MNRSIM: An Interactive Visual Model which Links Thermal Hydraulics, Neutron Production and other Phenomena

D. Gilbert, Wm. J. Garland, T. Ha

McMaster University, Hamilton, Ontario, CANADA

gilbert@cas.mcmaster.ca, garland@mcmaster.ca, hats@mcmaster.ca

The goal for the McMaster Nuclear Reactor Simulator (MNRSIM) is a first order visual approximation of the major elements of the reactor including flux calculations, reactor control, thermal hydraulic calculations and eventually fuel management all within a graphical windows environment. The main purpose in the development of this tool is not to provide a high accuracy competitor to the existing simulation tools, but is rather to provide the staff and researchers at the reactor with a tool for understanding the reactor as an integrated system of simulations. The tool follows an extensible modular program design.

1 Introduction

Light weight simulators which are used as teaching or training tools are not a new idea in the nuclear industry. There are a variety of commercially available simulators see for example [1, 2]. The reactor simulation research group at McMaster needed a model which on the one hand could be used to study the McMaster reactor as a system for teaching and training, and on the other hand could also be used as the basis for the existing research program. An on-going problem with software developed by successive generations of graduate students was that the software was usually designed to answer a single specific modeling question, and was rarely reused.

To facilitate reuse of modeling software it was decided that a modeling core, or base should be designed. The modeling base represents the fundamental simulation elements of the MNR and allows for flexible extension. This article will describe the design of MNRSIM's program core, both in terms of the physical models, and in terms of the program design. MNRSIM is freely available for download, both the compiled binary version along with the C source code can be found on the primary web site [3].

The current version of MNRSIM simulates two major physical processes, the reactor core, and the reactor cooling system using six basic nuclear physics modules: a cross section collapsing module, a two dimensional core flux module, a hydraulic pressure module, a cooling system heat transfer module, a core heat transfer module, and a simple control and instrumentation module (See Fig 1).

This paper will first present an overview of the modular software structure, and will then present a short discussion of each module in turn. Finally the preliminary calibration and validation of the model using both experimental data and data from more accurate simulations will be discussed.

2 Software Design

MNRSIM is written in C and compiles under LabWindows, which is a compiler supplied by National Instruments. This compiler was chosen for several reasons, firstly LabWindows provides a functional set of Microsoft-based windows controls including buttons, sliders, and dialogue boxes. Secondly the LabWindows C compiler is compatible with the same set of data acquisition hardware that LabView is capable of using. Although Linux and Unix based implementation offer an attractive set of advantages in terms of stability, network access and free development tools, since software reuse was a primary concern, and since the majority of incoming students only have experience with Microsoft operating systems a Unix based design was not considered. For the similar reasons C rather than C++ was chosen as the implementation language.

One of the principle problems in choosing C as a programming language is that C does not offer some of the more advanced programming structures that can be found in languages like C++, Java or Visual Pascal (Delphi) might have like classes, and object structures. Java, Pascal or Visual Basic might have been better language choices for accommodating students with only a few courses in programming since these languages do not depend on the use of pointers or explicit memory allocation and deallocation. Both Java and Pascal also have well developed user interface design platforms. C is however well known for its high performance characteristics and in the last 20 years has become an almost defacto standard choice for teaching introduction to computing.

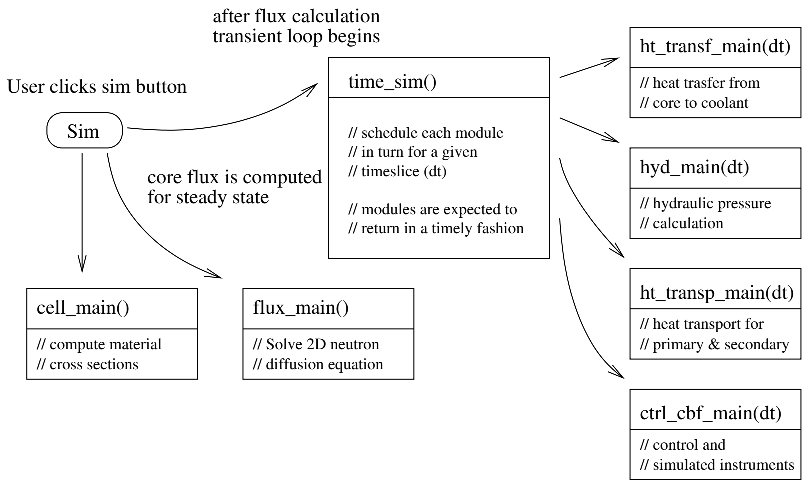


Figure 1: modules and concurrency

To address the need for structure within the program's design, the concept of a module was used as a basic building block. Modules are defined for the purposes of MNRSIM as a C file which contains a collection of related functions which all use similar names. For example, the hydraulic pressure module is contained in a file named hyd.c and has functions with names like hyd_main(), hyd_friction(), hyd_pipe_junct(), hyd_solve() and so on. This gives the flavor of a C++ class with a single object instance without actually requiring programmers to learn C++ syntax.

