2105623 Optimization of Chemical Processes

Department of Chemical Engineering, Chulalongkorn University

OpenSolver Setup Guide

Semester 1/2026 | Companion to the workbooks in `excel/` | Weeks 1 to 5

Purpose. This guide gets every student to a working spreadsheet optimizer before the Week 1 session ends, and then explains how to read what the optimizer reports. Keep it beside you while working through `W01_product_mix_STUDENT.xlsx` to `W05_fixed_charge_STUDENT.xlsx`.

OpenSolver is a free, open-source Excel add-in from the University of Auckland, distributed at `opensolver.org`. It reads a model laid out in worksheet cells, exactly the way Excel's own Solver does, and hands it to a real mathematical programming code; the bundled default engine is COIN-OR CBC, a branch-and-cut solver for linear and mixed-integer linear programs. It reads the same hidden model definition the built-in Solver writes, it has no size limit, and Show Model lets you audit the model that was actually built.

1 Installing OpenSolver on Windows

1. Confirm Excel 2007 or later for Windows. OpenSolver does not work in Excel Online, the mobile apps, LibreOffice, or Google Sheets.
2. Download the OpenSolver Advanced zip from `opensolver.org`. That build carries the bundled CBC binaries; the linear-only build does not carry everything Week 5 needs.
3. **Unblock the zip before extracting it.** Right-click the `.zip`, choose Properties, tick Unblock on the General tab, click OK. Windows marks downloaded files, and Excel silently refuses to load a macro-enabled add-in that still carries the mark. This is the commonest cause of a failed installation.
4. Extract to a permanent folder you will not move or rename, for example `C:\OpenSolver`. Do not run it from inside the zip, from Downloads, or from a synchronized cloud folder.
5. Double-click `OpenSolver.xlam`; an OpenSolver group appears on the Data ribbon. To load it at every start, use File → Options → Add-ins → Manage: Excel Add-ins → Go → Browse, select the file, and confirm its box is ticked.
6. Verify with Data → OpenSolver → About OpenSolver: a CBC version number must be reported. An empty solver line means the install failed.

2 Installing OpenSolver on macOS

OpenSolver supports Excel for Mac 2016 and later. Two steps exist here that do not exist on Windows, and both stop students every year.

1. Expand the macOS build and copy `OpenSolver.xlam` into your Excel add-ins folder, which on a current system is `~/Library/Group Containers/UBF8T346G9.Office/User Content/Add-Ins`, or use Tools → Excel Add-ins → Browse. Restart Excel; an OpenSolver item appears on the Data tab.
2. **The security prompt for the bundled CBC binary.** The first time you press Solve, macOS Gatekeeper intercepts the attempt to run the bundled CBC executable and Excel's sandbox asks for access to the folder holding it. Expect two dialogs: an Excel Grant Access dialog naming a folder, which you must accept, and a macOS dialog saying the developer cannot be verified. If the second offers only Move to Trash or Cancel, choose Cancel, open System Settings → Privacy & Security, find the message naming the blocked binary, and click Allow Anyway. Press Solve again and confirm Open. This is once per machine, but until you do it every solve fails with a solver-not-found error even though the file is present.
3. **Apple Silicon and Rosetta 2.** The CBC and Bonmin binaries shipped with OpenSolver are compiled for x86_64, so on an M-series Mac they run under the Rosetta 2 translation layer. If Rosetta 2 is absent the solve fails immediately. Install it once with
   `softwareupdate --install-rosetta --agree-to-license`
   and restart Excel. Translation costs some speed, immaterial for the course models and one more reason the large re-solve experiments live in Python. Verify through About OpenSolver.

3 Enabling the built-in Excel Solver, and when it is enough

The built-in Solver ships with Excel but is not enabled by default. On Windows: File → Options → Add-ins → Manage: Excel Add-ins → Go → tick Solver Add-in. On macOS: Tools → Excel Add-ins → tick Solver. Enable it even if you intend to use OpenSolver: the two coexist and share the same hidden model definition, and Week 1 deliberately starts with the built-in Solver so that every student succeeds on day one without an install.

The Solver that ships with Excel is a size-limited edition of Frontline Systems' product. It accepts at most **200 decision variables** and **100 explicit constraints**, where simple bounds on the variables do not count toward the 100. Exceeding either limit produces *The problem is too large for Solver to handle*: the model is refused outright, not solved approximately. Two hundred sounds generous until you count. The Week 4 blending model has 5 components and 2 grades, so 12 variables; extend it to 20 components, 6 grades and 12 monthly periods and you have $20 \times 6 \times 12 = 1440$, which the built-in Solver will not accept at all. That is the point of the closing question of Week 2, *what happens at 52 periods*.

