Don't Let Recycle Streams Stymie Your Simulations

Some simple tips and techniques can help you converge flowsheet recycles.

Ryan C. Schad,
Eastman Chemical Co.

Most chemical engineers today use process simulators in a variety of applications. Once relegated to process design engineers, simulators now can be found on the desks of environmental engineers, process improvement engineers, and even plant engineers. Modern simulators have progressed to the point where you as an engineer need little or no programming skill to model a complex flowsheet. Problems can still arise, however, when a flowsheet contains one or more recycles that necessitate a trial-and-error solution method called convergence. Although flowsheet simulators have built-in ways of solving recycle problems, you ultimately are responsible for providing sensible specifications that will lead to meaningful simulation results. In this article, I will describe practical tips and techniques to help you characterize and converge simulation flowsheet recycles — with an emphasis on "what to do" more than "what theory to use."

A quick review

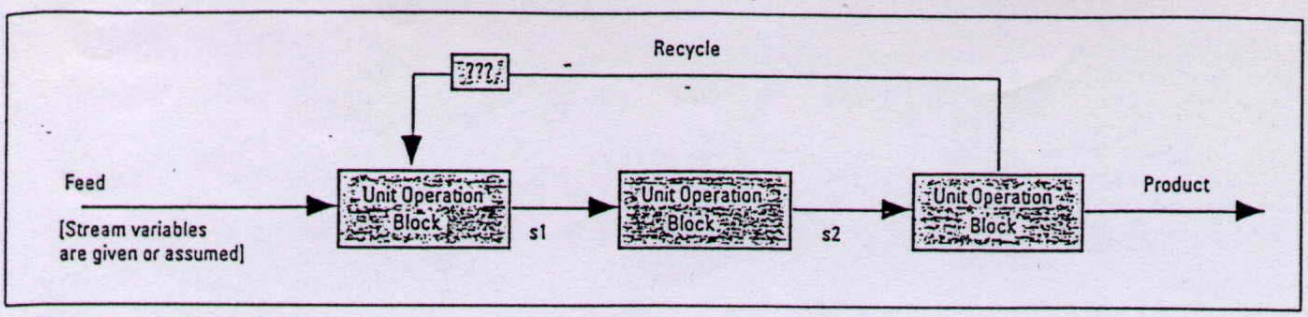
Most process simulators are sequential-modular, which means that calculations start with known feeds and continue on a unit-by-unit basis. This calculation style is advantageous, because it requires that we solve at one time only a small number of equations associated with a single unit operation and its associated streams. (It also is the way you would solve a flowsheet by hand.) Some process simulators are "equation-based," and can solve everything associated with a flowsheet simultaneously — unit operations, recycles, and optimization problems. Here, we will assume that you have the more-common sequential-modular simulator (like ASPEN PLUS, for instance).

A "recycle" stream (see Figure 1) complicates the simulation problem because we don't know, at the outset, the flow and composition of the recycle stream. As a result, we will need to employ a trial-and-error solution method. If we guess values (flows, compositions, pressure, and so on) for the recycle stream, then we can proceed with sequential calculations. When we solve equations for the block where the recycle stream originated, we can compare our calculated variables of the stream with the values that we guessed. If they are close enough, we are finished; if not, we must update our guess and cycle through the calculations again. This process of guessing, calculating, comparing, and reguessing stream variables is called flowsheet convergence.

The stream for which we guess values is called the "tear stream" (because that stream has been *torn*). The desired scaled difference between guessed and calculated tear stream values is referred to as "tolerance." The difference between the guessed and calculated tear stream values for a particular iteration is referred to as "convergence error." The order in which we solve the flowsheet is called the "sequence," while the numerical method used to arrive at a better next guess for the tear stream is known as the "convergence method."

Usually, the equations used to solve specific unit-operation blocks are so complex that a similar iterative solution

-194-



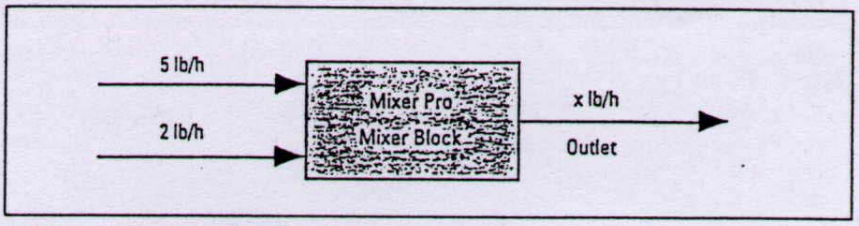
■ *Figure 1. A simple block flowsheet.*

method is required to solve them. When these calculations are complete, the block also is said to be "converged." It is not block convergence on which this article focuses; however, it is important to understand the distinction between flowsheet and block convergence because they sometimes interact with each other.

Many aspects related to convergence — such as how many and which streams should be torn, what sequence to use for the flowsheet, and which convergence method to employ — are subjects of continuing research. More detailed and comprehensive discussions on flowsheeting are given in Refs. 1 and 2. Experienced users of process simulators, however, can naturally acquire keen skills relating to practical convergence techniques. Most of the tips and heuristics that follow are empirical in nature, a result of converging a variety of complex flowsheets using a commercial simulator.

