ANL-ARC-208

Simulator Platform for Fast Reactor Operation and Safety Technology Demonstration

Nuclear Engineering Division

ANL-ARC-208208

Simulator Platform for Fast Reactor Operation and Safety Technology Demonstration

R.B. Vilim, Y.S. Park, C. Grandy, H. Belch, P. Dworzanski, and J. Misterka
Nuclear Engineering Division
Argonne National Laboratory

September 15, 2011

ABSTRACT

A simulator platform for visualization and demonstration of innovative concepts in fast reactor technology is described. The objective is to make more accessible the workings of fast reactor technology innovations and to do so in a human factors environment that uses state-of-the art visualization technologies. In this work the computer codes in use at Argonne National Laboratory (ANL) for the design of fast reactor systems are being integrated to run on this platform. This includes linking reactor systems codes with mechanical structures codes and using advanced graphics to depict the thermo-hydraulic-structure interactions that give rise to an inherently safe response to upsets. It also includes visualization of mechanical systems operation including advanced concepts that make use of robotics for operations, in-service inspection, and maintenance.

Table of Contents

LIST OF FIGURES

1 Introduction

The fast reactor in principle can be configured via its underlying nuclear technologies to operate as an energy source that is very-nearly sustainable with respect to fuel supply. To improve economic viability, however, continued development and innovation of the underlying technologies is needed. These technologies span the disciplines of reactor safety, fuel cycles, and reactor operations. The technologies involved must work together so that maximum energy is extracted from the heavy metal elements, the waste stream loading is minimized, favorable reactor safety characteristics during operation are obtained, the mechanical support systems operate efficiently and reliably, and the fuel form supports safe and efficient handling.

Given the complexity of this system, computer-based visualization methods can help with better understanding the interactions among the component technologies. Visualization tools can serve as aids to the non-specialist, i.e. an individual not involved in the intimate details of fast reactor design. Presently the only alternative for acquiring an understanding of the concepts is a detailed reading of the technical literature i.e. papers and reports. The non-specialist individual can appear in several roles and capacities. There is the program sponsor who can be better served through a high-level interactive rendering of the fast reactor deliverable. There is the equipment operator whose performance can be improved through training on a virtual representation of the system he will eventually operate. Traditionally an operator is trained on a physical mock-up of the equipment. Training on a virtual representation of the equipment can proceed more quickly and at reduced cost. Then there is the licensing agency. An interactive virtual rendering of the facility and its operation can provide a better picture of the overall system and serve as a basis for focusing discussions between the submitter and the agency. Additionally an integrated simulation capability can provide a common point around which specialists and designers can interact in the course of optimizing a design.

2 Objective

This multi-year project is to develop a state-of-the art software platform as an aid to visualization of fast reactor technology advances and innovations. The project is to make use of recent and expected future developments in high-fidelity plant simulation software and display hardware to achieve a whole-plant simulation capability. The simulator will provide an environment for studying and demonstrating how technology advances in passive safety, instrumentation, control, and mechanical and fluid systems integrate at the plant level making for a safer and more reliable and economic plant.

Long term goals include:

- Simulating, in real-time, the performance of the extant reactor condition to design basis and beyond design basis accidents to demonstrate inherent safety performance.

- Visualization and support of specific operations including refueling and in-service inspection and repair operations.

- Visualization of the plant behavior and performance capabilities arising from the integration of individual component technologies.

- Demonstration of improved operation through use of computer-based operator aids for inspection, diagnostics, and control.

- Demonstration of opportunity for economic performance improvement achievable through high fidelity models, particularly through margin recovery by better prediction of plant condition.

- Demonstration of the simulator as a training tool to accelerate reactor operator training and maintenance training and operations.

Objectives in FY11 were:

- Supercritical Carbon Dioxide Balance of Plant - Install balance of plant model for simulating S-CO_2 cycle. This requires linking the G-PASS/S-CO_2 code to SASSYS, developing a G-PASS/S-CO_2 code input deck for the ABTR 2006 primary system, and interfacing operator panels to G-PASS data output using Python.

- Parallel Processing for Real Time Simulation - Balance of plant simulation codes for energy conversion cycles that make use of a change of phase or operate near a super critical point run slower than real time, principally a result of numerically intensive calculation of thermodynamic properties. One solution to obtain real-time simulator performance is to run the G-PASS code on multiple cores. This is a software architecture problem. We are to begin venturing down the parallel processing path by developing a multi-core parallelized solution for G-PASS.

- Robotic and Related Mechanical Systems Visualization - Process technologies such as refueling that rely on coordinated management of equipment in time and space lend themselves to simulator display using 3-D motion programming. A 3-D CAD driven model driving visualization software will be explored. A multi-modal sensory feedback experience will tested using a combined 3-D video / haptic display technology. In addition a high-resolution large-scale display assembly will be assessed for improved information conveyance. Integrated demonstration will provide a virtual experience of plant operation, inspection, and maintenance.