The program itself is over 12,000 lines of code, or roughly 250 pages, consisting of 20 modules, 9 of which are dedicated to the user interface, 7 handle physics related issues, and 4 are for general program maintenance. Over half of the development time was spent on the user interface.

2.1 Concurrency

Since there are several physical processes which are modeled in what should appear to be a simultaneous fashion some model for concurrency had to be chosen. The guiding principle of choice again was software reusability and ease of understanding. LabWindows provides a threads-based package, and this was experimented with early on in the design process. Threads are normally defined as light weight processes, and in LabWindows are implemented with special primitives which create an independent execution context from a function. While this seemed to be an elegant model of concurrency, it was decided that the threads package would be inappropriate since threads would probably be unfamiliar to most of the project programmers.

A simpler alternative was chosen based on a time management module. When the simulation is initiated by the user the time module runs each physics module in sequence and provides each with a time slice to simulate (see Fig. 1). Each physics module is expected to return control to the time module after completing. This is sometimes called the good citizen approach to concurrency since it trusts that each module will consume only a small portion of the CPU, and that each module will always return to the scheduling function *time_sim()*.

2.2 Shared Variables

Since several modules must access the same set of simulation variables, in some cases concurrently, structures for sharing information across function calls were needed. Global variables were rejected as a data sharing principle since they can dramatically complicate the development of a large project and using them is considered to be bad programming practice. Instead a convention called shared variables was adopted. When a function needs access to a specific set of variables, the flux shape for example, it makes a call to a function named *share_access()* which returns a pointer to the flux array. When the function is finished with the data structure it releases it by calling *share_release()*. All major variables are stored in a global variable store and are accessed through these special function calls which lock the variables effectively ensuring that their usage is exclusive. An active variable function log is kept to sort out conflicts when they do arise.

2.3 User Interface Design

The physical systems that are modeled by MNRSIM are represented diagrammatically by a schematic figure which appears on the upper right of the main MNRSIM window (see Figure 2). The user may select a variety of views, one of either the control panel mockup, the core display, the primary or secondary thermal hydraulic loops. Several other views have been suggested for the simulator including a fuel management panel, a panel for studying beam experiments, and a panel for examining the ventilation systems of the reactor. Selecting a new view activates one or

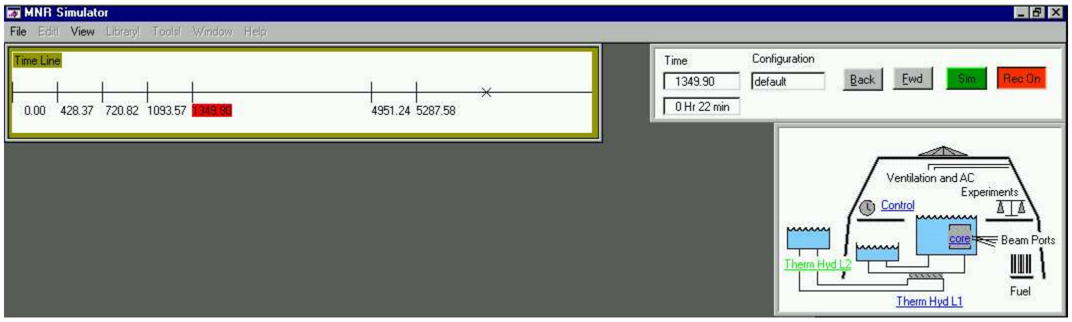


Figure 2: MNRSIM time-line and reactor schematic

more sub windows in the lower left of the screen, but the time-line and schematic always remain visible. The simulation also supplies a set of VCR like buttons for controlling the time-line. The simulation can be started and stopped at various stages, and can also be rewound and restarted with new control settings.

3 Physics Processes

Two major physical processes are modeled in MNRSIM, reactor core physics, and heat transfer from the core and throughout the cooling system. These models have a variety of limitations, and were designed as coarse approximations only. The reactor core is modeled in two dimensions and only for the steady state case. Transient core physics is modeled simply by maintaining the core flux shape and modifying the total core power.

Thermal hydraulics are modeled with a combination of a steady-state Hardy-Cross model, and a transient heat advection routine which uses the flow rates computed by the Hardy-Cross method. Heat transfer from the core to the heat transport system is modeled in detail on a fuel channel by fuel channel basis, the amount of heat transfered to each channel is determined by the flux shape.