Garbage in equals garbage out

Flowsheet convergence is needed not only when a process has recycle streams, but also when flowsheets have design specifications or optimization problems. (Design specifications arise when we want to set an output of a block or a product stream by varying some input, but aren't sure of the input value to arrive at the desired output.) Because many flowsheets have at least one of these features, all modern simulators automatically can choose tear streams, convergence methods, and calculation sequences for you. For a majority of your simulation runs, this



■ *Figure 2. A simple mixer flowsheet.*

will work fine, and you will experience no difficulties. If convergence problems develop, however, the simulator will report an error message that often is general and cryptic. You, the user, will have to intervene to solve the problem. Here, we will focus on recycle loop problems, but many of the techniques apply equally well to design or optimization specifications.

Actually, most convergence difficulties begin with the user. Convergence problems generally are caused by poor input variables, improper output specifications, programming errors, or faulty physical properties. Ideally, we want to avoid such dubious input to begin with, but it helps to be able to interpret convergence warning signs and determine how they relate to the potential shortcomings of the flowsheet.

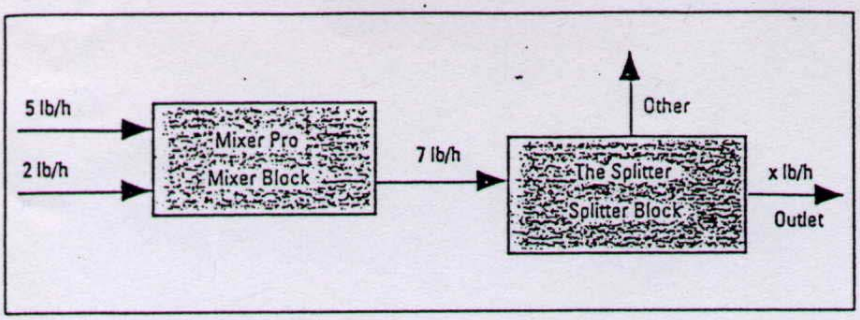
On the other hand, sometimes your process simulator does a poor job of automatically choosing tear streams, convergence methods, or the calculation sequence (especially if your model is quite complex). Therefore, you have the option of controlling any and all aspects of the convergence process — such as simply adding initial guesses for tear streams or changing convergence methods, all the way up to specifying the order in which multiple tear streams are solved. It is beyond the scope of this article to instruct you when and how to control the many aspects of convergence. Note, however, that you can add significant power to your simulation with just a little customization, which is something that beginning and casual users tend to avoid.

Another thing that users sometimes avoid is generating a good block flowsheet of the process. (See Figure 1.) This should, at the minimum, detail the following: types and names of unit operation blocks, types and names of streams (*e.g.*, heat, material), and flowsheet connectivity. Most commercial process simulators now have graphical interfaces that generate the block flowsheet automatically. An effective process flowsheet is essential for evaluating and debugging a convergence situation.

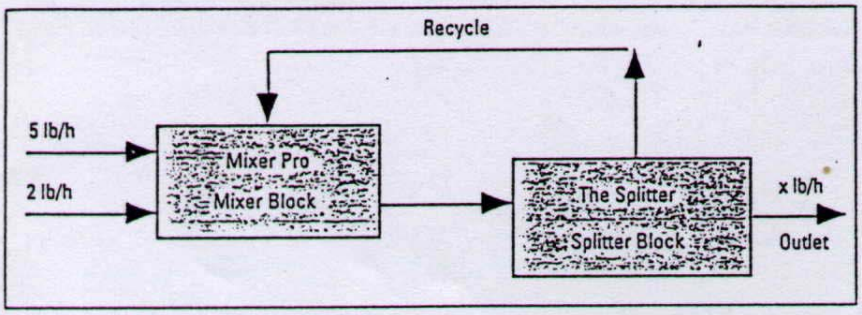
In equals out

Believe it or not, most convergence problems are caused by chemical engineers neglecting the most basic of all mass balance fundamentals.

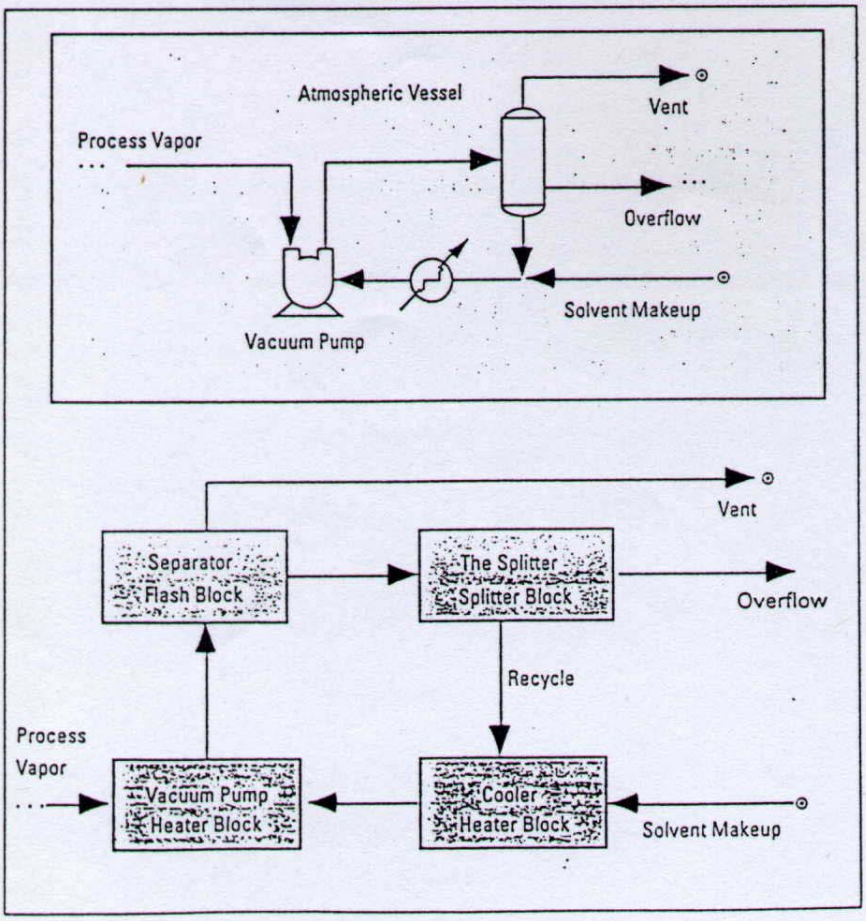
Say you have a mixer combining two user-specified streams, as in Figure 2. You can see by inspection that "outlet" stream flow must be 7 lb/h. In fact, you could not specify that this stream flow be *anything*