- Plant Thermodynamic State Diagram - Develop graphical means for displaying coolant state properties for balance of plant and for sodium system. This graphic will appear alongside the current rendering of plant conditions that are displayed as numerical values in text boxes. The state diagram is a graphical means for understanding plant state that is an alternate to process variable values displayed in text boxes.

3 Technologies and Demonstration

This section identifies innovative technology concepts in safety, operations, in-service inspection, and maintenance. There are eight innovations in total, each potentially a candidate for demonstration on the simulator.

3.1 Safety

The reactor must operate safely to protect the plant investment and the public. This will be accomplished if the fission products are retained within the fuel pins over their life. This can be achieved by limiting the temperature of the fuel pins to prevent breach of the mechanical boundary that is the cladding. But in limiting temperatures economic penalties are incurred in the form of margin and protection system complexity. This subsection describes innovations for reducing the size of the penalties.

3.1.1 On-Line Limiting Safety Settings

The active protection system is independent of the plant control system and serves as a diverse means for preventing core temperatures from moving outside safety limits. It is designed to protect against upset events resulting from operator error and equipment failure. At normal operation, as a consequence of various uncertainties, the actual cladding temperatures are purposely set significantly below safety limits. This "margin" is included to eliminate reactor trips that would otherwise result from uncertainties associated with inferring the operating state of the reactor. The values of process variables at which the protection system trips the reactor are referred to as *limiting safety settings*.

In practice during normal operation margin results in the reactor outlet temperature being tens of degrees Celsius below what would be the case if the actual reactor conditions were known exactly. There are two components to the margin. The first accounts for uncertainties in the model used during plant design to predict full-power operating temperatures. The second accounts for anticipated variability in the configuration of the reactor at full power and its effect on operating conditions (e.g. control rod position and its effect on local power).

The margin component associated with the model uncertainties is potentially recoverable. It can be reclaimed through more precise estimation of peak cladding mid-wall temperature in combination with on-line measurements. A high-fidelity model provides a basis for core thermal margin recovery. Margin can be recovered if an improved estimate for peak cladding mid-wall temperature is obtained. The high fidelity model in combination with real-time temperature measurements at the outlet of the core subassemblies at a selected number of points in the core can provide the improved estimate.

In a plan for margin recovery by adjusting limiting safety settings while at power, the digital plant instrumentation system provides the required computational infrastructure. Uncertainty analysis methods provide the statistical distribution of the estimated temperature in the cladding at all points in the core as derived from the model and data. This is used to generate

an on-line operations-based estimate for the peak cladding mid-wall temperature. The estimate is statistical and hence a confidence level can be stated. A fuel pin cumulative-damage model takes this as input and determines the permissible power level such that the cumulative damage over fuel pin life is acceptable. The limiting safety settings are then set. Margin is recovered by use of on-line models combined with subassembly outlet temperature measurements to yield a more accurate estimate for core conditions than is obtained using only reactor outlet temperature measurement which is the current practice.

This technique for recovering margin by adjusting limiting safety settings as the plant operates can be displayed on a visualization platform as an aid to understanding the process of margin recovery.

3.1.2 Inherent-Core Protection

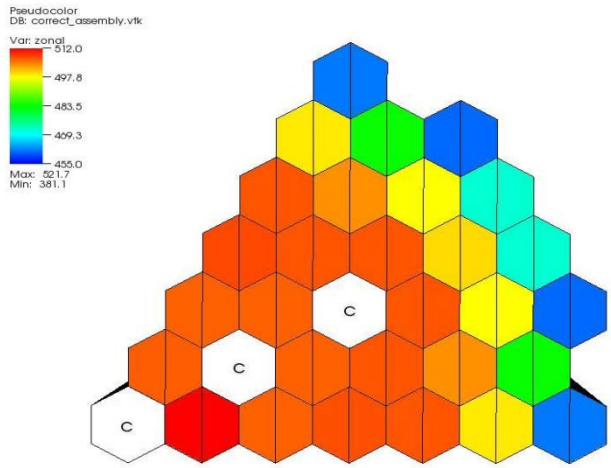
The probability that core temperatures will exceed safe limits during an unprotected upset event can be reduced by designing the reactor core so that temperature feedbacks operate to inherently limit the size of any temporary imbalance between heat generation and removal rates. The size of the imbalance can be limited through choice of fuel form, engineered core restraint system, and core layout.[1] Through such choices the frequency over life with which peak temperatures approach safe limits is reduced and can be used as a basis for simplifying the protection system to realize a cost savings.

