Application of LabVIEW for dynamic simulation of Hydraulic piping systems

Dr.Ing Bjørnar Svingen
SINTEF Energy Research
N-7465 Trondheim, Norway
bjornar.svingen@sintef.no

Abstract

Transient simulations of hydraulic piping systems consists of simultaneous solution of the 1D wave equations that relates flow to pressure in a pipe, and solutions of all other non-pipe components constituting the total system. The method of characteristics (MOC) is the most widely used method due to its analytic accuracy and speed. This paper presents an implementation of MOC as a so-called toolkit or library in LabVIEW. Each type of physical component (pipe, valve etc) is programmed as a component sub-vi (component sub *virtual instrument*). A component sub-vi can best be described as a lightweight variation of an *object* in an object oriented programming language, because data and procedures are encapsulated within each component. Hydraulic systems, with virtually limitless complexity, are then constructed intuitively using point and click methods in the standard LabVIEW interface, and later run, without needing any prior knowledge of LabVIEW. The modularity means that new components and upgraded components are easy to incorporate, while the LabVIEW interface provides all interactive graphical representation, both in batch mode and in real-time simulations. This toolkit is implemented in a package called LVTrans (LabVIEW Transient Simulations). This package is actively used and under continuous development at SINTEF Energy Research, and is also licensed to Statkraft ASA.

Introduction

LabVIEW[1] is not normally seen as a general purpose simulation environment, although toolkits and writings exist on the subject [i]. LabVIEW is best known as a programming environment for data acquisition and analysis, and as such has achieved widespread use throughout the scientific and industrial communities. Newly gained experience with LabVIEW as a simulation environment for dynamic analysis of piping systems, shows that it will make an intuitive and practical platform, both for the hydraulic analyst and the programmer. By using component sub-vis[2] based on LCOD (LabVIEW Component Oriented Design) as described in [ii] and utilizing the graphical programming environment and language known as G, piping systems with literally limitless complexity can be analyzed efficiently. This analysis can be done completely interactively or as a more batch like process. It can be done in real-time or faster/slower and with any degree of detail. In addition, stand-alone compiled simulators can be made, and real-time measurements can easily be added. All this can be done with very little programming by using existing libraries and procedures in connection with the component sub-vis.

The origin of the simulation program (LVTrans) was to be able to analyze relatively simple systems using the Method Of Characteristic (MOC) without resorting to costly commercial software. The basics of a MOC analysis for a limited sized piping system (2-3 pipe reaches) are rather simple technically [iii] [iv], and a solution can be made very rapid even in Excel. Due to the author's previous experience with LabVIEW in laboratory environments, this platform was chosen for this simple, rather ad hoc analysis. From there on the program has steadily evolved. The introduction of component sub-vis enables LabVIEW to handle complex and large data structures in much the same manner as objects handles large data structures in a text based object oriented language [ii]. Component sub-vis also makes it simple and tidy to build large systems. A component sub vi is a vi (sub routine) that has both data and functions, and these data and functions are encapsulated

[1] LabVIEW is a registered trademark by National Instruments, USA.

[2] A component sub-vi is defined here as a sub-vi based on the principles of LCOD (Labview Component Oriented Design) as described in [ii]

within the vi. The main difference from ordinary objects is the implicit graphical approach with seamless interaction with existing LabVIEW libraries and user interface, and the fact that component sub-vis are not "true" objects, but must be considered more of a "lightweight" object due to lack of some object functionality[3]. The use of component sub-vis is generally not needed when the task is to make a simple to moderately sized data acquisition system. When the complexity of the LabVIEW program and the data structures increases, component sub-vis and/or true object-orientation is considered almost essential. Most of the ready made subroutines and libraries that exist in LabVIEW by default are made with LCOD principles or is object oriented.

The Method of Characteristics (MOC)

MOC is the most widely used method for analysis of transients in piping systems. Most, if not all, text books on the subject include MOC as the most essential tool in analyzing transients in systems. The reason for this is that MOC has an analytical[4] accuracy regardless of grid size and it is extremely efficient in terms of computing power [iii][iv].

