FluidFlow3 is a steady-state pipe network analysis program, solving for flows and pressures throughout a pipe system. Its solution is essentially a "snap-shot" in time of the system operating under the conditions pertaining at that instant.

Users are, of course, not limited to just a single solution to a model. Keyboard changes enable equipment items such as orifice plates or control valves to be re-defined in order to obtain the desired outcome. "What-if?" calculations and this type of repetitive simulation can be quite straightforward, if somewhat time consuming.

What becomes more involved is the simulation of real-time or extended-time conditions, such as how gravity fed system would perform as the head in the reservoir reduces over time. To simulate this, a sequence of steady-state calculations could be performed based on a time increment. After each flow calculation the time increment is then used to calculate the volume of water discharged and the consequent lowering of the level in the reservoir. A new flowrate is then calculated on the basis of the revised level. With time-increments set small enough to make the calculation sufficiently accurate, the number of calculations (and keyboard actions) would become prohibitive. And even more so if other variables need to be included, such starting a pump to recharge the reservoir once the level reached a set position and then stopping the pump once the reservoir was full – a sequence of events that may take place over several days.

This is where the Scripting module comes in. Scripting allows you to make changes to any flowsheet element (node or pipe) via a Pascal-based scripting language and then study the effects of these changes at any other flowsheet component or components. So, for the above example, a simple loop calculation would suffice. Starting with the initial reservoir level and the first calculation of flow and the time increment, the corresponding fall in level in the reservoir would be determined and the new level set. Subsequent calculations would repeat the process until a defined condition brings the sequence to a halt.

The Pascal-type code has been specifically developed for use with FluidFlow3, but the coding style would be very familiar to anyone with a modest exposure to programming in Basic or Pascal. Typical *FluidFlow3*-specific commands include the *Get.Element* command which identifies a component by its ID ready for some change to its properties, or the *GetPipes* command which identifies all the pipes in the model and places them in an array. An example of how the *GetPipes* command might be used in a script would be to change the roughness of all the pipes in a model to determine the effect of sliming or deterioration.

The Scripting Results window allows the graphing of results – essential for time dependent simulations. So, for instance, the fall in reservoir elevation described above could be plotted. Results can also be exported to other applications such as Excel. Inbuilt code allows for convenient plotting of the hydraulic grade and energy lines.

The applications of the Scripting module are extensive, from the simple - such as sizing an orifice plate to generate a desired pressure drop, to something more complicated - such as simulating the interaction between variable speed pumps and associated control valve positions. Some examples are listed below:

Aging of a cooling water system:

Assume a cooling water system will deposit 0.07 mm of scale each year. Heat transfer considerations dictate that once the flow drops below 95% of its clean initial value the system will need cleaning. Looping through a sequence of calculations with increasing levels of scale will calculate the time interval between cleaning.

Pipe Sizing:

From past operating experience on similar plants we know that operating with pipe velocities between 1.5 and 3.5 m/s has been successful. Therefore we require a script which highlights all pipes whose velocity falls outside this range.

Valve Closure and Pressure Loss

Calculate the pressure drop across a valve with constant flowrate as the valve closes in 2.5% increments and chart the results.

Troubleshooting of Existing Plant:

The condition of existing plants is often uncertain. Pipes may be slimed or roughened compared to new conditions; orifice plates may be eroded; valves may be in positions different to those shown by the tell-tales and pump impellers may be worn. Physical investigation such as dismantling pumps is expensive, time consuming and may require shutdown. However, if is often possible to obtain reliable measured flows and pressures at some locations around the plant. Scripting allows the *physical* investigation to be replaced by a *virtual* investigation using measured flows and pressures as a guide.

A simple script could run the model with different impeller diameters in order to simulate impeller wear. Model results would then be compared to known data to see if there was any relation. The same approach could be applied to pipe roughening. The objective would be to "calibrate" the model to agree with known site conditions and then use the conditions in the model as a guide to the cause of those conditions.

Hazard and Operability Safety Studies:

Determine whether a given/unsafe pressure is exceeded if a pump speed is increased to maximum, or a valve is closed etc. Failure conditions could be simulated such as control valves failing closed.