The self-regulating behavior is the result of a reduced fission energy release as structural materials expand in response to coolant temperature increase. The coolant heat up and the structural changes can be displayed visually on a visualization platform as an aid to understanding this process. In addition, the visualization platform can incorporate real-time data into the model to establish the extant reactor condition. The reactor operator can then run various transient simulations that will demonstrate to management and regulatory authorities that the reactor plant continues to meet the specification of inherent safety. This simulation can be run for design basis and beyond design basis events without putting the actual plant and investment at risk.

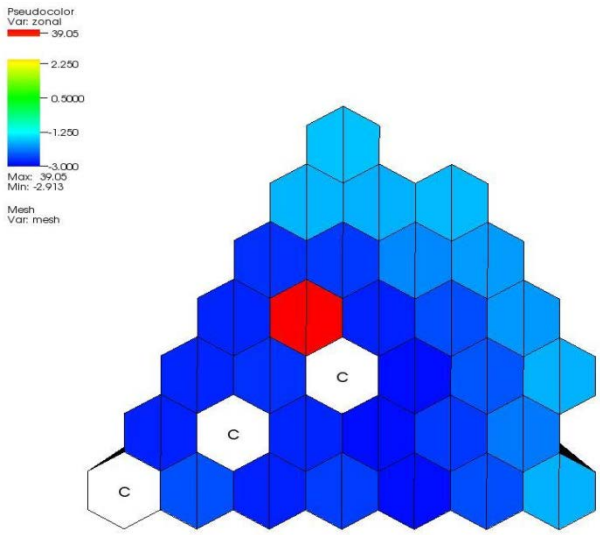
3.1.3 Detection of Subassembly Coolant Misallocation

The margin recovery innovation of Section 3.1.1 has applicability to the detection of subassembly coolant misallocation. A misallocation of flow occurs when a subassembly is placed in a core lattice position for which it is not appropriately orificed. For example, a fresh driver placed in a lattice position orificed for a once-burnt driver would be under cooled. Unless the subassembly is instrumented with a thermocouple at its outlet, this event will go undetected. An instrument at the outlet of each subassembly has its disadvantages, however. An alternate approach recognizes that misplacement results in a perturbation to normal operation and can be detected by comparing outlet temperatures from a subset of instrumented subassemblies against model predictions for temperature. A discrepancy larger than the uncertainty of the prediction indicates a coolant misallocation fault. The fineness to which a misallocation can be resolved is dependent on the number of subassembly outlet temperature measurements, their precision, and the accuracy of the high fidelity model simulation.

The technology for detection of coolant misallocation can be demonstrated on the visualization platform.



Subassembly Outlet Temperatures At Full Power (^{o}C)



Differential in Subassembly Outlet Temperature Arising with Misplacement of a Single Subassembly (^{o}C)

Figure 3.1 Subassembly Outlet Temperature Perturbation Resulting from Misplaced Subassembly

The viability of this approach is suggested in Figure 3.1 where a multi-assembly simulation is shown for a one-sixth core of the ABTR.[2] The left side of the figure shows the core temperature distribution by subassembly during normal operation. The right side shows the result of a single subassembly (among the whole 360^{o} core) placed in a wrong orifice zone resulting in an under-cooling of about 25 percent. The outlet temperatures in all but the misplaced subassembly are reduced by 2-3^{o} C compared to the case of no misallocation. A systematic bias of this size across multiple subassemblies is statistically significant and can form the basis for detection of misplaced driver assemblies during startup of the reactor.

3.2 Operations

Nuclear plant performance is measured in part by how well the plant can operate to meet grid demand, to startup and shutdown for maintenance and refueling, and the degree to which failing equipment can be taken out of service on a planned rather than unplanned basis. This subsection describes how recent plant innovation, new plant missions, and monitoring innovations can impact operational performance.

3.2.1 Advanced-Energy Conversion

The supercritical carbon dioxide (S-CO_2) cycle [3] shown in Figure 3.2 is under investigation as an alternative to the Rankine energy conversion cycle. The interest in this cycle stems from