3.1 Reactor Core Physics

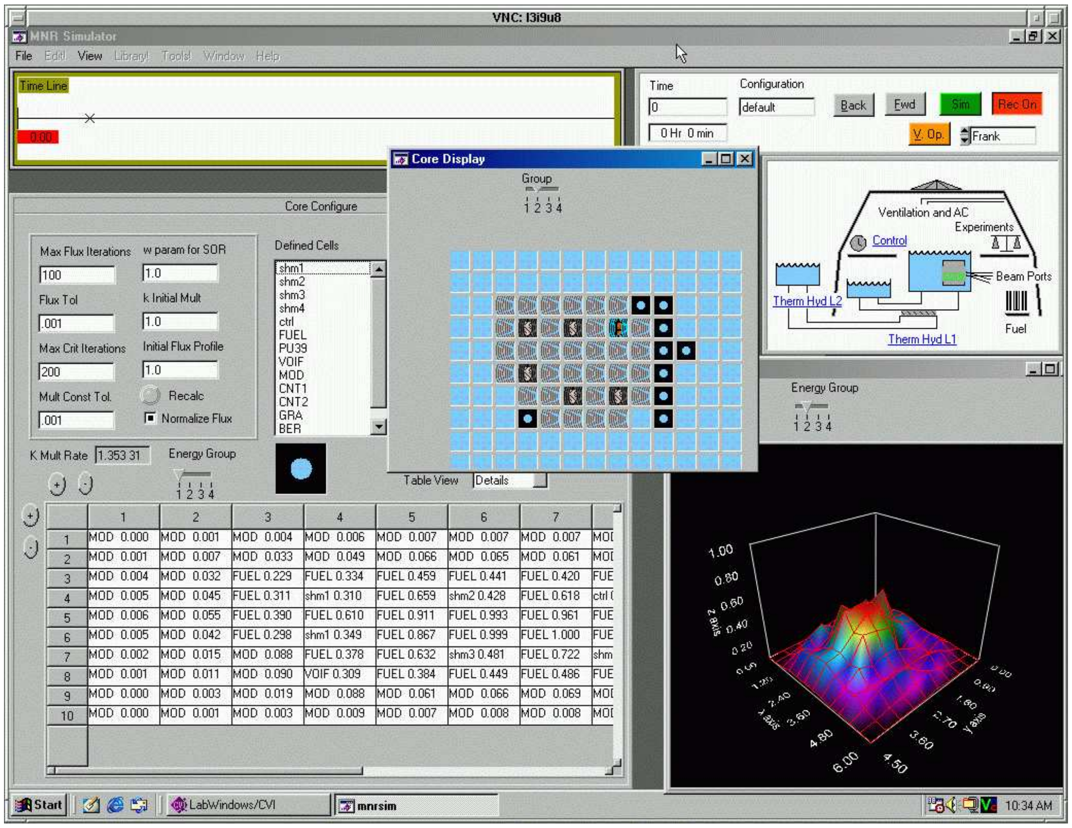
One of the goals for MNRSIM is to incorporate several selectable approximations for each module, depending on precisely what the user is interested in studying. One, two and three dimensional approximations of the reactor core have been planned, but at this stage core physics is only modeled in two dimensions.

The materials cross section module, named *cell* in the code architecture, reads in a simplified data base of material constants extracted from the WIMS-AECL code. Four energy group data is used to represent cross sections from 34 elements typically needed for reactor simulations. A second configuration file defines typical cells in the reactor, such as fuel cells, control rods, irradiation points, and moderator cells based on a proportional weighting.

Currently the flux module only computes a steady-state shape for the core flux based on the neutron diffusion equation. Even though the core mesh is quite coarse, usually on the order of 20x30 cells, the calculation is still slow enough that it is only performed once before the transient modules begin. The multi-group neutron diffusion equation that is solved is derived from [4] and in second order finite difference form is written as:

$$\left[\Sigma_{R_i}^g + \sum_j^J \frac{D_{ij}^g}{\triangle_{ij}^2}\right]\phi_{ig} - \sum_j^J \frac{D_{ij}^g}{\triangle_{ij}^2}\phi_{jg} - \sum_{g'=1}^{g-1} \Sigma_{s_i}^{g'\to g}\phi_{ig'} = \frac{\chi^g}{k}\sum_{g'=1}^{G} \nu_{g'}\Sigma_{f_i}^{g'}\phi_{ig'}$$

Where the superscript g represents the energy group, G is the total number of energy groups, the subscripts i and j represent spatial indecies, $1/k$ and represents the first eigenvalue. ϕ represents the neutron flux, Σ_R, Σ_s, Σ_f the removal, scattering, and fission cross sections respectively, D is the diffusion length, $\triangle$ is the diffusion distance, χ represents the fission spectrum, and ν is the number of neutrons produced per fission. The delayed precursors are not included in the steady state calculation. A simple successive over relaxation solver is used.