Deriving the basic equations that govern the transients in liquid filled piping systems can be done in a number of ways. The most elaborate way is to start with the full 3D Navier-Stokes equations in cylindrical coordinates. By doing the derivation one will also show how and when, several simplifications and assumptions are made [v]. This derivation is not shown here. Only the assumptions and simplifications that lead to the 1D wave equations are listed.

Due to the cylindrical and symmetric layout of a pipe, the 3D equations are reduced to 2D equations. All variation in tangential direction is zero due to symmetry.

A low compressibility is assumed. This is true for all liquid filled systems. The variation of density with respect to pressure then becomes a constant value:

$$\frac{\partial \rho}{\partial p} = \frac{\rho}{K} = \frac{1}{a^2}$$

The next assumption is to assume very low Mach numbers ($M^2 << 1$). This holds for all liquid filled systems, except extremely flexible tubes such as for instance blood vessels. With this assumption the convective terms can be disregarded.

One dimensionality is assumed by averaging the pressure and the velocity in the cross section of a pipe:

$$V = \frac{1}{\pi R^2}\int_0^R 2\pi R v_x dr$$

$$P = \frac{1}{\pi R^2}\int_0^R 2\pi R p dr$$

Then the so called "long wavelength approximation" is applied. This is true when the characteristic length, L, to diameter ratio, D, is large. This assumption is correct when the frequencies of interest are lower than 63% of the first radial fluid mode [v]. For normal piping systems the first radial fluid mode is several orders of magnitude higher than the modes of interest.

Then body forces are excluded due to low values compared with other forces[5]. The remaining equations are 1 dimensional momentum and continuity, the so called water hammer equations [vi]:

$$g\frac{\partial H}{\partial x} + \frac{\partial V}{\partial t} + \frac{f}{2D}V|V| = 0$$

$$\frac{\partial H}{\partial t} + \frac{a^2}{g}\frac{\partial V}{\partial x} = 0$$

These two equations are essentially an ordinary 1D wave equation with a non linear dissipating term. Integrating along the characteristic lines of the equations and substituting the velocity with the flow produces the well known characteristic solution of the water hammer equations that are valid on the characteristic grid (see Figure 1). In the characteristic grid each pipe is grided in the x-t plane [vii]:

$$C^+ : H_i = C_P - B_P Q_i + dyn$$

$$C^- : H_i = C_M + B_M Q_i + dyn$$

[3] True object oriented add-on tool for LabVIEW do exist both commercially and as freeware, (can be downloaded from www.ni.com and www.openg.org), but when doing MOC analysis in piping system they do not add any advantage over the simpler and more efficient component sub-vis.

[4] When disregarding secondary effects of friction.

[5] Exclusion of body forces and assumption of low Mach number is not necessary, but convenient and correct especially for liquid systems.

$$C_P = H_{i-1} + BQ_{i-1}, \quad B_P = B + R|Q_{i-1}|$$
$$C_M = H_{i+1} - BQ_{i-1}, \quad B_M = B + R|Q_{i+1}|$$

$$H_i = \frac{C_p B_M + C_M B_P}{B_P + B_M}, \quad Q_i = \frac{C_p - C_M}{B_P + B_M}$$

Where $B = \frac{a}{gA}$ and $R = \frac{f\Delta x}{2gDA^2}$. The value of point P is calculated by using the values of point A and B. This means that when knowing the values at the previous time step, all the values at the next time step can be calculated when the boundaries are known. In addition a dynamic damping term as described in [v] is included in the calculations where:

$$dyn = \lambda\left(Q_{i-1} - Q_{i+1}\right)$$

And $\lambda = \frac{\lambda_f}{2\Delta x \rho g A}$ where λ_f typically is in the range of $5.0*10^4$ to $5.0*10^6$.