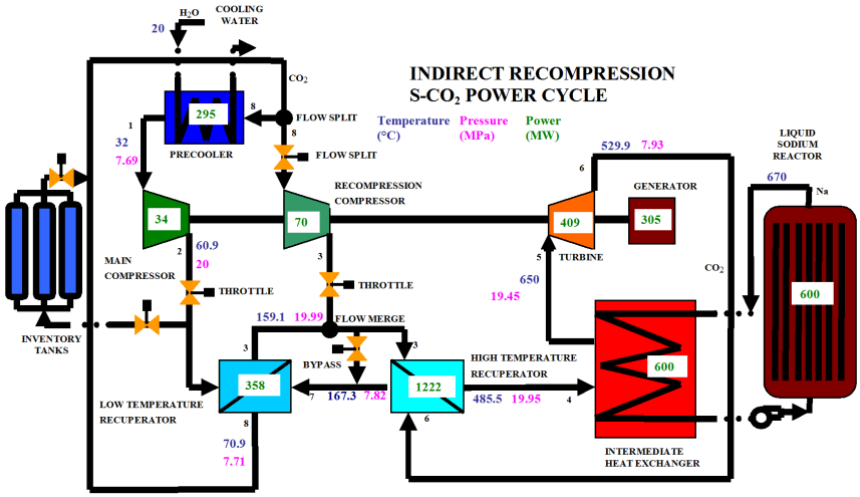


Figure 3.2 Schematic of Supercritical Carbon Dioxide Power Conversion Cycle

the potential for greater thermodynamic efficiency and lower equipment cost. In part these advantages arise from the attractive compressor performance with S-CO_2 as the working fluid. Work in the compression leg of the cycle is less and the component size is reduced compared to, for example, the Rankine cycle.

An important issue is the control and stability of the cycle. The highly non-linear properties of carbon dioxide near the critical point can potentially make control of the cycle based on temperature and pressure measurements more difficult compared to the Rankine cycle. [4] In addition the efficiency of the cycle at partial power is sensitive to the control strategy. Inventory control yields high efficiency but rates of power change are limited by the capability to store inventory. Bypass control operates very quickly but gives poor cycle efficiency performance. Another issue is how performance varies with nearness of compressor operating point to the critical point and with the temperature of the ultimate heat sink.

Visualization methods can aid in the development and demonstration of the S-CO_2 cycle. An appreciation for the novel aspects of S-CO_2 components is a prerequisite for an understanding of plant operating characteristics. Visualization should make available 3-D images of the plant piping and component layout, heat exchanger internals and manifolds, and turbomachinery. Images of this type are routinely generated by the CAD software used to design these components. Process variables and performance measures presented interactively rather than batch mode can facilitate an understanding of cycle characteristics. In addition, a real-time simulation tool can be used by operators to understand the performance of the S-CO2 system on the overall reactor system during maintenance and operational transients.

3.2.2 Non-Base Load Plants

A potential synergy exists between nuclear energy and renewable energy. Wind and solar energy have the potential to contribute to a more sustainable energy base as the store of fossil fuels is depleted. Before wide-spread market penetration can occur, however, the challenge of uncertain production rates due to seasonal effects, weather-related effects, amount of daylight, etc., which prevents use of renewable energy as a base-load energy source, will have to be solved. Small nuclear reactors can complement renewables by providing a load-following source of energy that can compensate for renewable energy's inherent variability. Nuclear power provides a potential solution to the intermittent availability problem.

However, the quasi-stochastic time-varying nature of the mismatch between demand and production that nuclear would fill is different from the historical stable base-load generation role of nuclear power. New design options are needed and must be explored to support this altered mission. First, the use of innovative heat exchangers might be used to eliminate high thermal stress points such as tube sheets during rapid load changes. Second, any design concept that shifts temperature change from the nuclear plant, particularly the primary system, out into non-safety grade systems is preferred from a material stress standpoint. A large thermal inventory such as a molten salt reservoir might permit the nuclear plant to run at near full capacity, re-charging the thermal reservoir during times of peak renewable energy production and directly feeding the grid at time of reduced renewable energy production for production load leveling. Third, reactor core thermal feedbacks can be engineered through the selection of structural materials and fuel type to safely limit temperatures so that the reactor is immune to essentially any operator error or control system failure. Fourth, heat transport paths and coolant choice in concert with core design can provide for near-autonomous load following operation. Fifth, through the use of internal conversion a core can be designed to maintain power for up to 15 years with only nominal reactivity shim to make up for design uncertainties.

The benefits of interactive visualization methods for the development and demonstration of innovative design concepts for non-base load plants parallel those identified for advanced energy conversion concepts above. The methods provide an improved means for understanding and viewing relationships in a design space that is large compared to that of mode traditional reactor applications. In addition, the simulator provides management and operations with a tool that allows them to understand and predict the behavior of the overall integrated plant during operations, maintenance, and upset events.

3.2.3 On-Line Diagnostics

Nuclear plant performance can potentially be improved if failing equipment can be taken out of service on a planned rather than unplanned basis. Early detection of impending failure provides greater flexibility for maintenance planning. With new reactor missions where fuel cycle lengths of ten or more years have been proposed, the reliability of instrumentation will become a larger factor in plant availability. It will be important to be able to detect instrument incipient failure and to provide redundancy or fault tolerant features to avoid shutdown.[5]

Innovations in monitoring and prognostics to this end should be developed. For mechanical components that are contributors to plant unavailability condition-based maintenance can be used to trend and monitor the component. Signals such as motor current, valve position, and vibration provide signatures for determining equipment condition and on-set of failure. Uncertainty analysis and prognostic methods can be used to signal onset of failure well in advance based on the deviation of actual performance from projected performance. [6, 7] A capability to estimate a process variable using analytic redundancy and to estimate its statistical properties using uncertainty methods presents an opportunity to quantitatively test either a measurement or a process for consistency with assumed behavior.