Time: 0
Configuration: default
0 Hr 0 min
Frank

Core Configure

Max Flux Iterations: 100
w param for SOR: 1.0
Flux Tol: .001
k Initial Mult: 1.0
Max Crit Iterations: 200
Initial Flux Profile: 1.0
Mult Const Tol: .001
Recalc
Normalize Flux [✓]
K Mult Rate: 1.353 31
Energy Group: 1 2 3 4

Defined Cells: shm1, shm2, shm3, shm4, ctrl, FUEL, PU39, VOIF, MOD, CNT1, CNT2, GRA, BER

Table View: Details

	1	2	3	4	5	6	7	
1	MOD 0.000	MOD 0.001	MOD 0.004	MOD 0.006	MOD 0.007	MOD 0.007	MOD 0.007	MOI
2	MOD 0.001	MOD 0.007	MOD 0.033	MOD 0.049	MOD 0.066	MOD 0.065	MOD 0.061	MOI
3	MOD 0.004	MOD 0.032	FUEL 0.229	FUEL 0.334	FUEL 0.459	FUEL 0.441	FUEL 0.420	FUE
4	MOD 0.005	MOD 0.045	FUEL 0.311	shm1 0.310	FUEL 0.659	shm2 0.428	FUEL 0.618	ctrl (
5	MOD 0.006	MOD 0.055	FUEL 0.390	FUEL 0.610	FUEL 0.911	FUEL 0.993	FUEL 0.961	FUE
6	MOD 0.005	MOD 0.042	FUEL 0.298	shm1 0.349	FUEL 0.867	FUEL 0.999	FUEL 1.000	FUE
7	MOD 0.002	MOD 0.015	MOD 0.088	FUEL 0.378	FUEL 0.632	shm3 0.481	FUEL 0.722	shm
8	MOD 0.001	MOD 0.011	MOD 0.090	VOIF 0.309	FUEL 0.384	FUEL 0.449	FUEL 0.486	FUE
9	MOD 0.000	MOD 0.003	MOD 0.019	MOD 0.088	MOD 0.061	MOD 0.066	MOD 0.069	MOI
10	MOD 0.000	MOD 0.001	MOD 0.003	MOD 0.009	MOD 0.007	MOD 0.008	MOD 0.008	MOI

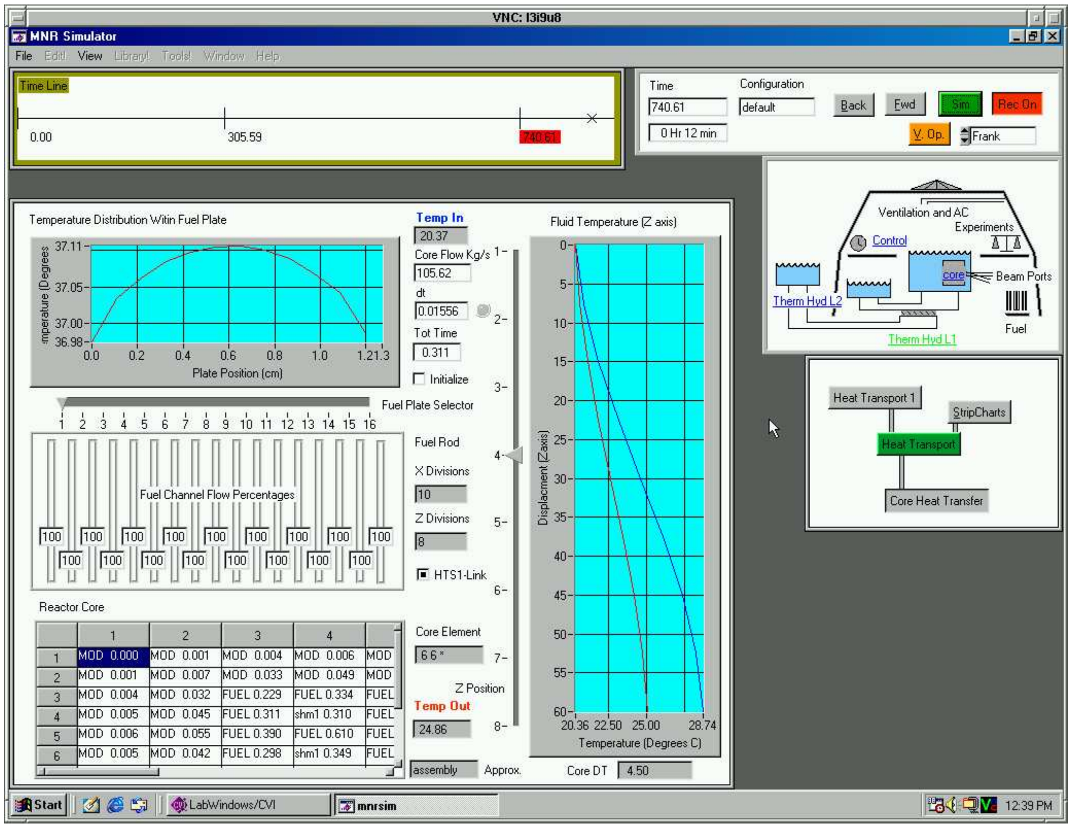
Figure 3: core assembly window

The core assembly window (Fig. 3) allows the user to pick up core elements as defined by the cross section module and arrange them into a wide variety of configurations. Core configurations can be stored on the disk allowing the user to build up a catalog of reactor core testing scenarios.

