Week 4: Modeling III, Blending, Pooling and Quality Specifications

2105623 Optimization of Chemical Processes

Assoc. Prof. Dr. Soorathep Kheawhom

Department of Chemical Engineering, Chulalongkorn University

Semester 1/2026

Outline

1. Lecture 1: linear blending and quality specifications
2. Lecture 2: the pooling problem
3. Lecture 3: local versus global in practice

Session plan, 180 minutes

Time	Block	Duration
0:00 to 0:10	Opening: recap of Week 3, HW1 collected, today's objectives	10 min
0:10 to 0:40	**Lecture 1**: blending as a linear model, quality specifications, the multiply-through rule	30 min
0:40 to 1:00	**Activity 1**: specification negotiation (two halves of the room, paper)	20 min
1:00 to 1:30	**Lecture 2**: the pooling problem, bilinear terms, p- and q-formulations, nonconvexity	30 min
1:30 to 1:45	Break	15 min
1:45 to 2:10	**Activity 2**: blend it (computer, individual)	25 min
2:10 to 2:35	**Lecture 3**: local versus global in practice, multistart, when to distrust a converged answer	25 min
2:35 to 2:55	**Activity 3**: the pooling trap (computer, groups of 3 to 4)	20 min
2:55 to 3:00	Wrap-up, takeaways, what is due next week	5 min

This is the designed crossover session of the course: OpenSolver before the break, Python and `ipopt` after it. Notebook `W04_modeling3_blending_pooling.ipynb`; workbook `W04_blending_STUDENT.xlsx`.

Learning objectives

After this session you should be able to

1. Formulate a multi-product blending problem with quality specifications as a linear program, and explain why the specification rows must be written in unnormalized form.
2. Apply the multiply-through-by-the-denominator rule, state the three conditions under which it is exact, and recognize the cases where each fails.
3. Explain the structural reason a pool introduces bilinear terms, and write both the p-formulation and the q-formulation.
4. Demonstrate that a local NLP solver returns different answers from different starting points, and quantify the penalty of accepting a local solution.
5. Compare the McCormick relaxations of the two formulations and report the bound each one delivers.

CLO 1 (formulation) and CLO 4 (implementation and solution in Pyomo). Reference: Rao, Ch. 1 and Ch. 10; Edgar, Himmelblau and Lasdon, Ch. 7; Haverly (1978); Tawarmalani and Sahinidis, Ch. 9.

Blending: the most common linear program in the process industries

The question

Choose the least costly, or most profitable, mixture of purchased component streams that still meets *every* quality specification of *every* finished product.

- Gasoline and diesel blending, the classic case.
- Feed blending, animal nutrition, cement raw meal.
- Electrolyte formulation: salt concentration, water content, conductivity, viscosity.
- Cathode precursor blending: metal ratios, impurity ceilings.

The one modeling assumption that makes it linear

Properties blend linearly on a mass basis. Good for sulfur and density, acceptable for RVP if a blending index is used, and only rough for octane, where real blending is measurably nonlinear.

The second, usually unstated, assumption

Every component reaches the blender as a *separate* stream, perfectly segregated up to the blend header. Lecture 2 removes exactly this assumption, and the model stops being linear.

The instance used all session

Five component streams

Component	RON	RVP kPa	S ppm	cost USD/t	available t
Reformate	98.0	3.0	110	760	3200
FCC naphtha	92.0	6.5	380	730	4000
Alkylate	95.0	4.5	15	805	2000
Butane	93.0	62.0	5	510	900
Straight-run naphtha	68.0	9.0	520	695	2600

Two finished grades

	Regular	Premium
RON minimum	91.0	95.0
RVP maximum, kPa	9.0	8.5
Sulfur maximum, ppm	290	190
Price, USD/t	742	806
Minimum volume, t	4000	2500
Maximum volume, t	7000	5000

The same instance appears in the notebook (Section 1) and in `excel/W04_blending_SOLUTION.xlsx`. The two routes must agree to the cent.

The linear blending model

Decision variables

$x_{cg} \geq 0$ tonnes of component c sent to grade g;
$y_g \geq 0$ tonnes of grade g produced.

