6	0,0.4	0,0	Inflated h_2 meets the limit
MILP	0.7,0.3	0,0.4	0,0.4	Matches the original output

After incorporating the equality, the second triangle has upper bound $h_2 \le x_1$. Thus $h_2 \ge 0.4$ requires $x_1 \ge 0.4$. Yet at $x_1 = 0.4$, the actual $a_2 = -0.2$ gives zero ReLU output. Improving the lower bound does not make the relaxation identical to the graph.

Branching and termination in the two-neuron example

After enforcing $h_1=0$, only h_2's binary z_2 remains. Its bounds are $L=-1, U=1$ and $a_2=2x_1-1$. Adding the root LP rows $h_2 \le z_2$ and $h_2 \le 2x_1 - z_2$ gives $h_2 \le x_1$. Since $h_2 \ge 0.4$, $x_1 \ge 0.4$ is a lower bound.

Problem	Calculation	Conclusion
Root LP, $0 \le z_2 \le 1$	$x_1=0.4, h_2=0.4, z_2=0.4$ is feasible	lower bound 0.4; not integer
Child $z_2=0$	$h_2=0$ contradicts $h_2 \ge 0.4$	infeasible; close
Child $z_2=1$	$h_2=2x_1-1 \ge 0.4$	minimum attained at $x_1=0.7$

The children cover both possible binary values. One is infeasible and the other has minimum 0.7, establishing a global lower bound of 0.7. It matches feasible cost 0.7, so the gap is zero.

With several open nodes, the smallest node lower bound identifies a branch that may still improve the incumbent. A branch whose bound reaches the incumbent cannot improve it and can be closed. A real solver may use different branching and cuts.

An interval-propagation calculation

Consider a = $2x_1 - 3x_2 + 0.5$, $x_1 \in [0, 1]$, and $x_2 \in [-1, 2]$. Positive coefficients use the matching extreme; negative coefficients use the opposite extreme.

$$L = 2 \cdot 0 - 3 \cdot 2 + .5 = -5.5$$
$$U = 2 \cdot 1 - 3 \cdot (-1) + .5 = 5.5$$

The ReLU output interval is [0, 5.5]. Both endpoints are attainable over this box, but the calculation can become loose when an equality couples the inputs.

Validity of LP-OBBT bounds

Let S be the exact network feasible set and P an LP relaxation with $S \subseteq P$. Minimization over a larger set cannot increase the optimum; maximization cannot decrease it.

$$\min_P a_j \leq \min_S a_j$$
$$\max_S a_j \leq \max_P a_j$$

The two values over P are valid outward bounds for S. Intersect them with the existing interval bounds. The lab P omits product specifications, so these bounds remain reusable after specification changes within the same input box.

Bound directions in unfinished optimization

A feasible point found while minimizing a gives an upper bound on its minimum. Using that value as a lower bound on a can cut off valid network points, so use a valid lower bound on the minimum.

When maximizing a, a valid upper bound on the maximum is required. A maximization incumbent alone does not supply that upper bound.

The implementation requires optimal termination for every LP and uses `best_objective_bound`. It adds outward padding of 1e-7 × (1 + abs(bound)); this numerical margin does not replace an exact-arithmetic proof.

Suppose minimizing a stops with feasible value −0.2 and valid lower bound −0.6. Its true minimum lies in [−0.6,−0.2]. Use −0.6 as the neuron’s lower bound: using −0.2 could cut off a valid point with a=−0.4. For a maximum, use a valid upper bound instead.

Fixing neurons and changing the domain

The pre-activation a = $x_1 + x_2 - 1$ changes sign over [0, 1] × [0, 1]. With $x_1 + x_2 = 1$, it is identically zero and h can be fixed to zero.

Removing that equality while retaining h = 0 makes the formulation incorrect. At (1, 1), for example, the actual h is 1.

Store the input domain, additional constraints used, and frozen-weight version with the bounds. Recompute bounds after refitting the model or expanding its domain.