3.3 In-Service Inspection and Maintenance

To protect the plant investment and to assure reliable plant operations, routine in-service inspection of structures and periodic maintenance of equipment is essential. The mechanical operations involved in these tasks are thus critical to reactor operation. Performing them in a liquid-metal reactor is, however, challenging because of the opaqueness, temperature, and reactive nature of the coolant. There then is a role for innovative technologies that can reduce the likelihood of error and increase the efficiency of how tasks are executed in an environment where direct viewing may not be possible. One promising approach is the use of *augmented reality* described later in this report.

3.3.1 Inspection and Servicing

Routine and non-routine in-service inspections are required to detect early failures in plant structures and provide support to maintenance and component servicing. [8] Since direct visual inspection is not possible due to opacity of the sodium coolant, there is a need for alternative inspection technologies. As a case, the visual inspection of structural integrity can be replaced by positional indexing with a mechanical device (e.g. fuel handling machine or a robotic mechanism) - failure of an index point to be located within prescribed bounds indicates a structural problem. It may be possible to demonstrate haptic (sense of touch) feedback to the operator to improve the dexterity of the indexing operation for better accuracy and safety. Haptic feedback can also be instrumental for surveillance in part of component servicing operations under sodium environment. The added sense of touch may supplement low visibility. To facilitate close-up visual inspection in support of non-routine maintenance, the ultrasonic sensor array is expected to provide high resolution visual image. However, the ultrasonic imaging arrays currently under development are complex systems and require major installations. A physically compact image scanning is possible by utilizing a single range-finding sensor under robotic control. An investigation of this approach is described later in this report.

A visualization platform can serve four purposes in the development and demonstration of inspection and servicing technology. First, operator training is important for safe and efficient execution of plant procedures. An operator can work and experience the indexing operation on a visualization platform that simulates the mechanical devices and the environment. His training can proceed in stages, first where the target is in view and then later, in a setting akin

to under-sodium operation, tasks are performed having only haptic feedback. Second, the use of ultrasonic ranging for indexing can be simulated for demonstration and prototyping of human factors. Models of 3-D surfaces and a sensor are used in place of the actual environment. Third, robotic imaging described above can be similarly demonstrated and prototyped. Fourth, the simulator can aid in the in-service inspection task by providing simulated data of actual reactor conditions and comparing this data to real-time data from under-sodium viewing images (as an example).

3.3.2 Refueling

Reactor refueling is a mission-critical procedure that involves shutting down the reactor and losing electrical power production. Refueling needs to be performed in a timely manner but at a measured pace so that the chance for error is minimized. These somewhat conflicting goals are complicated by the fact that the operation must be performed under sodium. Currently refueling is a semi-automated process using open-loop control based on interlocks. Accidents reported have involved misalignment, inadequate pulling, and jamming – each causing costly down time and safety issues. Extrication from such events can be difficult. There are limited technology options to effectively recover from accident situations such as fuel jamming, breakage, and dropped fuel.

Technology innovation is possible for improving the fuel handling operation by providing artificial sensory feedback (virtual fixture) in human-machine shared control. In addition, the visualization may provide a multimodal experience of the various incidents of fuel handling operations as part of technology demonstration or training.

The probability of a misalignment can be reduced by shifting toward a closed-loop mode of operation. In this approach, the current state is well characterized in 3-D and is used to constrain future control actions to prevent those that can lead to mechanical interference or misalignment. It can also be computer-rendered as an image of current equipment and part locations for reference by the operator. For example, the operator could have a simulated image of the top of the core linked to real-time data of the location of the fuel handling device to inform the operator of which core assembly is being removed or inserted. An investigation of this approach using the *virtual fixture* concept is described later in this report.

In those refueling operations where the operator guides the trajectory of a component, haptic feedback has a role. The mechanical devices used in refueling are high impedance devices. They are inherently non-compliant and, in the presence of mechanical interference, they can easily result in mechanical damage. An operator provided with haptic (force and tactile sense) feedback can improve dexterity of contact manipulation, thus improving accuracy, efficiency and safety. It can be used for avoiding excessive forces, identify jamming conditions, and sensing proper engagement. An investigation of this approach is described later in this report.