Built-in Solver is adequate when the model has fewer than roughly 200 variables and

100 constraints, is linear or smoothly nonlinear, and you want the sensitivity report with no installation.

OpenSolver is needed when the model exceeds those limits, when it contains binary or integer variables and you want a real branch-and-cut search rather than the built-in evolutionary engine, or when you want to inspect the model that was actually built.

Python with Pyomo is needed when the model must be built from data rather than typed, re-solved many times, or verified programmatically. That threshold is reached in Week 4 and never released.

4 Setting up one model, step by step

The sequence below uses `W04_blending_STUDENT.xlsx`, whose Readme sheet repeats these ranges. Every workbook in `excel/` follows the same layout.

1. **Complete the sheet first.** Fill every yellow cell with the formula the structure implies and check that the green changing cells hold numbers, not text. A solver cannot repair a broken sheet; it optimizes the wrong thing instead.
2. **Open the dialog.** Select the Model sheet, then Data → OpenSolver → Model. The fields are usually pre-filled, because the workbook already carries a stored model definition.
3. **Objective cell.** Enter `$C$29` and select Maximise. It must be one cell holding a formula, never a constant and never a range.
4. **Changing cells.** Enter `$C$19:$D$24`. If the variables are not contiguous, list several ranges separated by commas, as `$C$40:$E$45,$H$40:$H$45` in Week 5.
5. **Constraints, added as ranges.** Never add constraints one cell at a time. Each family occupies a block of rows whose VALUE column holds `SUMPRODUCT` formulas and whose RHS column holds data, so the family is one entry. Click Add and enter in turn `$E$19:$E$23` $\leq$ `$G$19:$G$23` (component availability), `$E$33:$E$34` $=$ `$G$33:$G$34` (grade material balance), `$E$35:$E$36` $\geq$ `$G$35:$G$36` (octane floor), `$E$37:$E$40` $\leq$ `$G$37:$G$40` (vapor pressure and sulfur ceilings), `$E$41:$E$42` $\geq$ `$G$41:$G$42` and `$E$43:$E$44` $\leq$ `$G$43:$G$44` (the contract volume window). The left and right ranges of one entry must have the same shape. For integrality choose the relation int or bin and leave the right-hand side empty; Week 5 needs `$H$40:$H$45` bin.
6. **Non-negativity.** Tick Make unconstrained variable cells non-negative. Every model in Weeks 1 to 5 assumes $x \geq 0$, and forgetting this box is the second commonest source of a nonsense answer, after a mistyped range.
7. **Choose the engine.** Set the solver to COIN-OR CBC: simplex for linear programs, branch and cut for mixed-integer linear programs. Do not select a nonlinear engine for a linear model, which trades a global guarantee for a local one at no gain.

8. **Request the sensitivity report.** Tick Get sensitivity report; OpenSolver writes a new worksheet after the solve. The checkbox is meaningful only when the model has no integer variables.

9. **Solve**, read the status bar, and **check against an independent computation.** Every workbook states the objective value its Pyomo counterpart returns. For Week 4 the target is 413,439.75 USD; for Week 5 it is 2,533.226 kUSD/yr.

5 Reading the sensitivity report

The report is written to a new sheet and has two tables. It describes the *optimal basis*, not the whole model, so every number in it is valid only while that basis does not change.

Variable cells table. Final Value is the optimal value of that decision variable, the number now in the green cell. Reduced Cost is the change in the objective per unit if that variable were forced into the solution against the optimizer's judgment; a variable positive at the optimum has reduced cost zero, and one sitting at zero with reduced cost -18 in a maximization tells you that forcing one unit in costs 18, so its price must improve by 18 before it becomes attractive. Objective Coefficient is the cost or profit coefficient currently in the model. Allowable Increase and Allowable Decrease say how far that one coefficient may move, all else fixed, before the optimal solution changes: inside the interval the recipe stays put and only the objective value moves, and outside it another vertex becomes optimal.

Constraints table. Final Value is the left-hand side at the optimum, the number in the VALUE column. Shadow Price is the change in the objective per unit increase in that right-hand side. This is the dual variable and the most useful number in the report: in the Week 4 workbook the binding reformate availability row has a shadow price of 41.94 USD/t, so one more tonne is worth 41.94 USD above its purchase price. A non-binding constraint has shadow price zero, because relaxing something that is not restricting you is worth nothing. Constraint R.H. Side is the right-hand side currently in the model, and Allowable Increase and Allowable Decrease give the range over which it may move while the shadow price stays valid. A shadow price is a derivative and therefore local: quoting it for a change larger than the allowable increase is the commonest error in economic interpretation.