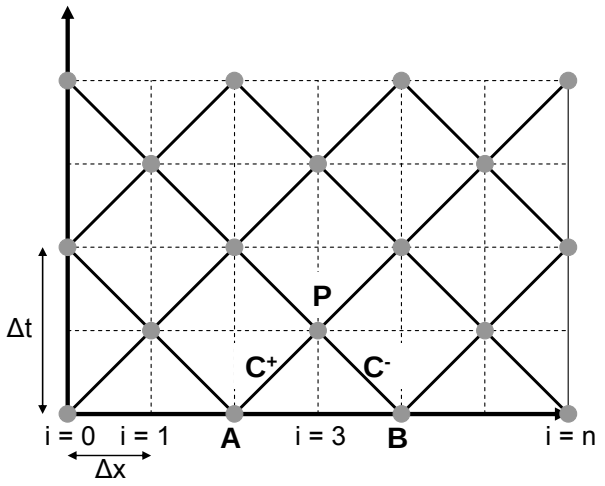

Figure 1 The characteristic grid, staggered type.

Boundaries and formalism

Each pipe consists normally of internal points as well as boundary points[6]. To be able to calculate the next points one therefore needs the values of the previous points and the values at the boundary. By using a staggered[7] grid, some intermediate points are also calculated. The values at the boundaries are normally non-pipe elements such as valves, pumps, turbines and so on.

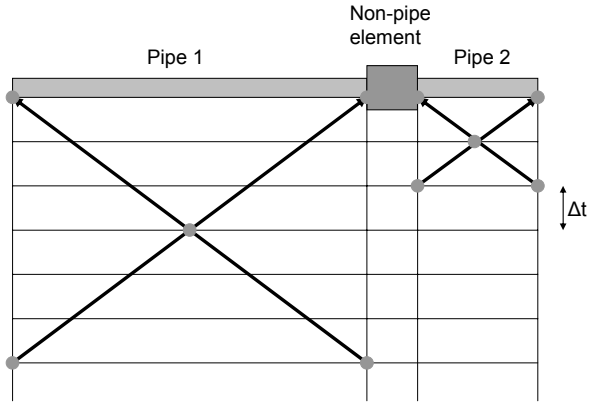

Figure 2 The characteristic grid for two different length pipes connected with a non-pipe element using Algebraic, staggered MOC.

This time stepping procedure can be formalized in terms of time steps and inter-component data transfer. This formalism can be done in several slightly different ways; the one shown here is based on how MOC is formalistically working when the elements are objects.

Initialization

The initialization consists of obtaining parameters for each element (diameters, lengths, fluid properties and so on), that is unique for each element. Inter-element data transfer is generally not necessary during initialization, but values can be passed if this is needed.

Iteration

When the initialization is done, the iteration can start[8]. During the iteration, the following is done:

1. Each pipe element calculates its internal flow variables based on its previous values

[6] In so called algebraic MOC, only the boundary values need to be known since the discretization extends backward in time instead of in space (see Figure 2). This method is also implemented in LVTrans, but is not discussed in this paper.

[7] A staggered grid is mathematically more concise than a regular grid [iii].

[8] A steady state calculation can also be incorporated before the time stepping procedure, but this has not been implemented yet in LVTrans. The simplest (and probably fastest) way of implementing this is to choose a very large time step and iterate until steady state is reached, and then continue with small time step.

and the previous flow variables from the non-pipe element.

2. Each pipe transfer it's newly calculated characteristic values, C_P, C_M, B_P and B_M to its adjacent non-pipe element.
3. Each non-pipe element use their newly received characteristic values together with any additional variables (for instance interactively varied RPM and so on), and calculates all their internal variables together with H and Q at the boundaries. H and Q are then sent back to adjacent pipe elements via feedback nodes.

In LVTrans this can be shown graphically as displayed in Figure 3 where an excerpt from a simple hydropower plant simulation built in LVTrans is shown.