3.2 Core Heat Transfer

The core heat transfer module uses the two dimensional flux array computed by the flux module to model the transfer of the heat generated in the fuel plates to the coolant. The McMaster Nuclear Reactor uses plate style fuel, with 18 plates per assembly, 16 inner fueled plates, 2 outer dummy plates, and a total of 17 flow channels.

Four approximations to the full core heat transfer calculation are supplied. The user may simply specify the temperature increase which should occur across the core, or may use an approximation which computes heat transfer for a single plate, a single assembly or for the entire core. In the figure (Fig. 4) the left most fuel plate (#1) is selected. The heat profile for this plate is somewhat assymetrical since this plate is the first in the assembly. The heat profile for the coolant channels on either side of this fuel plate are plotted in red and blue.



Time: 740.61 — 0 Hr 12 min
Configuration: default
Frank

Temp In: 20.37
Core Flow Kg/s: 105.62
dt: 0.01556
Tot Time: 0.311
☐ Initialize
Fuel Rod
X Divisions: 10
Z Divisions: 8
■ HTS1-Link
Core Element: 6 6 *
Z Position
Temp Out: 24.86
Approx.: assembly
Core DT: 4.50

Fuel Channel Flow Percentages: 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100

Reactor Core

	1	2	3	4	
1	MOD 0.000	MOD 0.001	MOD 0.004	MOD 0.006	MOD
2	MOD 0.001	MOD 0.007	MOD 0.033	MOD 0.049	MOD
3	MOD 0.004	MOD 0.032	FUEL 0.229	FUEL 0.334	FUEL
4	MOD 0.005	MOD 0.045	FUEL 0.311	shm1 0.310	FUEL
5	MOD 0.006	MOD 0.055	FUEL 0.390	FUEL 0.610	FUEL
6	MOD 0.005	MOD 0.042	FUEL 0.298	shm1 0.349	FUEL

Figure 4: core heat transfer view, showing thermal distribution within plate and channel

Heat distribution within the plate is straight forwardly derived from:

$$\rho C_p \frac{\partial T}{\partial t} = q''' + \frac{\partial}{\partial x} k \frac{\partial T}{\partial x}$$

C_p and ρ represent the density and the heat capacity for water respectively, q''' is the heat generated by the nuclear reaction per volume, T is the temperature, and k is the heat diffusion constant. Several numerical implementations of this formula were tried including, implicit, explicit, semi-implicit and Crank Nicholson. Since the fuel plate is so thin, a very small time step was required to maintain stability in the computation. While both the implicit and semi-implicit formulations allowed for larger time steps, they both tended to produce results which were inaccurate by about an order of magnitude. A Crank Nicholson formulation provided a reasonable compromise between the two extremes of the unstable explicit formulations, and the inaccurate implicit formulations.

Heat advection through the coolant is solved by treating the coolant as a single phase channel. The coolant temperature is used by the plate heat distribution calculation as an external boundary condition. The last plate division is used to compute the surface temperature of the fuel plate, and the heat flux into the fluid volume element. The energy balance within the fluid is defined as:

$$A\rho C_p \frac{\partial T}{\partial t} = -w\frac{\partial h}{\partial z} + q'(z)$$

A represents the plate area w represents the flow rate, h represents the enthalpy of the fluid, q' is the energy transfered to the fluid from the fuel. Implementation of the heat advection formula was straightforward. The current version always assumes that $w > 0$, which cannot model certain core failures where the flow out of the core is stopped altogether and the core flapper opens to allow reverse flow. Water boiling is also not accounted for.

3.3 The Hardy-Cross Method

The Hardy-Cross method [5] is used to compute flow rates within the thermal hydraulic system. During normal operations the transient phases which include opening the pool and hold-up tank valves and starting the primary pump are very short, and at least in terms of flow rates the inertial effects of water starting and stopping motion are negligible. Since Hardy-Cross is a steady state model, a sequence of steady states must be computed as the fluid levels in the reactor pool and hold up tank change. While certain hydraulic effects are neglected by this approximation it was felt that the advantage in terms of real time responsiveness of the model was worth the loss in the precision.

The general momentum equation for flow in a pipe is given as follows:

$$LA\rho\frac{\partial v}{\partial t} = A\left[\Delta P - \left(\frac{fL}{D} + k\right)\frac{\rho v^2}{2g_c} - \rho g \triangle z\right]$$

L, A, and D are the length, cross-sectional area and diameter of the pipe respectively, $\triangle P$ represents the pressure drop across the pipe, f is the friction factor, k is the minor loss coefficient, g is the acceleration due to gravity, h is the drop in elevation of the pipe, v is the fluid velocity and g_c is the mass pressure conversion ratio which $= 1$. Since we are interested only in a steady state scenario we set $\frac{\delta v}{\delta t} = 0$, and we lump together the pressure terms due to a change in height with those induced by the flow so that rather than discussing

$$0 = \Delta P - \left(\frac{fL}{D} + k\right)\frac{\rho v^2}{2g_c} - \rho g \triangle z$$