■ *Figure 3. Mixer/splitter flowsheet.*



■ *Figure 4. Mixer/splitter/recycle flowsheet.*



■ *Figure 5. Vacuum system schematic and block flowsheet.*

other than 7 lb/h without creating an anomaly.

Figure 3 presents a similar situation. We now have some freedom to specify our unknown outlet stream flow without causing any problems (provided, of course, our specification is less than 7 lb/h). The "other" stream from our flow splitter will carry off enough mass to complete the balance and must remain unspecified.

Finally, let's turn our attention to Figure 4, which looks deceptively similar to Figure 3. Although the splitter has two exiting streams as before, we no longer are free to specify the flow of the outlet stream, because it must be exactly 7 lb/h to satisfy the overall mass balance (but flow in our recycle stream can be any value). You probably think that you would never make a mistake like this (that is, improperly specifying a stream constrained by mass balance), but the fact is that a majority of convergence problems arise from this very mistake.

Imagine that you were simulating this flowsheet. Because you are permitted to specify the outlet stream flow to be anything that you please while setting up your model, let's say that you specify 6 lb/h. (Your simulator will not catch this mistake *a priori*.) While attempting to solve the flowsheet, your simulator would tear the recycle stream (or the splitter feed). A good initial guess for recycle flow, as far as your simulator is concerned, is 0 lb/h. Cycle through a few convergence iterations in your head, starting with the mixer and keeping track of the flow rate of the recycle stream. Your result will look like this: 0 lb/h... 1... 2... 3..., and so on — this flowsheet is doomed never to converge. After some maximum number of iterations, your simulator will give up and report some kind of convergence error to you.

You can solve this problem in one of two ways: if you must specify an exact flow rate, specify the flow of recycle instead; another effective solution is to specify the flow of the outlet stream to be a *ratio* of the block inlet feed rate.

 -196-

In general, it is not a good idea to specify the exact outlet flow — which is 7 lb/h here — because, in a real flowsheet, the mixer feed streams most likely will be outputs from other parts of the flowsheet and, thus, unknown at the start. An exact specification like 7 lb/h only works for a unique case like that shown in Figure 4, while the other recommended solutions will work for all cases.

Using a ratio works well because it allows the magnitude of the flow of the recycle (which is the tear stream) to directly determine the mass purge rate from the loop. This usually will lead to quick flowsheet convergence. the way, "flows" can mean component flows or overall stream flows.

It can't happen to you, right?

Before you scoff that you never would make such a mistake, consider the vacuum pump system shown in Figure 5 — because it illustrates one of the many forms in which this type of flowsheet convergence mistake can disguise itself. Assuming this vacuum system is a small part of a complete process being simulated, the process vapor stream will come from some other vessel; you will not know its composition or flow rate when you are inputting the vacuum system block specifications.

It probably doesn't seem unreasonable to specify the "overflow" stream flow. Perhaps you have designed a similar system and know about how much flow to expect, say 10 gal/min. If you were to specify the exact overflow stream flow, you would be making the same mass-balance error explained above. Here's why: the vacuum pump solvent recycle creates a simple convergence problem whose solution will be governed essentially by the mass balance around the vacuum system. Inlets to the system are "solvent makeup" and "process vapor;" outlets are "vent" and "overflow." Depending on the pressure of the vacuum pump, a nontrivial quantity of higher boilers could have been burped over into the vacuum system through process vapor. The vent stream essentially is saturated noncondensibles at atmospheric pressure, which means that everything else — solvent, low-boilers, and high-boilers — must exit the loop through the overflow stream. Unless you have been *extremely* lucky to guess the exact amount of solvent plus process material that must be purged in overflow, your simulation likely will not converge.

If you guessed an "overflow" flow that is too low, your simulator will guess higher and higher recycle flows on subsequent convergence iterations in a vain attempt to increase the amount of process material that exits through the vent. After the maximum number of iterations, your simulator will give up and report an error. You may notice simultaneously that the recycle flow in your vacuum system is abysmally high. (A note of caution: if the recycle flow is allowed to become three to four orders of magnitude greater than the purge stream due to this convergence ramping effect, the loop may report convergence because the mass balance error is less than the convergence tolerance!) Likewise, if you guess too small an overflow flow, your simulator on subsequent iterations will "drain" your recycle flow (reducing the vent flow) until the maximum number of iterations is again reached.

The correct and robust solution is the same as before: specify the recycle flow rate in your loop, or specify the overflow flow as a ratio of the inlet flow.