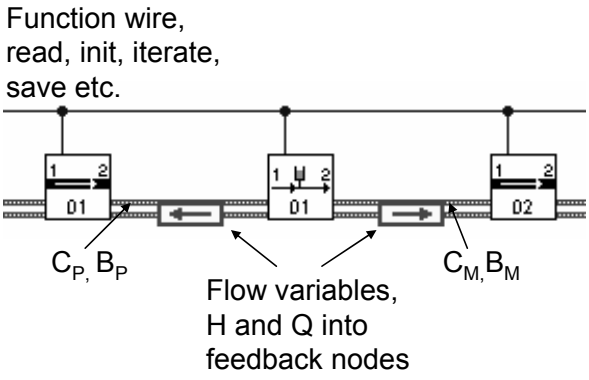

Figure 3 Data transfer during initialization and iteration for two pipes on each side of a surge shaft.

Formalism of MOC with objects

As mention earlier, the MOC procedure is a relatively simple iteration procedure. When building each element as self-contained and encapsulated objects, the simplicity of MOC is also extended to systems with virtually endless complexity. This is so because the inter-element transfer of data is programmed into each element in a consistent manner so that the complexity of each element is hidden. Only the simple formalism of MOC is shown to the hydraulic analyst.

In LVTrans this formalism is graphical and consists of only 2 connections[9] that have to be connected with wires.

1. The characteristics from each pipe. A wire with C_P, B_P, C_M and B_M going from pipe elements to non-pipe elements
2. A bundled wire containing the boundary values for the flow variables, H and Q, going from non-pipe elements to pipe elements via feedback nodes.

When studying the mathematical formalism of MOC, one will find that this procedure (see Figure 3) is a graphical representation of the basic mathematical/technical formalism of MOC. Figure 3 therefore can be said to represent a physical consistent topology as well as a mathematical consistent formalism of MOC in piping systems.

In LVTrans both the two inter-element data transfer interfaces is strongly typed, so that cross-connection is impossible. The feedback node is also automatically made in LabVIEW. This means that when building systems, one only has to concern oneself with the topology and the physical variables. The wiring will automatically be checked by LabVIEW.

Topology

When building large systems, the topology normally must be contained in some framework of data structures [viii]. When using a *true* graphical programming language together with self contained and encapsulated objects, any topological consideration is completely unnecessary. The reason for this is that the topology of the system is an implicit feature of the graphical programming itself. The numbering, or naming, of elements is also essentially unnecessary. However, if one wants added functionality by programmatically calling the component sub-vis at other locations, some naming or numbering of the elements is needed.

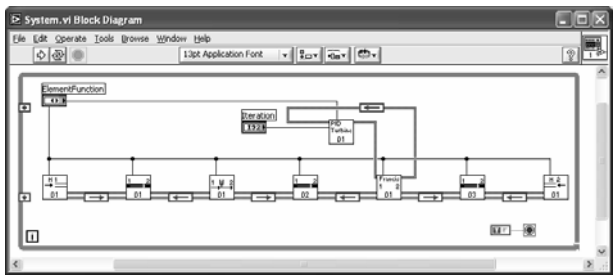

Figure 4 The graphical programming and implicit topology of a simplified hydro power station.

[9] If any eventual values need to be passed between elements during initialization, this also has to be included. In the general case this is not needed.

The elements

When using component sub-vis, each unique physical element may represent one unique copy on disk of a subroutine, or one can use function pointer techniques to make virtual copies in memory instead. Basically this means that when using one element version, naming or numbering must be done programmatically and when using the other element version, naming or numbering is done explicitly. In LVTrans both versions are available at the moment, but only the uniquely/explicitly copied version is used. The explicitly named version is made by making several copies of each element type, and calling them for instance pipe01.vi, pipe02.vi and so on. This copying is done automatically with a separate routine. The two versions are essentially identical otherwise. The reason for using the explicitly named version is that it is much more practical when using LVTrans only as a design tool, which again is caused by some subtleties of LabVIEW[10].

A LabVIEW routine consists of a Front Panel (FP) and a Block Diagram (BD) [i]. A typical analogy is that the FP is the knobs and dials of a radio, while the BD is the inner workings, the transistors, resistors etc or the code. It also has an iconic form when using the routines as subroutines, as shown in Figure 3 and Figure 4. An example of this is the front panel and block diagram of the surge shaft in those figures are shown in Figure 5 and Figure 6.