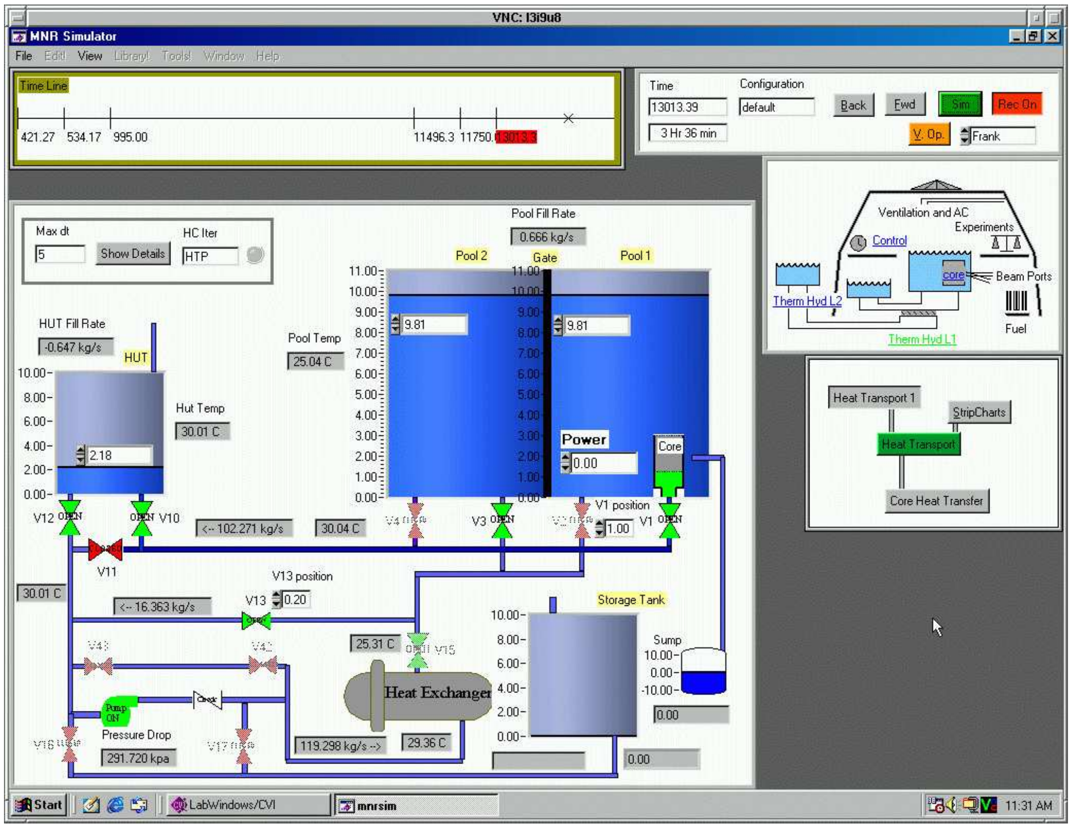


Figure 5: thermal hydraulics, primary loop view

We now rewrite the flow equation as:

$$P^* = P + \rho g \triangle z$$

$$0 = \Delta P^* - \left(\frac{fL}{D} + k\right)\frac{\rho v^2}{2g_c}$$

Rewriting this equation again in terms of flow rather than velocity, and lumping constants together as K we get:

$$K = \left(\frac{fL}{D} + k\right)\frac{1}{2\rho A^2 g_c}$$

$$0 = P_1^* - P_2^* - Kw^2$$

This means that for an arbitrary connection where *#links* pipes meet, the net flow at this connection must be zero, so we now have a general non-linear formula for solving the pressure at every point in the pipe network:

$$P_n^* = \frac{1}{\sum_{j=1}^{\#links} \frac{1}{\sqrt{\left|P_j^* - P_n^*\right| K_j + \varepsilon}}} \times \sum_{j=1}^{\#links} \frac{P_j^*}{\sqrt{\left|P_j^* - P_n^*\right| K_n + \varepsilon}}$$

Since the Hardy-Cross method is not designed to handle situations where the flow rate drops to zero we have included a small $\varepsilon = 0.000001$ in the denominator of the fractions to prevent division by zero. To recover the true pressures from the P^*pressures we simply use $P = P^* - \rho g \triangle z$.

In practice the Hardy-Cross method performs reasonably well. A fixed point iterative solver was used to solve the non-linear system. It was found that if an SOR style acceleration method was applied to the pressure calculations that good convergence was observed.