It takes some practice to look at a large process block flowsheet and spot all critical "mass balance" (or, more correctly, "mass- and enthalpy balance" envelopes). Unfortunately, many more blocks can separate inlet and outlets of a mass balance loop than in the examples above. Because a large majority of convergence problems are related to this problem of illegal specifications within mass balance loops, it is a particularly valuable skill to note what streams can and cannot be specified exactly. The unit operations that are most likely to be illegally specified are flow splitters, flash blocks, nonrigorous separator blocks, and distillation blocks.

Tip: Do not specify or limit key outlet flows from convergence loops. (Let them "float.") Instead, specify recycle flows.

Symptoms: Recycle flow rates within a convergence loop "blowup" or "drain" (that is, become extremely large or small) during simulation, usually followed by a convergence error. Alternatively, a loop won't converge given any number of iterations. (This is indicative of other problems as well.)

Loop Structure 101

It is handy, but not necessary, to be able to analyze from experience a moderately complicated flowsheet and realize how many recycles are present. The number of recycles usually corresponds to the number of tear streams that will be required to simulate the flowsheet. Here is a short test to see if you have mastered this capability: study Figure 6 and decide the minimum number of recycle, or tear, streams needed to simulate this flowsheet. Now do the same for Figure 7.

You might be surprised to discover that both Figure 6 and 7 are identical flowsheets, with *no* recycles. Sometimes the way a block diagram of a flowsheet is sketched can give the appearance of recycles where there actually are none. Of course, flowsheets with true recycles cannot be redrawn to eliminate them!

Simulators already employ sophisticated techniques to determine the minimum number of tear streams for you, so the ability to successfully analyze the recycle structure of a flowsheet is not critical. It still can be helpful, however, for several reasons. For one, if you ever need to sequence part or all of your simulation (for instance, because the flowsheet is especially complicated, because you have design specifications that must be solved in a set order, or because you wish to "nest" some convergence loops), you

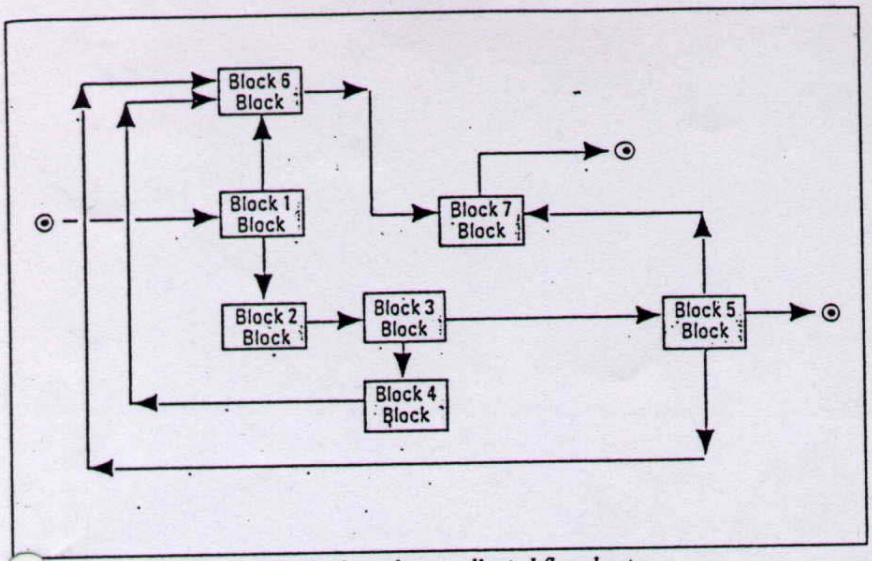


Figure 6. An example of a moderately complicated flowsheet.

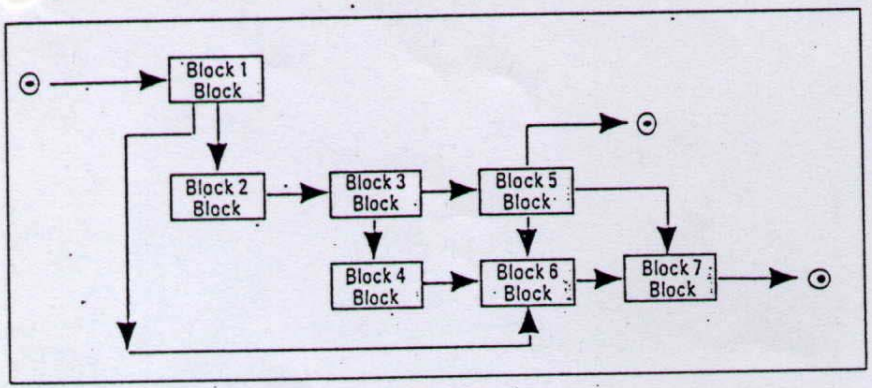


Figure 7. Another moderately complicated flowsheet.

will need to be able to specify your flowsheet solution sequence. For another, it will help you recognize mass balance envelopes, to avoid or troubleshoot convergence difficulties.