Model

$$
\begin{aligned}
\max \quad & \sum_g \text{price}_g \, y_g - \sum_c \sum_g \text{cost}_c \, x_{cg} & \\
\text{s.t.} \quad & \sum_c x_{cg} = y_g & \forall g \\
& \sum_g x_{cg} \leq \text{avail}_c & \forall c \\
& d_g^{\min} \leq y_g \leq d_g^{\max} & \forall g \\
& \sum_c \text{ron}_c \, x_{cg} \geq \text{ron}_g^{\min} \, y_g & \forall g \\
& \sum_c \text{rvp}_c \, x_{cg} \leq \text{rvp}_g^{\max} \, y_g & \forall g \\
& \sum_c \text{sul}_c \, x_{cg} \leq \text{sul}_g^{\max} \, y_g & \forall g
\end{aligned}
$$

Row by row

- objective: revenue minus component purchase cost, in USD
- grade material balance
- component availability
- contract volume window
- octane floor
- vapor pressure ceiling
- sulfur ceiling

Look at the last three rows

Every specification row has the decision variable y_g on its *right-hand side*. That is not an accident, and the next slide explains it.

Quality specifications are ratios, and the ratio is removed by hand

The physical requirement

$$\frac{\sum_c \text{ron}_c \, x_{cg}}{\sum_c x_{cg}} \;\geq\; \text{ron}_g^{\min}$$

is a ratio of decision variables and therefore nonlinear.

The rule

Multiply through by the denominator, which is y_g and is non-negative:

$$\sum_c \text{ron}_c \, x_{cg} - \text{ron}_g^{\min} \, y_g \;\geq\; 0,$$

or equivalently $\sum_c (\text{ron}_c - \text{ron}_g^{\min}) x_{cg} \geq 0$.

Why this is not an approximation

The linear form is not a linearization of the ratio form. It is *the same feasible set*, intersected with $y_g > 0$. Solving the ratio form with `ipopt` returns the same optimum to within solver tolerance, while the constraint reports a polynomial degree of `None` instead of 1.

On a spreadsheet

The specification `RHS` cell contains a changing cell, `=C11*C26`. That is what "multiply the ratio through by its denominator" looks like on a sheet, and the plain `RHS` column convention does not anticipate it.

Three conditions, and each of them fails somewhere

Let a, b be data vectors and x the decision vector. Then $(a^{\mathsf{T}}x)/(b^{\mathsf{T}}x) \le \beta$ is equivalent to $a^{\mathsf{T}}x - \beta\, b^{\mathsf{T}}x \le 0$ **provided that** $b^{\mathsf{T}}x > 0$ **at every point of interest**.

The three conditions

1. **Sign**. Multiplying an inequality by a negative number reverses it. The rule needs $b^{\mathsf{T}}x > 0$, not merely $b^{\mathsf{T}}x \neq 0$. In blending the denominator is a total mass and the contract window forces it strictly positive.
2. **Zero denominator**. If $b^{\mathsf{T}}x = 0$ is attainable, the ratio is undefined while the multiplied-through form reads $0 \le 0$ and is satisfied trivially. The linear model is then a *strict relaxation*: it permits a blend of size zero whose quality is meaningless. If a grade may be shut off, the honest model is a disjunction and needs a binary variable, which is Week 5.
3. **Same balance**. The denominator must be a linear function of the *same* variables that appear in the numerator. When the denominator belongs to a different balance, as in a pool, the product of two decision variables survives and the model is genuinely nonlinear.

The practical test, one question

Is the denominator the same aggregate that the numerator is normalized by, in the same balance? If yes, the constraint is linear in disguise. If no, expect bilinear terms.

Translation table: reusable, and worth copying into your notes

Engineering requirement	Natural form	Linear equivalent	Condition
average property of a blend at most β	$\sum_c q_c x_c / \sum_c x_c \le \beta$	$\sum_c (q_c - \beta) x_c \le 0$	$\sum_c x_c > 0$
average property at least β	$\sum_c q_c x_c / \sum_c x_c \ge \beta$	$\sum_c (q_c - \beta) x_c \ge 0$	$\sum_c x_c > 0$
recycle ratio at most r	$R/(R+F) \le r$	$(1-r)R - rF \le 0$	$R + F > 0$
conversion at least η	$(F_{\text{in}} - F_{\text{out}})/F_{\text{in}} \ge \eta$	$F_{\text{out}} \le (1-\eta) F_{\text{in}}$	$F_{\text{in}} > 0$
fixed split fraction θ	$S_1/(S_1+S_2) = \theta$	$(1-\theta)S_1 - \theta S_2 = 0$	$S_1 + S_2 > 0$
purity of a mixed stream at least π	$m_{\text{key}}/m_{\text{tot}} \ge \pi$	$m_{\text{key}} - \pi\, m_{\text{tot}} \ge 0$	$m_{\text{tot}} > 0$
ratio *objective*	$\max (c^{\mathsf{T}} x + c_0)/(d^{\mathsf{T}} x + d_0)$	Charnes-Cooper LP	$d^{\mathsf{T}} x + d_0 > 0$
property of a stream leaving a pool	$p = \sum_i q_i F_i / \sum_g P_g$, and p multiplies P_g	**no linear equivalent**	denominator is in another balance

The last row is the whole of the rest of this session.