3.4 Heat Transport

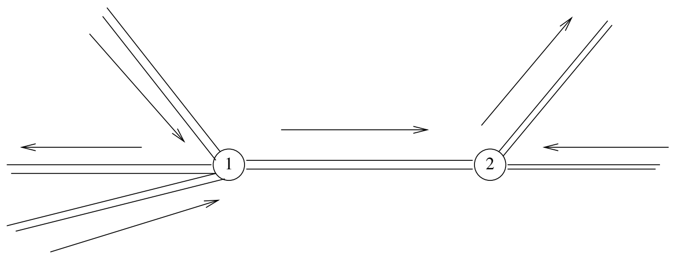


Figure 6: Pipe network example

Heat transport is computed using a second order Crank Nicholson time integration scheme based on the flow rates computed by Hardy-Cross. Both primary and secondary thermal hydraulic loops are computed. The basic heat transport equation used takes the following form:

$$C_p \rho V \frac{\Delta T_i}{\Delta t} = w_{in} C_p T_{in} - w_{out} C_p T_{out} + Q_i$$

V represents the volume of the pipe or vessel, and Q_i represents the amount of heat removed through the walls of the pipe. This equation was implemented by examining the amount of heat energy that flowed into one end of a pipe and out of the other end of the pipe. Since all flow rates are known as calculated by the Hardy-Cross method solving for the temperature of a given volume of water contained in one of the pipes turns into a book keeping problem.

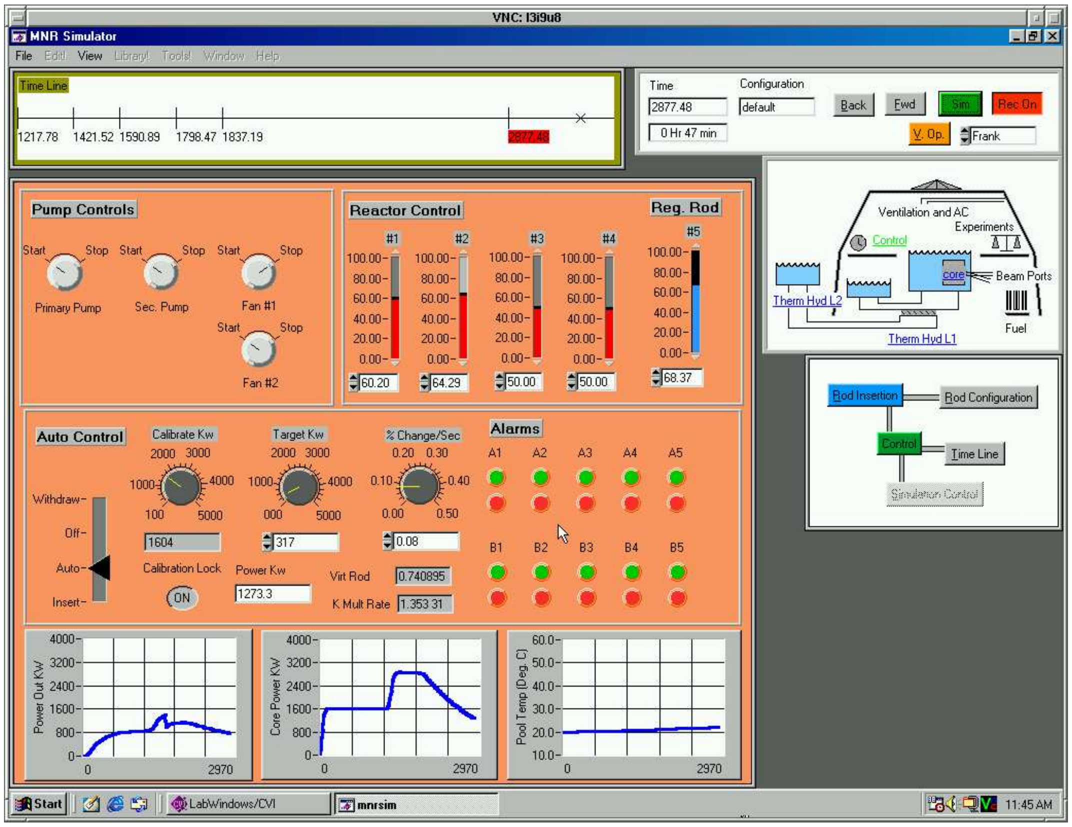
In Figure 6 fluid is flowing in a pipe with end points 1 and 2, from 1 to 2. To compute the change in enthalpy for this pipe we use the following equation:

$$\frac{\partial H}{\partial t} = C_p \sum_{i}^{\#links} w_i T_i$$

The data structures in the program must maintain a convention which allows the program to correctly determine the sign of the flow for connected pipes. The same convention is used to determine which end the pipe gains heat energy from. In Figure 6 since fluid is flowing out at end 2 the energy lost from pipe 1-2 will be part of the net gain for the remaining pipes connected to end 2. Likewise the net flow into end 1 of pipes 1-2 will represent the total energy gained at this end of the pipe. The Hardy-Cross method guarantees that the positive flow from end 1 to end 2, will be made up of the sum of all flows into end one, so the gain in enthalpy of the pipe can be deduced from the flows.

