**The Case for Equation Oriented Solver**
**By Hazem Haddad, Ph.D.**
**Part I. Overview**

## Introduction

This document explains a concept which can be used (at least partially) to solve simulation problems. The concept is Equation Oriented and can be programmed in a manner such that the solver does need to develop a flow sheet solution sequence.

## Simple Case

Consider a case with five heaters/coolers linked in series (See Figure 1), the coolers are named A, B, C, D and E and the connecting streams are 1, 2, 3, 4, 5 and 6. $\Delta P$ is the pressure drop across the heater/cooler.

A Momentum/pressure balance across Heat exchanger A yields
$P1 - P2 = \Delta P_A$. If we apply this balance to the entire process, unit by unit and tabulate the equations, we will arrive at the following Matrix.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | -P2 | | | | | $-\Delta P_A$ | | | | | =0 |
| | P2 | -P3 | | | | | $-\Delta P_B$ | | | | =0 |
| | | P3 | -P4 | | | | | $-\Delta P_C$ | | | =0 |
| | | | P4 | -P5 | | | | | $-\Delta P_D$ | | =0 |
| | | | | P5 | -P6 | | | | | $-\Delta P_E$ | =0 |

As far as pressure is concerned, this system is governed by 5 equations and 11 variables.

If the user specifies the pressure drops for all heaters/coolers, the governing set of equations reduces to 5 equations and 6 unknowns as follows:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | -P2 | | | | | $=\Delta P_A$ | | | | | |
| | P2 | -P3 | | | | $=\Delta P_B$ | | | | | |
| | | P3 | -P4 | | | $=\Delta P_C$ | | | | | |
| | | | P4 | -P5 | | $=\Delta P_D$ | | | | | |
| | | | | P5 | -P6 | $=\Delta P_E$ | | | | | |

The terms on the right of the "=" sign are simply the B vector. There are several observations that can not be ignored here. Note that if the user specifies ANY additional pressure, then the system can be solved. This means that the user can specify the pressure of stream 6 and after the matrix is solved, it would appear that the simulator solved "backwards". It is not backwards, it is simultaneous.

The other key observation here is that the matrix does not require an inverse to be solved, it is a Thomas matrix.

This concept can be applied to the mass balance and energy balance as well.

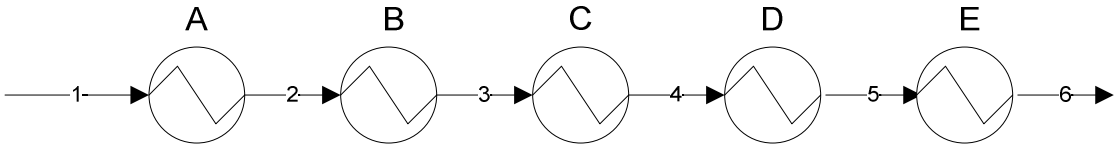


Figure 1.

This concept can be applied to Flow Rate, Enthalpy and Composition.

In the case described above the matrix looked good and solvable only because I did it sequentially. When done programmatically and if the user deleted operations and streams and re-inserted streams a matrix could look like the one shown below and referred to as the "Unarranged Matrix"

```
    X X X
    X X X
X X
            X X X
X X
            X X X
            X X X
          X       X X
          X       X X
          X       X X
```

It is however possible to re-arrange matrices to put them in a form where a human developed or another subroutine can examine it and determine if it is solvable. In this case the above matrix, when rearranged would look like this.

```
X X X
X X X
      X X
      X X
          X X X
          X X X
          X X X
                X X X
                X X X
                X X X
```

Mathematically, it is the same matrix but this one can be examine by another subroutine.

What should be noted here is that this matrix can be divided into four sub-matrices. Three of them are solvable but the first 2 x 3 matrix is underspecified. Note that a 3 x 2 matrix would indicate that the system is over specified and the user should be prompted.

Thus, no matter what the flow sheet looks like, and with the solver making no attempt at developing a "process sequence", it may be possible to solve certain portions of the flow sheet for Pressure, Enthalpy, Flow rate and Composition.

After doing that, the solver can go through all the streams and any stream that is thermodynamically specified can be solved.

**The Modular Approach Finishes the job started by the Equation Oriented Approach**

The streams that are not solved are the ones that are outlets of reactors or outlets of LNG type heat exchangers (MHEATX on Aspen plus) where the user made a specification like MITA=5 or a regular heat exchanger with a UA Specification or something like that.

At this point, the solver can go through the operations and check so see if the operation is solvable and if so, it will call a module specific to that operation and solve it.

In Conclusion, the Equation Oriented approach (or the Matrix Oriented Approach, if you wish) solves as much as it can and the modular approach will finish the job.

**The Preliminary Solving Algorithm**

The sequence of the following four steps does not really matter for a reason which will become apparent in the next few paragraphs

1. Flash all fresh feed streams. This is where the user will most likely enter stream compositions, flow rate, etc.
2. Composition Matrix. Develop the composition matrix and solve as much of it as possible. In the example above this step will simply transfer the overall composition from stream 1 (or whichever stream the user entered the composition for) to the remaining streams.
3. Flow Matrix: Develop the flow matrix and solve as much of it as possible. In the example above, this will set the flow rate of all steam equal to the flow rate of the stream the user chose to specify (most likely stream 1).
4. Pressure Matrix: Develop the pressure matrix and solve as much as possible.
5. Go through all the streams and solve all streams that have two of the following: Temperature, Pressure, vapor fraction and enthalpy (we will talk about the Vapor Fraction and Enthalpy specification later).
6. Enthalpy Matrix: Develop the enthalpy matrix and solve as much as possible. In the example above, this will end up calculating the heat duty for the exchangers.
7. Go through the streams again, and see if any more are solvable.
8. Repeat the procedure starting with step 2 until no more units/streams are being solved. To do that, each of the steps above should count the units/streams it managed to solve and pass this result in a argument or make it available somehow. When the number of units/streams being solved is ZERO, then it is time to call the Modular Solver.

**In Code**
When written in code, the solver may look like this

```
Subroutine General_Solver

Call Stream_Solver(N_Stream_Solved)    // This will flash the fresh feed stream
Repeat
        Repeat
            Call Composition_Solver(N_Comp_Solved)
            Call Flow_Solver (N_Flow_Solved)
            Call Pressure_Solver(N_Pres_Solved)
            Call Enthalpy_Solver(N_Enth_Solved)
            Call Stream_Solver(N_Stream_Solved)
            N_Solved = N_Comp_Solved + N_Flow_Solved + N_Pres_Solved +
                        N_Enth_Solved +N_Stream_Solved
        Until N_Solved = 0

        Call Modular_Solver(N_Modular_Solved)
Until N_Modular_Solved = 0
```

In part two, I will develop a real process and see if we can better study the feasibility of this approach and refine this preliminary procedure.