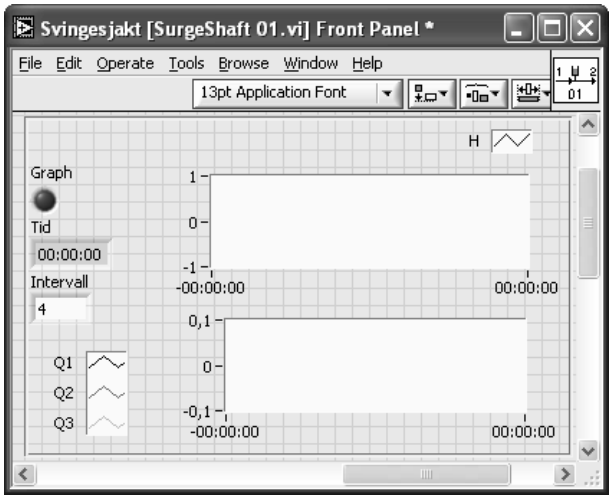

Figure 5 The front panel of a surge shaft element

[10] LabVIEW can make several virtual instances of the same Block Diagram (the actual code and data), but can only show one instance of the Front Panel. Thus, to show the Front Panel of several pipes, they have to have different names and therefore they need to be unique copies on disk.

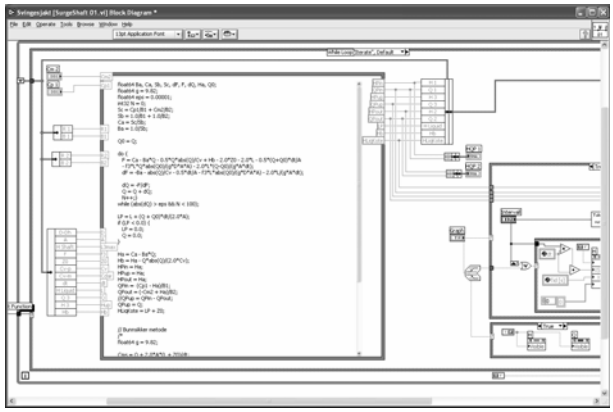

Figure 6 The block diagram of a surge shaft element

All elements in LVTrans are made as components according to the LCOD principles. In practical terms this means that all elements store their own data and their functions internally[11]. The external or public functions accessible from the main program are only "init", "iterate", "read", and "save" at the moment. In addition the front panel can be opened during running to show the graphs, and for interactive operation. In comparison, a "normal" LabVIEW subroutine or *vi* is like an ordinary subroutine that acts on the input data to produce output data.

In LVTrans the liquid filled pipes and tunnel use MOC. This can be either straight forward MOC as shown in Figure 1 or the faster but slightly less accurate algebraic MOC as shown in Figure 2. The non-pipe elements are calculated by algebraic relations as is done for valves. For the more complicated elements such as pumps and turbines, the discretization in time is done implicit, and solved by Newton-Raphson iteration [vii], and sometimes together with interpolation procedures of some characteristic variable (measured or calculated).

For hydro turbines, a very special and accurate dynamic model based on geometry and fundamental turbine equations are used. This model implicitly calculates all internal variables (flow, head, torque and rotational speed) at any instance in time fully dynamically without resorting to interpolations of charts that only are valid for steady state conditions [ix]. If the exact turbine is unknown, as it usually is for new systems, this model is combined with a turbine design routine based on available head and flow [x]. LVTrans will then design a modern type

[11] Data is stored in un-initialized shift registers within a while loop, in the same manner as so-called *functional globals* or *LV2 style globals* [i]

turbine that will be accurate to at least within a few percent, which is more than enough accuracy for the first transient analysis of a new system.

General purpose PID type controllers can be used or more specific turbine controllers from certain deliverers.

Building and simulating a system

LVTrans can be used both as a general purpose design/simulation tool for dynamic and steady state calculation of hydraulic system and as stand alone pre-compiled simulators and logging systems. The building of the hydraulic system is identical for these versions and the same components are used. The only difference is that to build for instance a stand alone simulator, some additional programming is needed for display and other eventual extra functionality.