Work with your simulator

The whole concept of tearing, sequencing, and converging flowsheets may still be a convoluted idea to some longtime users of process simulation. No matter what simulator you use, if you have a process with at least one recycle, your simulator is hard at work performing these and other functions in the background for you. To learn more about convergence and solving convergence problems, you need to know how and where your simulator reports which streams it is tearing, what convergence tolerance is being used, what the overall block sequence is, what convergence method is employed, and the progress or status of the tear stream solution for each iteration. To find out this information and how to customize each aspect, you need to consult the documentation, talk to experienced users in your organization, or call your simulator vendor's help line. Benefits include:

- Viewing the progress or status of the tear stream solution for each iteration can help you pinpoint exact convergence problems (like component buildup, as explained above).
- A simulator's initial guess for a tear stream is zero flow. Sometimes this is not a good place to start, and you would do better providing an initial guess for the stream. You must know which stream is being torn, however, to supply an initial guess for it. (Not all simulators are "smart" enough yet to choose tear streams based on the recycles for which you have supplied guesses.)
- Even if you know the flow rate or other information about a particular recycle, your simulator can choose some related stream to tear instead. You have the capability, however, to override the assignment and select for yourself which stream is torn, and then input all known stream values as the initial guess. The advantage of this is usually quicker loop convergence.

How to avoid convergence

Figure 8 shows some common unit-operation blocks that have recycle loops to accommodate heating or cooling associated with that unit. These types of constructs often are encountered in simulating flowsheets. On the surface, it seems that we will need several blocks and a convergence loop to simulate the unit operations shown. This is not the case, though.

Consider the flash drum in Figure 8. Mentally draw a mass balance envelope around the entire flashing system. Doesn't the envelope comprise a single-stage flash with duty? The whole construct, recycle and all, can be simulated by a *single* flash block. The same logic can be applied to the reactor with duty, and even the vacuum pump from Figure 5.

Some engineers may have a concern that the recycles will need to be shown on a heat and material balance. Because they are internal to the block, it is true that these recycle streams are not explicitly reported; however, they can be figured easily on-line from calculated block duties. (The correct procedure involves using dummy stream manipulators included in all simulators: duplicate the outlet stream, multiply the duplicate stream by some factor, send it through a dummy heater whose duty is the same as the flash drum, and vary the multiplication factor until the desired temperature difference across the dummy heater is obtained. Although this procedure

-198-

seems complicated, it is computationally simpler and preferable to modeling the flash recycle.)

Tip: A flash with recycle is still a flash overall — use a single flash block to model the overall process.

Some good advice

Here are some recommendations and rough heuristics.

- You probably don't need to worry about any aspect of convergence at all unless your flowsheet has complex blocks (like distillation towers) and more than a couple of significant recycles or design specifications.
- When you are building your flowsheet for simulation, start with a simmodel that converges, then increase the complexity. If convergence fails, you only need to retreat one step to troubleshoot.
- If your simple model is taking a long time to solve or converge, you may want to consider turning off the energy balance aspect of your simulation. Most simulators have an option to let you run a mass-balance-only simulation. You will have a much simpler model that will converge much quicker (at an obvious cost of decreased complexity — no rigorous distillation, and so on).
- Beware of components like polymers with huge molecular weights. Because simulators only converge on mole flow within a set tolerance, the mass balance of a component may not very precise. (For instance, consider a component with a molecular weight of 1,000,000. Because a typical default tolerance is 0.0001, your simulator reports a converged model if *in* = 1 mole/h and *out* = 0.9999 mole/h. This, however, represents a noticeable mass difference of 100 lb/h!)
- If you are the type of person who likes to control aspects of your flowsheet convergence, keep in mind that nested loops can add stability to your simulation. Solve convergence loops simultaneously if it seems logical to group them in that way. For instance, if you are depending on a design specification to control temperature in a recycle loop, it might be easier to converge the tear stream and design specification simultaneously (rather than sequentially nested).

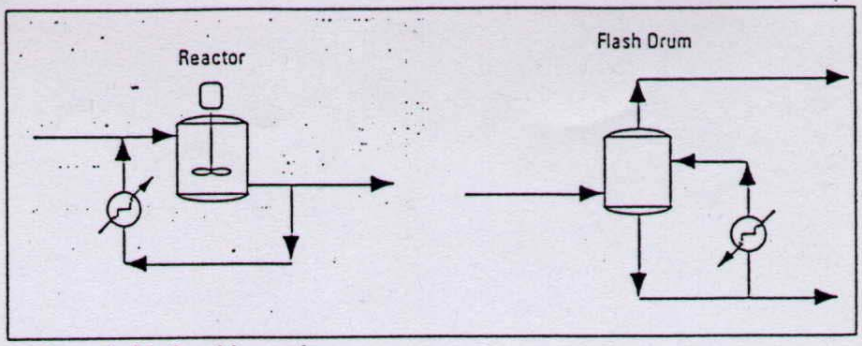


Figure 8. Blocks with recycles.

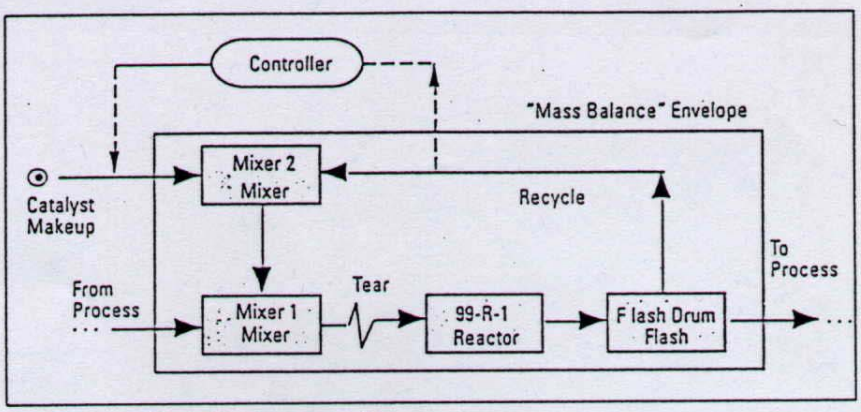


Figure 9. A block flowsheet to avoid.