Scaled inputs and product specifications

$$x_1 = \frac{T - 310}{40}, x_2 = \frac{\tau - 2}{8}$$
$$T = 310 + 40x_1, \tau = 2 + 8x_2$$

At T = 330 K and τ = 6 min, (x_1, x_2) = (0.5, 0.5). Pass input offsets [310, 2] and factors [40, 8] to OMLT `OffsetScaling`.

If an output were standardized as s = (B − 0.3)/0.1, B ≥ 0.38 would become s ≥ 0.8. The lab uses output factors of 1 and offsets of 0, so physical concentrations equal the OMLT outputs.

Objective transformations and gap calculations

$$J = \tau + \alpha\left(T - 310\right), \alpha = .02\text{min/K}$$
$$J = 2 + 8x_2 + .8x_1$$

An incumbent J of 3.2 and lower bound of 2.8 give an absolute gap of 0.4 and a relative gap of 12.5%. After dropping the constant 2, the same values become 1.2 and 0.8, giving about 33.3%.

The optimizer and absolute gap are unchanged, but the relative gap changes. State the reported objective, its constant term, and the solver gap convention.

Computing gaps with an objective constant

Keep the same incumbent and valid bound, removing only constant 2 from the objective.

Calculation	J with constant	J−2 without constant
Feasible objective	3.2	1.2
Valid lower bound	2.8	0.8
Absolute gap	3.2−2.8=0.4	1.2−0.8=0.4
Relative gap	0.4/3.2=12.5%	0.4/1.2≈33.3%

Subtracting the same number from every candidate preserves ranking and the optimal input. The relative-gap denominator changes, however. Percentages alone cannot compare solve quality when the objective definitions differ.

Without an incumbent there is no value for that denominator. Report the valid bound and absence of a feasible solution; do not interpret initialized variables as the answer.

Constructing OMLT input metadata

Set the scaling and input bounds for the two-input model. Positive factors map both input intervals to [0, 1].

```
scaling = OffsetScaling(
    offset_inputs=[310., 2.], factor_inputs=[40., 8.],
    offset_outputs=[0., 0., 0.],
    factor_outputs=[1., 1., 1.])
definition = NetworkDefinition(
    scaling_object=scaling,
    scaled_input_bounds={0: (0., 1.), 1: (0., 1.)})
```

Pass weights and biases to each `DenseLayer`, then connect layers with `add_edge`. The lab weight axes are [input, output], with input order [temperature, residence time].

A fixed-input validation procedure

1. Select corners and interior points of the input domain.
2. Compute outputs from the original network forward pass.
3. Fix the same inputs in the MILP and remove product specifications.
4. Check termination and solution availability before reading outputs.
5. Compare output differences, constraint residuals, and integrality residuals.

Check the encoding using only the network constraints, with product specifications removed. Retaining the specifications makes the check infeasible at inputs whose outputs violate them. The lab requires a maximum output difference below 1e-5 mol/L.

Define inequality violations to be zero when satisfied: $\max(0,0.38-B)$ for $B\geq 0.38$ and $\max(0,C-0.25)$ for $C\leq 0.25$. For binary value $z=0.999999$, the integrality residual is its distance 0.000001 to the nearest integer 1; compare it with the solver tolerance.

Reference CSTR balances and rate laws

Assume steady state, isothermal operation, constant density, and consecutive first-order reactions A → B → C, with feed A at 1.2 mol/L.

$$C_A = \frac{1.2}{1 + k_1\tau}$$

$$C_B = \frac{k_1\tau C_A}{1 + k_2\tau}, C_C = k_2\tau C_B$$

At 330 K, k_1 = 0.2 and k_2 = 0.1 per minute. The next page defines temperature dependence with educational Arrhenius parameters. Evaluate these reference balances independently of the network approximation.

Temperature dependence and data generation

$$k_i\left(T\right) = k_i\left(330\right)\exp\left(\frac{E_i}{R}\left(\frac{1}{330} - \frac{1}{T}\right)\right)$$

Activation energies are E_1 = 40,000 and E_2 = 50,000 J/mol, with R = 8.314 J/(mol·K). These are specified example parameters rather than estimates for a real chemical system.