General purpose simulation

To build a system, a "new"[12] procedure is started. This procedure sets up the basic folders and creates two main files: "Sims2005.vi" (for this example) which is the main program for starting, stopping etc. and "System.vi" which is the subroutine where the entire system is made.

Figure 7 Making a new system

Figure 8 Entering the name of the new system

[12] "Nytt Kraftverk" means "New Power Plant" in English. These routines where specially made for Statkraft ASA, but are general and could be called "New Hydraulic System", since they are not constricted only to power plant.

The system building consists of first opening the BD of the "System.vi" and add elements by clicking and dragging from the appropriate library (folder of component sub-vis), see Figure 9.

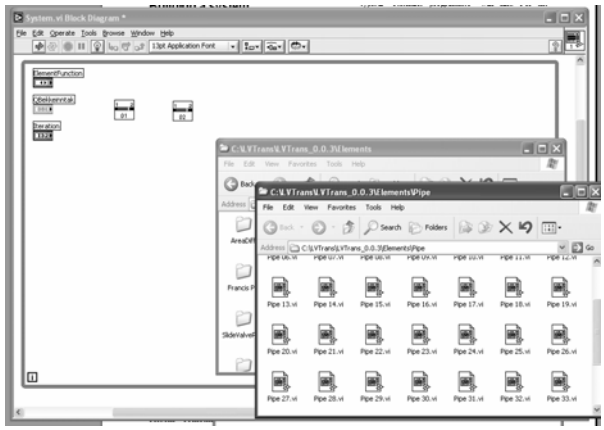

Figure 9 Building a system by "click and drag"

When all appropriate elements are added, the BD may look something like Figure 10.

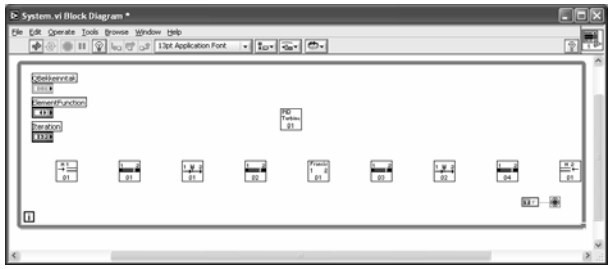

Figure 10 All components added

The next step is to connect the components with wires.

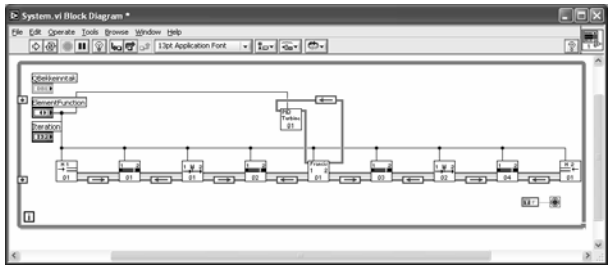

Figure 11 Finished system

The finished system may then look something like Figure 11. When running the system once, text files for input of element parameters are made in a "data" sub folder. These text files are descriptive and the user only has to change values from their default. This may or may not be changed to more interactive procedure in the future, but so far these text files work well.

Figure 12 Main window for starting, setting time steps etc.

Running the simulation consists of pushing the start button in Figure 12, and double clicking on the wanted component sub-vi in Figure 11, which opens the Front Panel of that component. With the front panel open one can draw graphs, save data, input different parameters and so on. Each front panel for all elements is fully interactive.

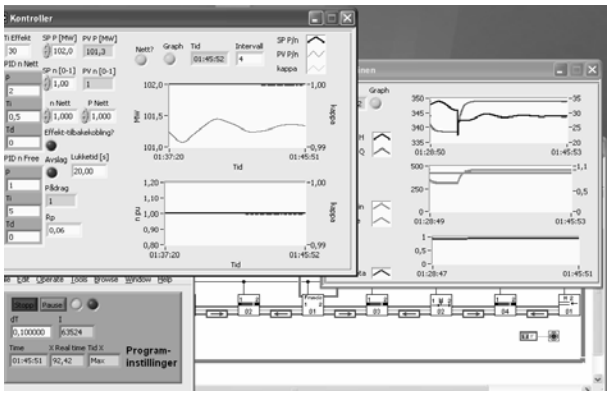

Figure 13 Running a general purpose simulation.