Don't sabotage your convergence

If you ever have tried to model an existing process, you know how frustrating it is to have actual data regarding output streams (that is, those for which you have no direct control) that don't match results from your simulation. In such a situation, you don't know if the problem is in the data, your physical properties, or your simulation input specifications. If you are modeling a process with a recycle loop where you have a specification on the recycle, one thing that you should *not* do is try to manipulate the input to match your specification while the loop is converging.

For instance, the recycle in Figure 9 is a reactor pumparound loop in which catalyst concentration is critical to the reaction. The figure shows a "controller" (design specification or a user-programmed function) that samples the recycle on each iteration and varies catalyst makeup to set catalyst level in the loop. This convergence problem has a low likelihood of success.

Without resorting to a discussion of numerical methods, the reason this is a bad idea is because the convergence solution, as stated above, is determined in general by the mass balance around the whole loop. Your simulator is guessing values for the tear stream that ultimately balances mass "out" of the loop with mass "in." You essentially are sabotaging the simulator's efforts by changing mass "in" on every iteration — in essence, it must try to hit a moving target. Once again, your loop will only converge if you are lucky and your controller's guess just happens to complement the simulator's guess.

The easiest resolution to this problem is to set the inlet flow, converge the loop, check the result to see if it matches your specification, and then

reset the inlet flow. You essentially have nested the convergence problem within your trials and error. If your recycle concentrations are highly sensitive (for instance, due to sensitive kinetics), an alternative solution is to add a couple of dummy blocks — one to "bleed" the key component away from the loop, the other to add back the correct amount of "makeup" key component. Now you set the process input flow, converge your loop, compare the bleed stream to the makeup stream, and adjust your process input flow until the bleed stream equals the makeup stream.

Tip: Never use a design or programmed specification to set the makeup in a loop based on a recycle stream value *at the same time* that the loop is converging.

Symptoms: Your recycle loop, in which you know what a composition should be, won't converge in the set number of iterations.

How to avoid convergence — revisited

Consider the heat interchanger schematic shown in Figure 10. If you were to use a "heat interchanger" simulation block to model the interchanger, it should be obvious that an iterative solution requiring stream convergence would be necessary.

On the other hand, say you used two simple heater blocks connected by a heat, or duty, stream (as illustrated in Figure 11) to model the interchanger. If you needed to specify stream "cold-out" temperature, you would completely eliminate the need for convergence. The required heat would be conveyed from the hot-side block to the cold-side one where it would be available when the "hot-in" stream had been calculated from the process.

Similarly, if your temperature specification is for stream "hot-out," you will introduce a trial-and-error convergence problem because the correct duty to convey to the cold-side block is unknown at the outset, because the hot-in stream has not yet been calculated. However, our tear stream will be the heat stream

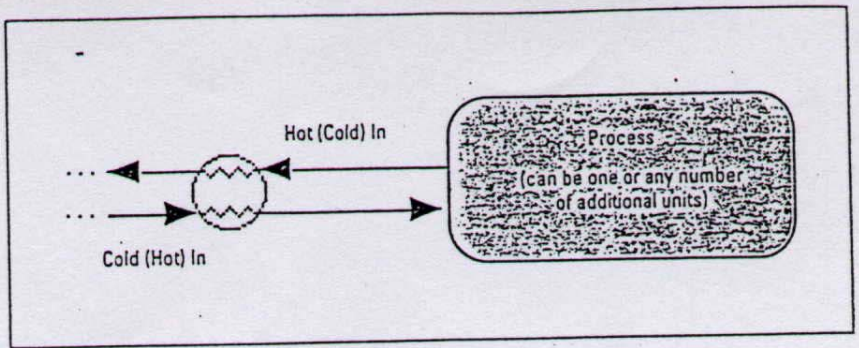


Figure 10. Schematic of a heat interchanger.

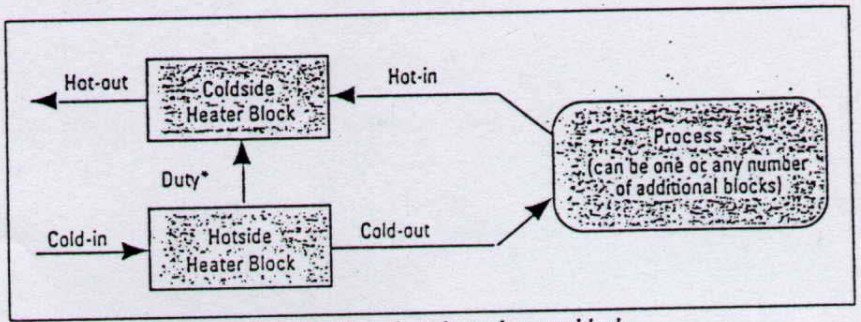


Figure 11. Preferred flowsheet for a heat interchanger block.

"duty," which requires convergence of only one variable — heat duty — instead of component flows, enthalpy, and pressure of a material stream.

Tip: When simulating a heat interchanger, use two simple heater blocks connected by a heat stream. The heat stream always should point from the first block encountered to the later block, even if actual heat flow is not in that direction.

More good advice

Before you get too caught up in convergence strategies, it always is worthwhile to step back and consider a key point.

• How important is it that you converge your recycles, anyway? If your purpose is to get a rough material balance, you might be better off breaking your recycles and converging them by hand (that is, iterating a couple times yourself to get close). If you are concerned with accurate bookkeeping of trace components, however, you had better worry about convergence and tear streams.