A ratio in the objective is a different objective

Charnes-Cooper

Maximizing $(c^{\mathsf{T}}x + c_0)/(d^{\mathsf{T}}x + d_0)$ over $\{Ax \le b,\ x \ge 0\}$ with $d^{\mathsf{T}}x + d_0 > 0$ is a linear-fractional program. Substituting

$$t = 1/(d^{\mathsf{T}}x + d_0) > 0, \qquad x' = tx$$

gives the equivalent LP

$$\max\ c^{\mathsf{T}}x' + c_0 t$$

$$\text{s.t.}\quad Ax' - bt \le 0, \quad d^{\mathsf{T}}x' + d_0 t = 1, \quad x', t \ge 0,$$

and $x = x'/t$ recovers the original solution.

On this instance

Total margin (Section 1 LP)	413,439.75 USD
Margin per tonne (CC LP)	45.8708 USD/t
its plan	4,000.00 t Regular, 5,000.00 t Premium
its total margin	412,837.10 USD
margin given up	602.65 USD

Read this carefully

The plant that maximizes the ratio *deliberately declines* the last 80.6 t of Regular, because that increment earns less than the current average. Which objective is correct is a business question, not a mathematical one, but the two must never be confused.

The linear optimum, and why binding specifications are the real output

Optimum: 413,439.75 **USD**

Component, t	Regular	Premium	slack
Reformate	107.625	3092.375	0
FCC naphtha	2623.629	1376.371	0
Alkylate	833.010	0	1166.990
Butane	206.667	367.834	325.499
Straight-run naphtha	309.669	163.420	2126.911
Volume y_g, t	4080.599	5000.000	

All six quality rows bind exactly

Regular sits at RON 91.00, RVP 9.00 kPa, 290.00 ppm sulfur; Premium at RON 95.00, RVP 8.50 kPa, 190.00 ppm.

That is normal, and it is the point of a blending model: **quality above specification is quality given away**, so the optimizer gives away nothing.

Marginal values

Availability duals: reformate 41.9412 and FCC naphtha 13.3609 USD/t; alkylate, butane and straight run have slack and a dual of zero. The optimum is degenerate: several recipes achieve the same objective, so compare objectives, not recipes.

Activity 1 | 20 min | two halves of the room, paper: specification negotiation

The situation, given to everyone

The refinery blends the **five component streams** of slide 6 into **two grades**, Regular and Premium. Component costs, availabilities and properties are as tabulated. Regular sells at 742 and Premium at 806 USD/t. Contractual volumes: Regular between 4,000 and 7,000 t, Premium between 2,500 and 5,000 t. The plan must cover one month.

Brief for the Production half

Your objective is **minimum cost**, that is maximum blending margin. You are accountable for meeting the contracted volumes from the components actually on site, and for not buying more of the expensive streams than you must. Write down the constraint set *you* would insist the model contain. Anything you leave out, you are agreeing to give away.

Brief for the Quality Control half

The product must meet three named specifications per grade. **Regular:** RON $\geq$ 91.0, RVP $\leq$ 9.0 kPa, sulfur $\leq$ 290 ppm. **Premium:** RON $\geq$ 95.0, RVP $\leq$ 8.5 kPa, sulfur $\leq$ 190 ppm.

The **octane floor is a customer contract term**: a delivery below it is a rejected cargo. The RVP ceiling and the sulfur ceiling are **internal laboratory targets**, set inside the legal limits with a deliberate margin. Write down the constraint set *you* would insist on.

Grouping two halves of the room, working in pairs within each half. **Time** 20 min: 10 min separately, then compare and agree *one* constraint set, explicitly labeling each constraint hard or soft. **Hand back** the agreed constraint list, one sheet per side, with H or S written against every row.