4 Integration on a Single Platform

A long term goal is to bring together on a single display platform the engineering simulation codes used to design a fast reactor and the visualization tools used to assess a design. The end

result is an integrated visual representation of all elements of reactor operation. These elements span power generation, reactor safety and response to design basis and beyond design basis events, refueling, inspection, maintenance, and the fuel cycle. This display concept is particularly suited for technology demonstrations such as refueling, inspection, and maintenance that rely on coordinated management of equipment in time and space. In such applications, 3-D visualization is a useful aid for assessment and evaluation of a design and for demonstration.

The simulation codes and visualization tools installed on the platform are described in the next two sections.

The platform hardware consists of the following components:

- Four 46-inch (diagonal) flat panels.
- Dell Extreme Core I7 Studio XPS
- NVIDIA GTX460 graphics card (2)
- Phantom Omni haptic device by SensAble Inc.

Figures 4.1(a) and (b) show an earlier prototype of the display system and the current version, respectively. In Figures 4.1(b), the display shows a 3-D model of the primary system of a fast reactor. The haptic feedback device that appears in the bottom right corner of the photograph is used to guide a refueling machine that is part of the model.

5 Software Development Tools

5.1 Scripting and Threading

The Python scripting language provides several capabilities needed for integrating engineering simulation codes in a real-time environment. Each engineering code exists as an executable that takes input from other codes as it executes and from the operator interface. Python can be used to launch these executables as processes and then to manage their execution so that they run in real time, or on some other user-specified time scale, and to synchronize the exchange of data between the codes. Python scripting capabilities can also be used to manage inter-process communication where the input and output formats may differ among executables. One way to do this is through a file-based handshake protocol described later in this report.

Python can also provide a degree of parallelization of simulation code execution when run on a multi-core computer. This capability is realized by launching multiple engineering codes as separate processes that run concurrently and then coordinating the inter-process communication between these simultaneously executing processes. The operating system automatically creates a separate thread for each process. When run on a multi-core machine, each thread will execute on a separate core so that multiple threads run concurrently.

Python provides capability for drawing and updating operator consoles and application windows. The wxPython development toolkit, a Pythonic proxy of the powerful C++ GUI programming library wxWidgets, has specialized tools for creating application windows. WxPython provides template objects of pre-constructed GUI items, such as the application loop, events, and buttons. This modular design makes the process of creating a highly customized GUI from scratch very quick and relatively simple, and because it is done in Python rather than C++, potentially tedious memory management is a non-issue. wxPython also has a versatile freehand drawing tool called DeviceContext, which was invaluable for drawing, coloring, and animating reactor diagrams.

Python also provides a capability for real-time drawing of data plots. The Python module MatPlotLib is a tool for generating customized plot graphics and NumPy is a numerical methods module. MatPlotLib has built-in support for integration with wxPython, making it easy to draw plots and display them on wx frames. MatPlotLib is in turn dependent on NumPy, which uses memory-efficient arrays to speed up the processes of drawing and redrawing hundreds of data points.

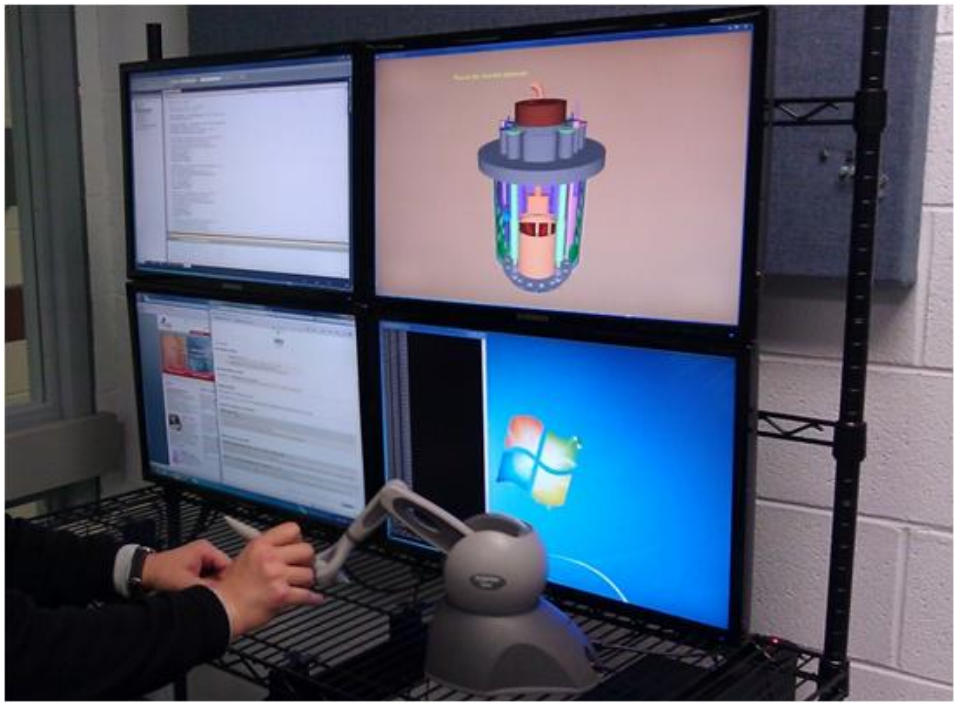


Figure 4.1 (a) Large-Scale Tiled Displays with Haptic Device. An earlier prototype.