Here are some additional recommendations and rough heuristics.

• Sometimes when your simulation doesn't converge, all it needs is a few more iterations. One easy thing to do before you start troubleshooting is restart calculations from the beginning, using the latest results as the starting point. This effectively will double the number of iterations for that stubborn convergence loop.

• If you only learn one thing from this article, let it be this: *never* set the flow of a purge stream out of a recycle loop. Analyze the mass balance envelope around your recycle loop to make sure that you have supplied a reasonable outlet for every ounce of every component that might find its way into the loop. Reactors in the loop can be considered pseudo ways for components to be purged from the loop (reactants) or injected into the loop (products).

• If you cannot get your distillation model to converge for what appears to be a straightforward separation, check your residue-curve map of the system *(3,4,5)*. The presence of a ternary azeotrope or distillation boundary might indicate that the desired separation *cannot* occur.

-200-

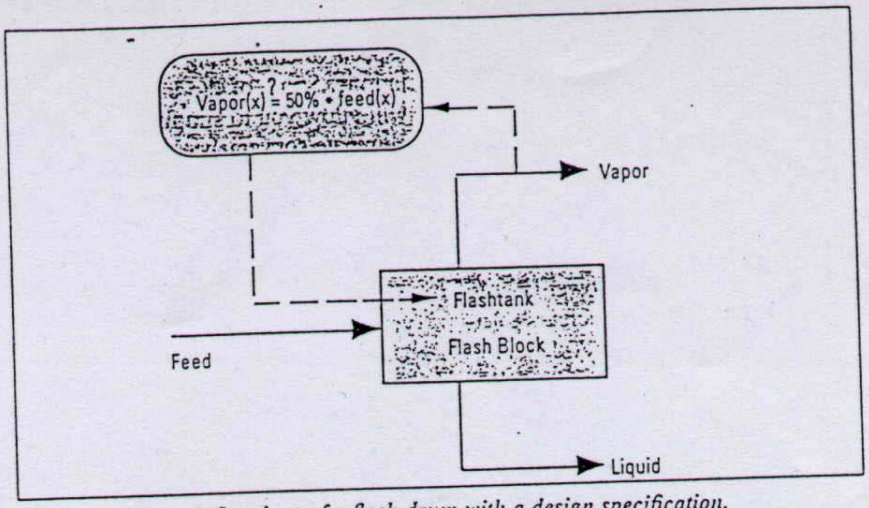
• To save computational time and greatly speed up convergence, a non-rigorous separator block can be used in place of a distillation model when the details of the separation are unknown or unimportant but component splits are known. It is always a good idea, though, to reinstate the rigorous model eventually once recycles have been better defined to insure technical feasibility.

Working with flash drums

Often when simulating a single flash system, you are given an alternate specification — key component ...pth of flash, for instance — instead of typical flash conditions. Because your flash block requires that you enter values for two of either temperature, pressure, vapor fraction, or duty, you will need to iterate on the flash conditions to meet your unique specification.

A programmed design specification to flash 50% of some feed component, graphically shown in Figure 12, should be able to locate the flash conditions to meet the specification. Consider for a moment the best way to program this problem. It might occur to you to set the pressure in the vessel and vary flash temperature or duty until the correct amount of vapor is boiled. Or maybe you should set temperature and vary pressure. Usually, however, the best way to program this is to set either temperature or pressure (as dictated by the process), and vary vapor fraction until the specification is met. (You cannot set just the duty and vary vapor fraction, or vice versa.)

Varying vapor fraction is a better solution than varying temperature (or pressure) because the temperature (or pressure) bounds of a reasonable solution are the bubble point and dew point. Inasmuch as the feed to the flash system likely originates from an upstream process, you will not know the feed composition when programming your design specification, and you will be forced to choose a wide temperature range to insure that you have encompassed the bubble and dew points. When solving the design specification, your simulator's algorithm very likely will choose two temperatures that fall outside of the two-phase region (all liquid or all vapor), at which point it will give up and report some kind of error (like: "function not changing" or similar). Vapor fraction, on the other hand, is a nicely bound function from 0 to 1, and *all* cases fall within the two-phase region.



■ *Figure 12. Block flowsheet of a flash drum with a design specification.*

Tip: When you need to use a design specification to set a flow from a flash tank, always set temperature or pressure (as dictated by the system) and vary vapor fraction.

Symptom: Your flash block design specifications often report errors such as "function not changing," or "solution outside of bounds," or "exited design specification due to error."

Adjustment of tolerances

When balancing a checkbook, some careful bookkeepers are not satisfied unless they account for every cent that leaves or is deposited in the account. Other folks accept that they may have missed a banking-machine transaction or monthly interest fee, and are happy if the account balances to within a couple of dollars.

A similar analogy holds for your process simulator when it solves complex unit-operation block equations or convergence problems. When a guessed process variable (like a bubble point temperature) is checked against the calculated result, the relative difference is called the tolerance. Calculations or tear streams are converged when:

$$-\,tolerance \le [X_{calculated} - X_{guessed}]/X_{guessed} \le tolerance$$

for all associated variables (X in the equation above).

The tolerance for unit-operation and convergence blocks is preset in your simulator to a reasonable value, usually around 0.0001. Most of the time, your simulator's built-in tolerance will serve you well unless, of course, you begin using complex models with extensive convergence problems.