The agreed set

Constraint	H / S
grade material balance $\sum_c x_{cg} = y_g$	H (physics)
component availability $\sum_g x_{cg} \le \text{avail}_c$	H (physics)
contract volume window $d_g^{\min} \le y_g \le d_g^{\max}$	H (contract)
octane floor $\sum_c \text{ron}_c x_{cg} \ge \text{ron}_g^{\min} y_g$	H (contract)
vapor pressure ceiling	S (internal target)
sulfur ceiling	S (internal target)
non-negativity	H

A soft constraint belongs in the objective

$$\sum_c \text{sul}_c x_{cg} - \text{sul}_g^{\max} y_g \le v_g, \quad v_g \ge 0, \qquad \text{objective} \; - \; \pi_g v_g .$$

The penalty π_g is the price of the violation, and choosing it is an engineering judgement that must be written down, not hidden.

The teaching point

Hard and soft is a **modeling decision**, not a property of the problem. Nothing in the physics distinguishes the octane floor from the sulfur ceiling. What distinguishes them is who wrote them down and what happens when they are missed.

What the numbers say once all three are hard

With every specification treated as hard the margin is 413,439.75 USD and **all six quality rows bind**. The value of relaxing each row by one unit (the LP dual of the multiplied-through row, identical for both grades) is

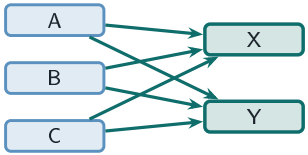
octane row	0.69276
vapor pressure row	5.12983
sulfur row	0.13507

So the internal RVP target is by far the most expensive of the three to hold, and it is precisely the one nobody in the room called a contract term.

Real plants do not have that many tanks

Segregated: linear

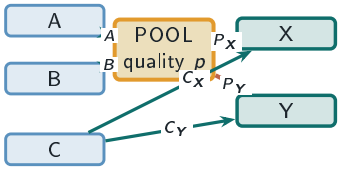
each product sees the *individual* feed qualities



Quality delivered to X is $\sum_i q_i x_{iX}$, a linear function. Multiply the ratio through by $\sum_i x_{iX}$ and the model is an LP.

Pooled: bilinear

rust arcs carry pP_g: variable times variable



Everything drawn from the tank has the *same* composition p, whatever it is fed by. p is a decision, and it multiplies flows.

One physical fact destroys linearity

Streams are collected in a common pool. The pool quality p satisfies $p\left(\sum_g P_g\right) = \sum_i q_i F_i$, and the quality delivered to product g is pP_g. Both expressions contain the product of two decision variables.

Why multiplying through does not rescue it

Condition 3 of slide 9, failing

The pool quality is the ratio

$$p \;=\; \frac{\sum_i q_i F_i}{\sum_g P_g}.$$

Its denominator is the **total pool throughput**. But the quantity that has to be constrained is the pool's contribution to the balance of a *different* stream, namely product g:

$$p\,P_g + q_C\,C_g \;\le\; \mathrm{spec}_g\,(P_g + C_g).$$

Multiplying by $\sum_g P_g$ clears the ratio and leaves pP_g untouched.

Consequence

The feasible set is no longer convex. The problem is a **nonconvex NLP with bilinear terms**, and every guarantee that came with linear programming is gone:

- no vertex theorem,
- no duality certificate of optimality,
- no reason a converged point is the best one.

Where this bites in practice

Refinery gasoline pools, crude tank farms, LPG caverns, hydrogen headers, and any electrolyte make-up tank feeding more than one cell chemistry.

Haverly's instance, 1978: small enough to reason about by hand

Data

Stream	route	S %	USD
Feed A	into the pool	3.0	6
Feed B	into the pool	1.0	16
Feed C	bypasses the pool	2.0	10
Product X	≤ 2.5 % S		9
Product Y	≤ 1.5 % S		15

Feeds are costs, products are prices. Demand limits $X \leq 100$, $Y \leq 200$ units.

Variables

A, B into the pool; P_X, P_Y out of the pool; C_X, C_Y direct; and the pool sulfur p. Seven in total. The bounds $1 \leq p \leq 3$ are physical: the pool is fed only by streams of 1 and 3 percent sulfur.