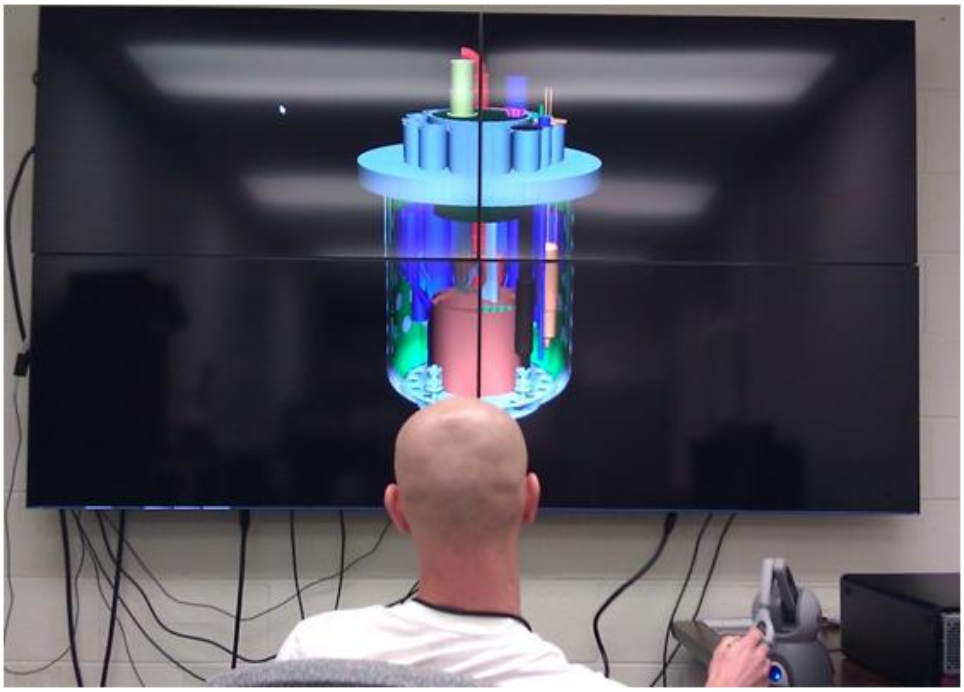


Figure 4.1 (b) Large-Scale Tiled Displays with Haptic Device. Current display.

5.2 Code Parallelization

In addition to the script-based parallelism of concurrently executing engineering codes described above, the otherwise serially executed instructions of each of these codes can potentially be divided up into blocks for parallel execution. There are three different approaches to achieving this. The first two are ideally suited for multi-core machines. The latter is suited for machine clusters.

Auto-parallelization is a compiler capability that identifies regions of code where instructions do not have to be performed in serial but can be executed in parallel. A Fortran DO loop is an example.

OpenMP is a compiler-directive language that permits a user to identify a region of code that is executed more than once and then to execute each instance as a separate thread. On a multi-core machine the operating system schedules threads across available cores so that in practice multiple threads will execute concurrently. Open MP is very efficient with respect to resource utilization. Memory is shared across threads and only one copy of a code block is required. The user must make sure, however, that his code is “thread-safe”. That is, a thread does not unintentionally alter memory of another thread. The trend toward increasing number of cores on a machine increases the attractiveness of OpenMP for parallelization.

MPI is a cluster-based compiler-directive language. Each thread is executed on a separate processor having its own memory. Thus code and memory copying operations must be

performed. As a consequence the speed-up under MPI is limited by the communication bandwidth between the master processor and the slave processors during the copying operations. This presents a bottle-neck that does not exist under OpenMP. In addition for memory updating operations the processor-cache connection exploited under OpenMP is much faster than the analogous connection under MPI.

5.3 Computer-Aided Design

The 3-D graphic models of the reactor and equipment are constructed with the commercial CAD package, Autodesk Inventor, which is also supported with an external motion program API. These part geometry models are then saved into a data format compatible for import into the haptic renderer program, Open Haptics Toolkit by SensAbel Inc.[9]

5.4 Augmented Reality

In the design of the simulator platform an objective is to incorporate an extended form of virtual reality (VR), namely *augmented reality* (AR). In addition to the realistic display of a virtual environment, in augmented reality the environment is displayed with an augmented perceptual overlay so as to enhance human perception and interaction. A key tool for VR and AR implementation are software tools for visual-haptic display. The following tools were used and evaluated in the course of building the simulator platform.