Implementing this model also required the use of a Crank Nicholson approximation to achieve adequate accuracy. As in the heat transfer module explicit methods required excessively small time steps to remain stable, and implicit methods had accuracy problems. The Q_i term which represents heat losses which are not due to fluid flows (radiative, or conductive) is only implemented for the heat exchanger. Much energy is lost to the atmosphere of the containment building and this phenomena is not modeled.

3.5 Control Panel



Time: 2877.48
0 Hr 47 min
Configuration: default
V. Op.: Frank
Reactor Control: #1 60.20, #2 64.29, #3 50.00, #4 50.00
Reg. Rod #5: 68.37
Calibrate Kw: 1604
Target Kw: 317
% Change/Sec: 0.08
Calibration Lock: ON
Power Kw: 1273.3
Virt Rod: 0.740895
K Mult Rate: 1.353 31

Figure 7: Control Panel Mock Up - Currently Incomplete

The control panel provides the MNRSIM operator with a basic mock up of the actual MNR control panel. The sliders and knobs adjust the boundary conditions of the various modules and give an operator's point of view for the simulation. Several strip charts allow the operator to track trends and help to identify potential problems with the system. The current version of the control panel is incomplete, the alarms are inactive, and several important controls are missing from the panel.

4 Calibration and Validation

Model validation has been fairly simple at this stage. The MNR staff[1] graciously supplied several weeks worth of data recorder information from the winter of this year to use as calibration data. The data recorder monitors temperatures at several key points in the reactor, this information was used to both calibrate and verify the primary and secondary thermal loops.

time	core.out	core.in	P.out HX	2.out HX	2.in HX	Pool	Air	Power	Flow	Pool	HUT
	C	C	C	C	C	C	C	%	KG/s	meters	meters
14:00	27.4	27.44	22.11	n/a	22.83	28.06	-20	90	98.5	9.25	2.27
15:00	31	26.78	22.06	20	16.67	27.72	-16	91	98.5	9.25	2.27
16:00	31.5	26.28	22.00	n/a	17.00	27.11	-15	93	98.5	9.1	2.31
17:00	31.8	26.67	21.94	20.28	17.17	27.11	-15	93	98.5	9.1	2.31

Table 1: Example Thermal Hydraulic Data

From previous studies and several internal reports it was also possible to collect information on the lengths and diameters of the major pipe components and the core design. When the model's core is calibrated to 2 MW, we are able to observe the typical temperature rise of about 5.5 degrees Celsius across the core, and we also observe a .5 degree temperature span within each fuel plate. These values are in good agreement with standard measurements and calculations (roughly +-5%).

Several parameters of the simulation are treated as unknowns since in practice they cannot be easily measured or calculated. These constants were adjusted to calibrate the model. Among these are:

- minor loss friction coefficients for both primary and secondary sides of the heat exchanger
- minor loss friction coefficient of the reactor core

The friction coefficients determine the flow rates on the primary and secondary sides, the geometry of the core and heat exchanger are too complex to permit a direct calculation of its friction coefficient. Also the friction coefficient of the heat exchanger has changed over time due to the

[1]Robert Pasuta, MNR analyst

build up of particulate matter from the coolant. This effect is especially prevalent on the secondary side.

Our core model only computes a shape for the flux profile at this stage, so the core flux model has only received basic verification. The flux shape shows reasonable agreement with closed form models, and also shows reasonably good agreement with simulation runs on more detailed models. The next stage in model development will include much more rigorous testing and calibration. Currently we consider a computed value to be in reasonable agreement if it falls within 10% of a measured value.

5 Conclusion

The attempt has been to design a tool which will help an operator to understand what the result of a decision would be if a certain course of action were taken. This strategy is helpful for both the novice learning the control and behavior of the reactor, but also potentially for the expert who needs a quick cause and effect sketch so the results of certain actions can be estimated.

This is an open project which will hopefully continue for many years; the work which still needs to be done is extensive. The model needs much more careful validation, in particular, comparisons with more detailed simulations will provide helpful insight into correcting the model's behavior. The software design is intended to encourage collaborative work by clearly defining interfaces for interacting modules. Two projects which plan to use the simulation include a failure modelling system which uses the simulation to build up a history of simulated failure scenarios. A second proposed project is the detailed modeling of the MNR's control panel and alarm system. Other suggestions include detailed modeling of fluid flows in the core, and in the pool, as well as detailed air flow models within the building.

References

[1] CAE, "Worldwide integrated training solutions." http://www.cae.com/en/general/index.shtml, 2004.

[2] C. S. International, "Innovative simulation tool for nuclear power plant." http://www.cti-simulation.com/ctisimulation/nuclear.htm, 2004.

[3] D. Gilbert and W. J. Garland, "MNRSIM: A Research and Training Model of the McMaster Nuclear Reactor." http://nuceng.mcmaster.ca/mnrsim, 2004.

[4] J. Duderstadt and L. Hamilton, *Nuclear Reactor Analysis.* John Wiley and Sons, 1976.

[5] L. Mays, *Hydraulic Design Handbook.* New York: McGraw-Hill, 1999.