The p-formulation

$$
\begin{aligned}
\max\ & 9(P_X+C_X) + 15(P_Y+C_Y) \\
& \quad - 6A - 16B - 10(C_X+C_Y) \\
\text{s.t. } & A + B = P_X + P_Y \\
& 3A + B = p(P_X+P_Y) \\
& p\,P_X + 2C_X \leq 2.5(P_X+C_X) \\
& p\,P_Y + 2C_Y \leq 1.5(P_Y+C_Y) \\
& P_X+C_X \leq 100, \quad P_Y+C_Y \leq 200 \\
& 1 \leq p \leq 3, \quad \text{all flows} \geq 0
\end{aligned}
$$

The rows are, in order: pool mass balance, pool sulfur balance, product X specification, product Y specification, demand limits, bounds. **Three constraints are bilinear**, and each one contains a product of two decision variables.

The q-formulation: carry proportions instead of quality

Change of variables

Let q_i be the fraction of the pool inlet contributed by feed i, so $\sum_i q_i = 1$ and $0 \le q_i \le 1$. Then $F_i = q_i \sum_g P_g$ and the amount of feed i reaching product g is $q_i P_g$.

$$
\begin{aligned}
\max\ & 9(P_X{+}C_X) + 15(P_Y{+}C_Y) \\
& - 6q_A(P_X{+}P_Y) - 16q_B(P_X{+}P_Y) - 10(C_X{+}C_Y) \\
\text{s.t. } & q_A + q_B = 1 \\
& (3q_A{+}q_B)P_X + 2C_X \le 2.5(P_X{+}C_X) \\
& (3q_A{+}q_B)P_Y + 2C_Y \le 1.5(P_Y{+}C_Y) \\
& P_X{+}C_X \le 100, \quad P_Y{+}C_Y \le 200, \quad 0 \le q_i \le 1
\end{aligned}
$$

What changed and what did not

- Every bilinear term is now a product $q_i P_g$ of a **bounded proportion** and a flow.
- The pool mass balance disappears: it is implied by $\sum_i q_i = 1$.
- The two formulations describe the **same feasible set** and the **same global optimum**, since $p = \sum_i q_i s_i$.

But they are not the same relaxation

And that, not the algebra, is the whole reason for preferring one over the other. Reparametrizing a nonconvex problem does not make it convex.

Nonconvexity: what a solver can and cannot promise you

What `ipopt` actually asserts

A termination status of `optimal` from a local NLP solver is a statement about the Karush-Kuhn-Tucker conditions at the point it stopped. It is not, and never was, a statement about global optimality.

What that means operationally

- Two runs of the same solver on the same model, differing only in the starting values, can converge to two different objective values, both reported as `optimal`.
- There is no gap, no bound and no warning in the output to tell them apart.
- A spreadsheet is worse, not better: Solver silently abandons Simplex LP for GRG Nonlinear and reports "Solver found a solution. All constraints and optimality conditions are satisfied." That sentence is true and misleading.

What a global solver adds

A spatial branch-and-bound code (couenne, BARON, ANTIGONE) maintains a **convex relaxation** of the nonconvex problem, so it can report an incumbent *and* a bound, and therefore a gap. That is the only kind of guarantee available here.

The discipline this forces

1. Never report a single local solve as *the* optimum.
2. Always report the starting point with the answer.
3. Run a multistart.
4. Report the relaxation bound alongside the incumbent, so the reader knows how much room is left.

McCormick envelopes: how a bilinear term is relaxed

The construction

Replace a product $w = uv$ with $u \in [u^L, u^U]$, $v \in [v^L, v^U]$ by a new variable w and four linear inequalities:

$$w \geq u^L v + v^L u - u^L v^L, \quad w \geq u^U v + v^U u - u^U v^U,$$
$$w \leq u^U v + v^L u - u^U v^L, \quad w \leq u^L v + v^U u - u^L v^U.$$

This is exactly the convex hull of the graph of uv over the box, so it is the tightest possible relaxation of one product in isolation.

Reformulation-linearization (RLT)

In the q-formulation the bilinear variables $v_{ig} = q_i P_g$ inherit the identity $\sum_i q_i = 1$. Multiplying it by P_g and by the pool throughput limit gives

$$\sum_i v_{ig} = P_g \quad \forall g, \qquad \sum_g v_{ig} \leq \text{Cap}\, q_i \quad \forall i.$$

These rows are **redundant for the nonlinear model** but **not for its relaxation**.

Why this is the practical lever

Relaxing every bilinear term turns the pooling NLP into an LP whose optimum is a valid bound on the global optimum. The bound is what a spatial branch-and-bound tree spends its time closing, so choosing the formulation with the tighter relaxation is worth more than any solver setting.