A typical running session may look like Figure 13. Since the Block Diagram is open, one can stop the simulation, do any changes to the topology and/or the input files, and start again.

For the user of the program, the hydraulic analysts, LVTrans has the functionality needed to do a thorough analysis of any liquid filled piping system, and to do that analysis efficiently and accurately. Since the source code is available[13], it is also possible to do changes and add new element types.

[13] The source code can also be password protected to prevent unwanted changes.

Stand alone simulators

Stand alone compiled simulators, and logging systems can typically be built from the general purpose simulation. For a simulator, changes to the topology are seldom needed, so the functionality of the Block Diagram (Figure 11) is transferred to the main window Front Panel together with any other functionality one may add.

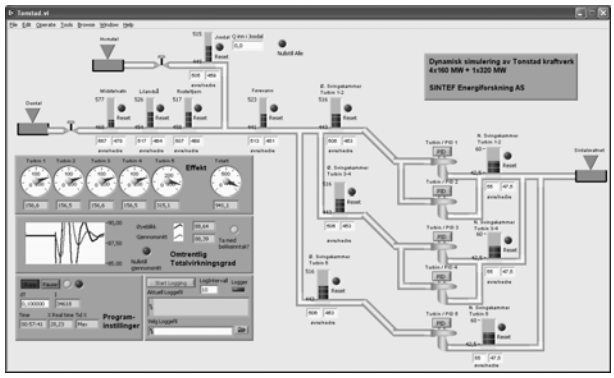

Figure 14 Simulator for Tonstad hydro power plant

A simulator that was made for Sira-Kvina, Tonstad power plant is shown in Figure 14. This plant has four 160 MW turbines and one 320 MW turbine. The front panel is fully interactive, and clicking on either of the tunnels etc will open the front panel of the appropriate component sub vi. The text files for the parameters are still available, so that the system can be fine-tuned to fit with measurements.

Summary

There are several benefits of using LabVIEW as a foundation for simulation of dynamics in piping systems. The most obvious is that since the graphical programming language G is used together with component sub-vis, the building of a physical system can be made, and later run, directly within the graphical user interface of LabVIEW. This is done while maintaining a consistent system topology in relation to the physical topology and at the same time representing the mathematics of MOC graphically. The only "real" programming needed is therefore to program the numerical solvers for the components. Another main benefit is that LabVIEW is specially made for displaying data, and to be used interactively, so that developing user interfaces for each sub vi and the main program is extremely efficient, and the result is very professional looking. Since LabVIEW is in widespread use, one can be certain that the software is constantly maintained and updated.

The MOC procedure for liquid filled pipes is relatively simple. The more general MOC procedure with convective terms and high compressibility, which also is valid for gas filled systems, is a bit more complicated. The overall procedure however, as shown in Figure 3, is identical in structure because the added complexity and difference exist only within each element. An extension to rapid and slow transients in gas pipes should therefore be rather straight forward.

MOC is a non-linear time domain procedure, and the non-pipe elements are usually very non-linear. Such a simulation will give very detailed and accurate results. However, for analysis of large systems, a frequency domain procedure is often just as important, since it gives a better overall picture of the dynamics. In the frequency domain the water hammer equations (1D wave equations) becomes hyperbolic tangent functions, $\tanh(xs/L)$, where s is the complex number $j\omega$ [vii]. Systems can be organized using the structure matrix method as described in [xi]. By using component sub-vis, hydraulic systems can be made with nearly identical layout as in Figure 11. The only difference (for the analysts) will be different wires. The overall solution-procedure will however be very different, and will basically consist of each sub-vi calculating its local transfer matrix for each frequency and inserting the result into a global matrix for solving.