Generally, you will need to consider revising tolerances when you have a very large, complex model with one main recycle loop at the front end; many nested convergence problems; or recycles that are much larger than "in" and "out" flows (like solvent recycles that can be two to three orders of magnitude greater than other process streams). Specifically, you will need to consider revising tolerances if your simulator reports that the convergence error comes very close to the tolerance, but randomly bounces around the criterion and cannot converge after a fair number of iterations.

Intuitively, it makes sense that if you have nested loops, error can be propagated from an inner loop (whose

tear stream values need only to be satisfied to the nearest 0.0001) to outer loops with the same tolerance. Also, consider that complex blocks have iterative equations whose tolerance is probably the same as that of the convergence loops — further contributing to the problem. The solution either is to manually adjust tolerances of the blocks that are located inside nested loops to be lower (that is, tighten the tolerance), or increase tolerances of your outer recycle loops (loosen the tolerance), or both. An alternative solution, which is not always possible, is to combine loops rather than nesting them.

A word of caution, though. Some new users go overboard when they learn how to loosen tolerances. It seems like the answer to all convergence problems — your recycle loops converge quickly in only several iterations. Loose convergence tolerances, however, translate to loose overall mass balances.

If you are simulating a process that makes a very small amount of byproduct on each reactor pass, it may be critical to the success of the entire project that you accurately model the buildup of the trace component in the reactor recycle loop, because, say, it poisons the catalyst or necessitates an additional separation step. A loose convergence tolerance would disguise the buildup and you would report flawed results. Put another way, if you were embezzling money by transferring $2.00 per month from checking accounts, the careful bookkeeper above would notice the first month, while the lenient bookkeeper might *never* catch on.

Tip: Tighten convergence tolerances on unit-operation blocks within a nested loop or loosen convergence tolerances on outer recycle loops, or combine (unnest) loops where possible.

Symptom: Your simulator reports that convergence error approaches, but randomly bounces around the tolerance and cannot converge.

That's not all

The preceding tips and techniques are a small, varied sample of the ways that you can control and manipulate aspects of converging flowsheet calculations. Some of the basic skills that will allow you to simplify or virtually eliminate convergence problems are:

- effectively analyzing the in/out flow structure of mass balance envelopes associated with recycles, to help you avoid poor or illegal specifications;
- properly exploiting the power of your simulator by understanding how to control aspects of convergence;
- correctly modeling and manipulating recycles of known composition; and
- avoiding problems by eliminating the need for convergence in some situations.

These tips are empirical in nature. They complement numerical methods and complex convergence algorithms that can be employed to quickly and efficiently solve iterative problems. (If you are interested in learning more about these mathematical methods, consult Refs. 1, 2, and 6.) You can achieve even greater power over flowsheet simulation and convergence by learning when and how to adjust parameters associated with a particular method and when to switch convergence methods entirely. Whether or not the complex mathematics interest you, it still can be rewarding and challenging for you to develop from your own experience simulation convergence shortcuts, heuristics, and techniques similar to those above.

A final thought for when you have successfully simulated and converged a complex flowsheet, whether or not you had to struggle with convergence: a converged model in no way guarantees a technically realizable process. Some, but not all, feasibility problems will manifest themselves in convergence or other simulation errors. You, the engineer, must use your judgment to insure that your reasoning and results are sound for all aspects of your process. CEP

For a free copy of this article, send in the Reader Inquiry Card in this issue with No. 184 circled.

Literature Cited

1. Westerberg, A. W., H. P. Hutchison, R. L. Motard, and P. Winter, "Process Flowsheeting," Cambridge Univ. Press, Cambridge, U.K. (1979).
2. Reklaitis, G. V., "Introduction to Material and Energy Balances," John Wiley, New York (1983).
3. Westbrook, B. M., and J. R. Knight, "Synthesis and Evaluation of Separation Systems using Simple Distillation Residue Curve Maps," presented at the AIChE National Meeting, New Orleans, LA (Mar. 1992).
4. Stichlmair, J., J. R. Fair, and J. L. Bravo, "Separation of Azeotropic Mixtures via Enhanced Distillation," *Chem. Eng. Progress*, 85 (1), pp. 63–69 (Jan. 1989).
5. Doherty, M. F., and J. D. Perkins, "On the Dynamics of Distillation Processes III. The Topological Structure of Ternary Residue Curve Maps," *Chem. Eng. Sci.*, 34, pp. 1,401–1,414 (1979).
6. "ASPEN PLUS User Guide," Release 8, Aspen Technology, Cambridge, MA (1988).

Acknowledgments

The techniques and heuristics presented in this paper are the result of extensive experience using simulation tools at Eastman Chemical Co. Contributions from Victor Agreda, Brenda Barnicki, Joe Bays, Ben Becker, Richard Bonner, Paul Mahaffey, Jim McGehee, Chris Niederer, Tim Nolen, Lee Partin, Joe Parker, Tom Tidwell, and Tom Yount in process technology groups of the Process Engineering Dept. were invaluable to this effort and are greatly appreciated.

R. C. SCHAD is an advanced chemical engineer in the Process Engineering Dept. of Eastman Chemical Co., Kingsport, TN (615/229–3139; Fax: 615/224–0453; e-mail: rcschad@emn.com). He specializes in process simulation and has been the leader of his department's flowsheet simulation technology team for four years. His assignments at the company have included the full range of process engineering, from flowsheet synthesis and economic feasibility studies to design of new facilities. A registered Professional Engineer in Tennessee and a member of AIChE, he received a BS in chemical engineering from Purdue Univ.

-202-