Break, 15 minutes

Next: open `excel/W04_blending_STUDENT.xlsx` with OpenSolver loaded, *and* check that `ipopt` is on your path.

Activity 2 starts at 1:45 on the spreadsheet. Activity 3, at 2:35, is the reason this course leaves the spreadsheet behind.

Activity 2 | 25 min | computer, individual: blend it

Part 1, reproduce the optimum and read off what binds

Open `excel/W04_blending_STUDENT.xlsx`, sheet `Model`. Complete the yellow cells: the availability `VALUE` column `E19:E23`, the constraint `VALUE` column `E33:E44` (every quality row in multiplied-through form, so its `RHS` is 0), and the objective `C29`. Run OpenSolver with CBC: objective `$C$29` set to **Max**, changing cells `$C$19:$D$24`, constraints as listed on the `Readme` sheet, and tick *Get sensitivity report*.

The objective cell must read 413,439.75 USD. Then write down, from the check block in rows 46 to 51, **which specifications bind**.

Part 2, predict before you solve

Change *one* specification by a small amount, for example tighten the Regular sulfur ceiling from 290 to 250 ppm. **Before re-solving**, write on paper: will the margin rise or fall, and roughly by how much? Use the sensitivity report from Part 1 to justify the number. Then re-solve and compare.

Grouping individual. **Time** 25 min. **Hand back** nothing: the instructor walks the room. Anyone whose objective differs has a formula error.

Activity 2 | [ANSWER] | the binding set is the interesting output

Part 1

Objective cell `C29` = 413,439.75 USD, matching the Pyomo model to the cent (413,439.749277).

Binding: all six quality rows; the Premium volume ceiling at 5,000 t; and the availability rows for reformate (3,200 t) and FCC naphtha (4,000 t).

Not binding: the Regular volume window (it settles at 4,080.60 t, above its floor of 4,000 and below its ceiling of 7,000), and the availability of alkylate, butane and straight-run naphtha.

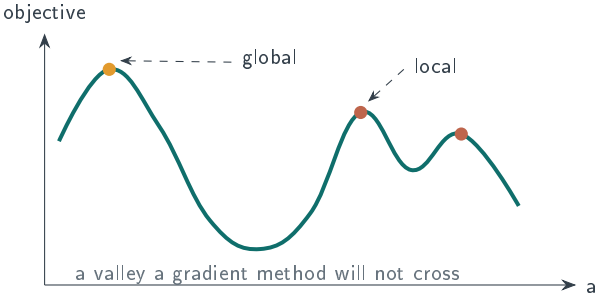
Part 2, the prediction rule

Predicted change = (dual of the row) × (change in the row). Tightening a ceiling that binds can only reduce the margin, so the sign is known before any arithmetic. The magnitude is exact **only while the basis does not change**: push the specification far enough and a different set of constraints binds, so the linear prediction over-states the loss. Notebook Exercise 2 makes exactly this comparison at 290 → 150 ppm.

The question to the room

Which specifications bind, and why is that the interesting output rather than the recipe? Because the optimum here is **degenerate**: several different recipes achieve the same margin, and a different solver may report a different one. The binding set and the objective are stable; the recipe is not.

What a nonconvex landscape looks like, and why a solver cannot see across it



Schematic. Every marked point satisfies the KKT conditions, and a local solver reports `optimal` at each.

The mechanism

A gradient-based method climbs the hill it starts on. It has no information about the landscape beyond the neighbourhood it has visited, and no mechanism for crossing a valley of infeasible or worse points.

So the starting point selects the answer

On a nonconvex problem the initial values are not a numerical detail, they are **part of the specification of what you computed**. A result reported without them is not reproducible.

Multistart: the poor person's global search

The procedure

1. Sample N starting points over the physically meaningful ranges of the variables that drive the nonconvexity.
2. Solve from each, keep every terminal point, not just the best.
3. Report the histogram of terminal objective values, the best value, the worst value, and the fraction of starts reaching the best.

What it does and does not give you

It gives evidence and a good incumbent. It gives **no bound**, so it cannot prove optimality. A multistart that always returns the same value has found either a convex problem or a badly chosen sampling range.

Practical advice

- **Prefer a physical starting point.** Initializing a pool quality from the actual current plant operation is worth a hundred random draws.
- Sample the *quality* and *split* variables. Sampling flows alone often lands in the same basin every time.
- Fix the random seed and record it.
- If the problem is small, use a global solver and stop guessing.