It is the author's personal experience that the main advantage with LabVIEW, namely the graphical programming paradigm, also in some circumstances is its greatest disadvantage. The reason for this is that a graphical programming paradigm is not in general a particularly practical approach for detailed and complex numerical calculations. In nearly all the component sub-vis, except for the pipes, the core numeric is programmed using so called "Formula Nodes" [i], which is basically a C like code inserted into a frame, see for instance Figure 6. This Formula Node can be replaced with a compiled DLL written in for instance C[14]. Originally this was done for several of the component sub-vis, because the compiled C DLL is approximately twice the speed of a formula node (author's experience). However, due to the massive computing power of modern PC's, this speed increase is only of academic interest when calculating liquid filled piping systems with MOC, even for the largest of hydraulic systems, and also because displaying of several graphs draw much more computing power than the numerical computations. LVTrans is today entirely programmed in G, and the total executing speed is anticipated to be somewhere in the range of 50-75% of optimized C code. Still, when the numerical complexity exceeds a certain amount, the Formula Nodes will also loose it's practicality due to a limitation of not being able to write functions *within* a Formula Node [iv]. For more complex numeric, the only practical solution is therefore to use DLLs written in some text based language for the core numerical operations, and link these DLLs into the component sub-vis. LVTrans for liquid filled piping systems is today close to the limit where DLL's would be a more practical solution.

For the hydraulic analyst who builds and analyzes piping systems, the choice of Formula Node or DLL for the core numerical operations is an irrelevant consideration, since the internals of each sub-vi is completely hidden and encapsulated. Thus the main program including all the various component sub vis will look and operate exactly the same regardless of the choice of formula node or DLL. This choice is therefore only a practical consideration for the programmer.

[14] LabVIEW has tools and procedures for including external codes, typically dlls, directly and seamlessly into the vis [i].

ACKNOWLEDGMENT

The development of the program LVTrans has partly been founded by Statkraft ASA and EBL together with Hymatek AS.

The excellent LabVIEW packages and tools at the open source community OpenG, www.openg.org has been extremely helpful.

NOMENCLATURE

a	[m/s]	Velocity of sound
A	[m^2]	Cross sectional area
C^+	[-]	Characteristic equation
C^-	[-]	Characteristic equation
D	[m]	Diameter
f	[-]	Friction factor
g	[m/s^2]	Gravity constant
H	[m]	Pressure head
K	[s^2/m^2]	Bulk modulus
M	[-]	Mach number
P, p	[Pa]	Pressure
Q	[m^3/s]	Volumetric flow
R, r	[m]	Radius
V, v	[m/s]	Velocity
Δt	[s]	Grid size in time
Δx	[m]	Grid size in space
λ	[-]	Dynamic friction
ρ	[kg/m^3]	Density

REFERENCES

[i] National Instruments web site and manuals: www.ni.com

[ii] Jon Conway, Steve Watts, A software engineering approach to LabVIEW, National Instruments virtual instrumentation series, Prentice Hall PTR 2003, ISBN: 0-13-009365-3

[iii] Wylie E B, Streeter V L, Fluid Transients, FEB PRESS, USA 1984, ISBN 0-9610144-0-7

[iv] Betamio de Almeida A, Koelle E, Fluid Transients in Pipe Network, Computational Mechanics Publications 1992

[v] Svingen B, Fluid Structure Interaction in Piping Systems, Dr.Ing Thesis NTNU 1996, ISBN 82-7119-981-1

[vi] Thorley, Fluid Transients in Pipeline Systems, London 2004.

[vii] Wylie E B, Streeter V L, Fluid Transients in Systems. Prentice Hall 1993.

[viii] Bjørke Ø, Maskinteknisk Systemteori, NTH Trondheim 1990

[ix] Nielsen T K, Transient Characteristics of High Head Francis Turbines, Dr.Ing Thesis NTNU Trondheim 1991

[x] Brekke H, Hydraulic Turbines, NTNU 2000

[xi] Brekke H, A Stability study on hydro power plant governing including the influence from a quasi nonlinear damping of oscillatory flow and from the turbine characteristics, Dr.Techn Thesis, NTNU Trondheim 1984.