Generate 1,200 uniform training inputs over the box. The training seed is 61; independent checks and test generation use seed 87. Test data are not used to fit the coefficients.

Training settings and frozen coefficients

The architecture is 2 → 12 → 12 → 3, with ReLU after the first two affine layers and a linear output layer. The loss is mean squared error across three concentration outputs in physical units.

Run 6,000 full-batch Adam updates: a learning rate of 0.006 for the first 3,999 updates, then 0.002, with β_1 = 0.9, β_2 = 0.999, and ε = 1e-8.

The default coefficients are in `frozen_cstr_network.json`. `--refit` saves a separate model in the specified output directory without overwriting the default. Keep comparison results with the coefficient file used.

Reading results for the same optimization problem

Formulation	LP lower bound	MILP optimum	Binaries
loose	2.0000	3.1656	24
interval	2.0000	3.1656	22
LP-OBBT	2.5215	3.1656	19

OMLT returns the same MILP optimum. Inputs, weights, product specifications, and objective are identical across all four runs.

Each LP lower bound is the optimum of a separately solved continuous relaxation. It can differ from bounds reported after solver presolve and cuts.

Timing scope and reuse of bounds

Formulation	Bounds [s]	Build [s]	MILP [s]	Total [s]
loose	0.0001	0.0016	0.3107	0.3125
interval	0.0001	0.0016	0.1539	0.1556
LP-OBBT	0.1132	0.0016	0.0955	0.2104

Total is bounds plus model construction plus MILP solution. It excludes the separately solved LP relaxation, prediction checks, and training.

Reusing bounds for a fixed model and domain can spread preprocessing cost across solves. Repeated comparisons should use the same hardware, threads, and limits, reporting medians and variability.

Total time when bounds are reused

In the recorded run, interval-bound MILP time was 0.1539 s and LP-OBBT MILP time was 0.0955 s. OBBT preprocessing required 0.1132 s.

Solves m	Interval: 0.0001+m×0.1555	OBBT: 0.1132+m×0.0971
1	0.1556 s	0.2103 s
2	0.3111 s	0.3074 s
5	0.7776 s	0.5987 s

Each per-solve cost includes 0.0016 s for construction. This arithmetic uses rounded recorded times and assumes identical solve times on every repetition; it is not a repeated timing experiment.

Bounds reusable for fixed weights and input domain spread preprocessing cost across solves. A new model or expanded domain requires renewed bound validation, so this calculation cannot automatically be carried over.

Exercise: scaling and gaps

Scale T = 340 K and τ = 4 min, then calculate J. Next calculate absolute and relative gaps for incumbent 5.0 and lower bound 4.5.

Check: (x_1, x_2) = (0.75, 0.25), and J = 4 + 0.02 × 30 = 4.6. The gaps are 0.5 and 10%.

What should be reported if only lower bound 4.5 is available? Check: report the valid lower bound and absence of an incumbent; leave the relative gap and selected inputs unavailable.

Exercise: bounds and reference-process checks

After computing bounds over the input box, the temperature upper limit is raised from 350 K to 360 K. May the old bounds be retained? Check: recompute bounds or establish their validity over the expanded domain.

At the selected input, network B is 0.380000 and reference CSTR B is 0.371157 mol/L. Calculate the specification margin. Check: reference minus lower limit is -0.008843 mol/L, which is negative.

Reducing the solver gap alone cannot correct this violation. The predictor or selected conditions require reconsideration.

Primary sources and lab files

Ceccon et al., OMLT: Optimization & Machine Learning Toolkit, JMLR 23(349), 2022 · Official quick-start

Anderson et al., Strong mixed-integer programming formulations for trained neural networks, Mathematical Programming, 2020

HiGHS and its MIT license · Pyomo HiGHS interface

`results.json` records termination and checks, `comparison.csv` contains the comparison table, and `frozen_network.json` stores the coefficients used. The verified environment is Python 3.12.14, NumPy 2.5.4, Pyomo 6.10.1, OMLT 1.2.2, HiGHS 1.15.1.

Lecture · Lab ZIP