When a plant engineer should distrust a converged answer

Six warning signs

1. A product of two decision variables appears anywhere in the model, including in a spreadsheet cell.
2. A quality, composition, temperature or split fraction is itself a variable.
3. A recycle or a shared tank connects two balances.
4. The solver engine changed from linear to nonlinear without you asking.
5. The answer moves when you change only the initial values.
6. The reported solution is much better than anything operations has ever achieved.

The consequence for refinery blending

A blend recipe computed from a local optimum is not wrong in the sense of being infeasible: it meets every specification. It is wrong in the sense of **leaving margin on the table, silently, every day, with no signal that it is doing so**.

And for electrolyte blending

The same structure appears whenever one make-up tank feeds more than one cell chemistry: the tank composition is a variable, and it multiplies every draw from the tank.

Activity 3 | 20 min | computer, groups of 3 to 4: the pooling trap

What to do

Open `notebooks/W04_modeling3_blending_pooling.ipynb` and run everything up to and including the Haverly section. The model builder is `build_haverly(p0=...)`, where `p0` is the **starting value of the pool sulfur** p, and the solver is `ipopt`.

Solve the *same model* **twice**: once from `p0 = 1.0`, once from `p0 = 3.0`. Nothing else changes: same data, same constraints, same solver, same tolerances.

Record, for each of the two runs

1. the solver's termination status,
2. the objective value,
3. the final pool quality p,
4. the six flows A, B, P_X, P_Y, C_X, C_Y.

Logistics

Grouping groups of 3 to 4, one machine per group.
Time 20 min.
Hand back nothing written. Each group **reports its two numbers out loud** when polled, and they go on the board.

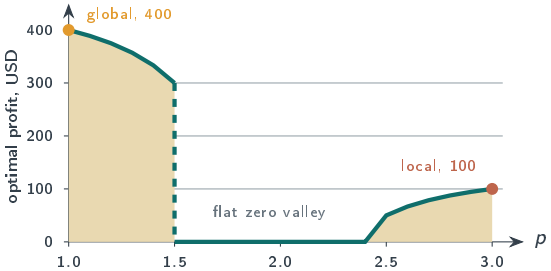
Do not compare with the group next to you until the poll. When both columns are on the board, the question to the room is simply: *which one is right?*

The two runs, same model, same solver

	start $p_0 = 1.0$	start $p_0 = 3.0$
termination	optimal	optimal
profit, USD	400.00	100.00
final p, % S	1.000	3.000
A	0	50
B	100	0
P_X	0	50
P_Y	100	0
C_X	0	50
C_Y	100	0

The cost of accepting the wrong one

300 USD out of 400, that is 75 **percent of the achievable margin**, on a problem with seven variables. Both points genuinely satisfy the KKT conditions, so both status words are honest.



Fixing p turns the pooling problem back into an ordinary LP, so sweeping p traces the true profile. Two separated hills, and a valley over which no feasible plan makes any money.

Multistart over nine starting values of p

starting p_0	profit	final p
1.00	400.00	1.00
1.25	400.00	1.00
1.50	400.00	1.00
1.75	400.00	1.00
2.00	100.00	3.00
2.25	100.00	3.00
2.50	100.00	3.00
2.75	100.00	3.00
3.00	100.00	3.00

Four of the nine starts reach the better solution. Every run reports `optimal`.

Why the split falls where it does

On the notebook's 81-point sweep of p over $[1, 3]$, the profit is **exactly zero for p from** 1.525 **to** 2.400. That flat valley is the boundary between the two basins of attraction, and it is why no gradient step crosses it.

The left hill is reached by feeding the pool with the low-sulfur stream B and selling the high-value product Y. The right hill is reached by feeding the pool with the cheap high-sulfur stream A and selling only the low-value product X.

The closing point of the session

A spreadsheet solver would have returned *one* of these two numbers and never mentioned the other. No message, no bound, no gap. **That is why the course moves to Python from here**, and the reason is not size: it is that a spreadsheet cannot tell you its answer is local.

Closing the loop: what a relaxation bound would have told the room

McCormick and RLT bounds against the global optimum

Instance	form	global	bound	gap
Haverly, 2 feeds, no pool limit	p	400	500.00	2
Haverly, 2 feeds, no pool limit	q	400	500.00	2
extended, 4 feeds, pool ≤ 150	p	400	555.56	38.9
extended, 4 feeds, pool ≤ 150	q	400	466.67	1

- On Haverly the two bounds coincide: a single pool fed by two streams and limited only by product demand leaves the RLT constraints nothing to add.
- On the extended instance the q-formulation cuts the gap from 38.9 to 16.7 percent, for the cost of extra rows. That is why global solvers build the pq-formulation internally.