The 1e100 entries. Values of `1E+100`, sometimes written `1e100`, appear in the allowable-increase and allowable-decrease columns. They are neither data nor an error: OpenSolver uses 10^{100} as its stand-in for infinity, in the role Excel's own Solver gives 10^{30}. On a constraint, `1E+100` means the right-hand side may be raised without limit and the shadow price will not change, which happens when the row is slack and its shadow price is already zero. On an objective coefficient in a maximization, it means raising that activity's profit never displaces the current solution, because the activity is already used as heavily as the rest of the model permits. Read `1E+100` as the word *unbounded*, never as a magnitude.

6 Troubleshooting

Symptom	What it means, and what to do
The model is infeasible	No point satisfies every constraint at once. The model, not the solver, is at fault. Zero the changing cells and read the VALUE column by hand, looking for a $\geq$ that should be $\leq$, a right-hand side in the wrong units, demand above total supply, a minimum above a maximum. Remove constraint families one at a time; the last one removed is the conflict.
The model is unbounded	The objective can be improved without limit, so no optimum exists. Almost always a missing constraint or a missing non-negativity box. Check that Make unconstrained variable cells non-negative is ticked, that every activity carries a capacity or availability row, and that you have not maximized what you meant to minimize.
Solver not found, or an empty solver line in About OpenSolver	Excel cannot execute the bundled CBC binary. On Windows the zip was not unblocked before extraction, or the folder was moved: re-extract after unblocking. On macOS the Gatekeeper prompt was declined: approve the binary under Privacy & Security → Allow Anyway. On Apple Silicon, install Rosetta 2.
Results not updating: the answer does not change after you edit the data	Either automatic calculation is off, or you edited a cell the model does not use. Set Formulas → Calculation Options → Automatic, press Ctrl+Alt+F9, and re-solve. Then use Show Model and confirm the highlighted cells are the ones you edited. A hardcoded number typed over a formula is the usual cause.
Sensitivity report unavailable for a model with binaries	A sensitivity report is defined only for a linear program. It comes from the optimal simplex basis, and a mixed-integer program has no such basis: its answer comes from a branch-and-cut tree and the objective is not a differentiable function of the right-hand sides. Untick Get sensitivity report for any model with int or bin constraints. For marginal information either relax integrality and read the duals of the resulting LP, remembering that they price the relaxation and not the true problem, or re-solve the integer model with a perturbed right-hand side and take the difference. Week 5 uses the second method.
The problem is too large for Solver to handle	The built-in Solver's limit of 200 variables or 100 constraints has been exceeded. Switch to OpenSolver, which has no such limit. If the model is also too large to lay out by hand, that is the signal to move it to Pyomo.

Symptom	What it means, and what to do
No OpenSolver group on the Data ribbon	The add-in is not loaded for this session. Re-tick it under Excel Add-ins and check that macros are enabled for the folder holding `OpenSolver.xlam`.

7 Which tool is used in which week, and why

The course alternates between the spreadsheet and Python on purpose. The governing rule is stated once and applied everywhere: *a spreadsheet is the right tool when the data is naturally tabular and the model is small; Python becomes necessary when the model must scale, be re-solved many times, or be verified programmatically.* Week 4 is the designed crossover point.

Week	Primary tool	Secondary	Rationale
1	Excel Solver	Pyomo mirror of the same model	First contact. Everyone succeeds on day one without an install.
2	OpenSolver	Pyomo comparison at the end	Periods across columns read naturally. Close by asking what happens at 52 periods.
3	OpenSolver	Python for shortest path and max flow	A cost matrix is a spreadsheet. Graph structures are not.
4	Excel then Python	ipopt, multistart	Linear blending in OpenSolver, then pooling forces the move. **The crossover lesson.**
5	Mixed	CBC through OpenSolver, then Pyomo	Fixed charge with binaries in Excel, then formulation-strength comparison needs dozens of re-solves.
6	Python only		Auditing a model requires reading code.
7 to 13	Python	Excel sensitivity report in Week 8	Algorithms and scale.

Two rows deserve a sentence of their own. In **Week 4** the pooling variant of the blending plant introduces bilinear terms: a spreadsheet solver returns a locally optimal blend without telling you it is local, and establishing globality needs a convex relaxation and many restarts. In **Week 5** the fixed-charge model is built in Excel, because the semicontinuous encoding is easiest to see in a sheet, but comparing big-M against the convex hull means solving the MILP and its relaxation for several formulations and several values of M and recording the gap and the solve time each time. Both are loops, and a loop belongs in Python.

Raise installation questions in the first ten minutes of Week 1, not in Week 4. Every workbook states the objective value its Pyomo counterpart returns; if your sheet disagrees, the sheet is wrong.