The honest report

"Profit 400" is an opinion.

"Profit 400, from $p_0 = 1.0$, with a valid upper bound of 500 from the McCormick relaxation, and a multistart over nine starts in which four reached 400 and five reached 100" is a result.

Forward

Week 10 supplies the KKT theory behind the word `optimal`; Week 11 supplies the branch and bound machinery that closes a gap; Week 13 returns to what to do when the model is not known at all.

Summary

Concepts

- A quality specification is naturally a ratio. Multiplying by the blend volume, which belongs to the same balance, makes it linear. That single manipulation is why blending is an LP.
- The multiply-through rule is exact only when the denominator is strictly positive and lives in the same balance as the numerator.
- A ratio objective is a linear-fractional program. Charnes-Cooper turns it into an LP, and its plan is generally not the plan that maximizes the total.
- A pool forces every product drawn from it to see the same composition. The pool quality multiplies flows belonging to other balances, bilinear terms appear, and the feasible set stops being convex.
- `optimal` from a local NLP solver asserts the KKT conditions, not global optimality.
- The p- and q-formulations share a feasible set and a global optimum but not a relaxation. The bound is what a global solver works to close.
- Report the relaxation bound with the incumbent. An answer to a nonconvex problem without a bound is an opinion.

Numbers established today

Linear blending margin	413,439.75 USD
Regular / Premium volume	4,080.60 / 5,000.00 t
Binding quality rows	all six
Availability duals, reformate / FCC	41.9412 / 13.3609 USD/t
Spec row duals, RON / RVP / S	0.69276 / 5.12983 / 0.13507
Margin per tonne (Charnes-Cooper)	45.8708 USD/t
its total margin	412,837.10 USD
margin given up	602.65 USD
Haverly from $p_0 = 1.0$	400.00 USD, $p = 1$
Haverly from $p_0 = 3.0$	100.00 USD, $p = 3$
Multistart, 9 starts	4 reach 400, 5 reach 100
Flat zero valley on the sweep	p in [1.525, 2.400]
McCormick, Haverly p / q	500.00 / 500.00
McCormick, extended p / q	555.56 / 466.67

Coming next

Week 5: logical and discrete decisions, big-M versus convex hull, and formulation strength.

Before next week

Reading

- Edgar, Himmelblau and Lasdon, Ch. 7 (blending and refinery planning models).
- Rao, Ch. 1 for problem classification and Ch. 10 for the nonlinear background.
- Haverly (1978), the original two-page note that named the pooling problem.
- Tawarmalani and Sahinidis, *Convexification and Global Optimization*, Ch. 9, for the p- and q-formulations.

Notebook to finish

`W04_modeling3_blending_pooling.ipynb`, all sections, plus **Exercises 2, 4 and 6**. Exercise 4 is the proper 100-point multistart on Haverly and is the one that settles the argument started today.

Homework

HW1 was due today. HW2 is released today and is due in Week 6. It covers blending, pooling, logical and discrete modeling, and it includes a *model-audit* task: you receive a plausible but wrong generated model and must find and correct its errors.

Further reading

- **Haverly, C. A.** (1978), "Studies of the behaviour of recursion for the pooling problem", *ACM SIGMAP Bulletin*. The instance used in Activity 3, and still the standard minimal counterexample.
- **Tawarmalani, M. and Sahinidis, N. V.**, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, Ch. 9: the p-, q- and pq-formulations and the relaxations they induce.
- **Biegler, L. T.**, *Nonlinear Programming: Concepts, Algorithms and Applications to Chemical Processes*, for what `ipopt` is actually doing and what its convergence test means.
- **Williams, H. P.**, *Model Building in Mathematical Programming*, 5th ed., Ch. 3 and the refinery case studies, for blending formulations written out in full.
- **Misener, R. and Floudas, C. A.** (2009), "Advances for the pooling problem", for the modern survey of formulations and solution methods.

The governing rule of the modeling block, in its sharpest form: a spreadsheet is the right tool when the data is tabular and the model is small; Python becomes necessary when the model must be re-solved many times or verified programmatically.