OpenHaptics Toolkit - OpenHaptics Toolkit is the basic haptic renderer program provided by SensAble Inc. with Phantom haptic devices. Equipped with part geometry mesh files, it provides high and low level C++ libraries for implementing visual-haptic feedback with a hand-controller. Haptic attributes – friction, stiffness, and damping – are associated with geometry surfaces. See Appendix A for the depiction of a cone-shaped geometry surface. The haptic interaction is rendered by a haptic cursor entity. The haptic cursor is moved by operating with the haptic hand-controller. Upon touching a surface with the haptic cursor, force is generated and reflected to the hand controller, resulting in sensation of touch.

Initial development work for demonstrating the feasibility of VR and AR was performed with OpenHaptics in the simulation platform. It proved very capable for haptic rendering of simple geometries. However, it lacks capability to define joint kinematics and dynamics which is essential for multiple body system modeling and simulation. Therefore, the alternative software development tools described below were investigated.

Chi3D - Chi3D is an open source set of C++ libraries that allows implementation of visualization and interaction applications. It has a physics engine, namely Open Dynamics Engine (ODE), which can be utilized to simulate the motion of multi-body motion. Specifically, the joints between parts of the reactor fuel handling system were implemented using ODE. Based on the outcome of this effort, a superior tool was identified as described below.

Sensimmer - Our final implementation was achieved using Sensimmer SDK developed by University of Illinois - Chicago. Sensimmer is a comprehensive haptic application

development kit which combines libraries for graphics (OpenInventor), haptics (OpenHaptics), and physics simulation (PhysX).

Open Inventor - For graphics rendering, Sensimmer adopts Open Inventor. In our implementation Coin3d, an open source implementation of Open Inventor, was used. The augmented reality scene is created based on scene graph, which defines inter-relations among graphic objects. Software methods for searching the scene graph and callbacks are provided. The integrated augmented reality scene is created with a number of simple .iv files.

6 Simulation Models

The simulator platform presently runs three dynamic simulations.

6.1 Liquid-Metal Reactor

The primary system of this liquid-metal reactor is configured in a pool-type arrangement with the reactor core, primary pumps, intermediate heat exchangers, and direct reactor auxiliary cooling system heat exchangers all immersed in a pool of sodium within the reactor vessel. The hot primary sodium transfers it heat to an intermediate heat exchanger. The specific plant modeled is the ABTR from [2].

The primary and intermediate system dynamic response are modeled with the SASSYS-1 code.[10]

6.2 Balance of Plant

The balance of plant is an innovative power conversion system using a Brayton cycle with supercritical carbon dioxide. A schematic of the balance of plant is shown in Figure 3.2. The intermediate (non-radioactive) system sodium exits the intermediate heat exchanger and flows to a Na-to-CO_2 heat exchanger. This heat exchanger is the interface between the nuclear heat supply system and the Brayton cycle power conversion system. The intermediate sodium heats the supercritical CO_2 which then flows into the turbine-generator performing work and generating electricity. The CO_2 then goes through a series of recuperator heat exchangers, coolers, and compressors before it re-enters the Na-to-CO_2 heat exchanger.

The supercritical carbon dioxide dynamic response is modeled with the G-PASS code.[11]

6.3 In-Vessel Fuel Handling Machine

The pantograph fuel handling machine shown in Figure 6.1 provides several distinct and separate motions to manipulate the fuel, control rod, reflector and shielded assemblies inside of the reactor vessel. The motions represented in the simulation are:

- Raising or lowering of the entire mechanism to apply a required hold down force to the core assemblies surrounding the core assembly being inserted or removed from the core (performed only when the pantograph arm is above the position of interest).

- Horizontal positioning of the pantograph arm above the core assembly of interest via extension or retraction of the pantograph arm linkages (performed only while the entire mechanism is lifted above the core).

- Rotation of the pantograph inside the vessel to reach various positions above the core by rotation of the mechanism on the bearings between the housing and stepped plug (performed only while the entire mechanism is lifted above the core and the arm is retracted).

- Raising or lowering the core assembly by rotation of a screw mechanism in the gripper head which uses a series of shafts and gears inside the pantograph arm (performed when the FHM assembly is in the lowered or "hold down" position).

The dynamic motion of the pantograph fuel handling machine is modeled using the Chi3D code. The model represents the kinematics of the machine and includes the inertia of members and the resistance at joints. The kinematic model was prepared from a design specification generated with the Autodesk Inventor CAD package.

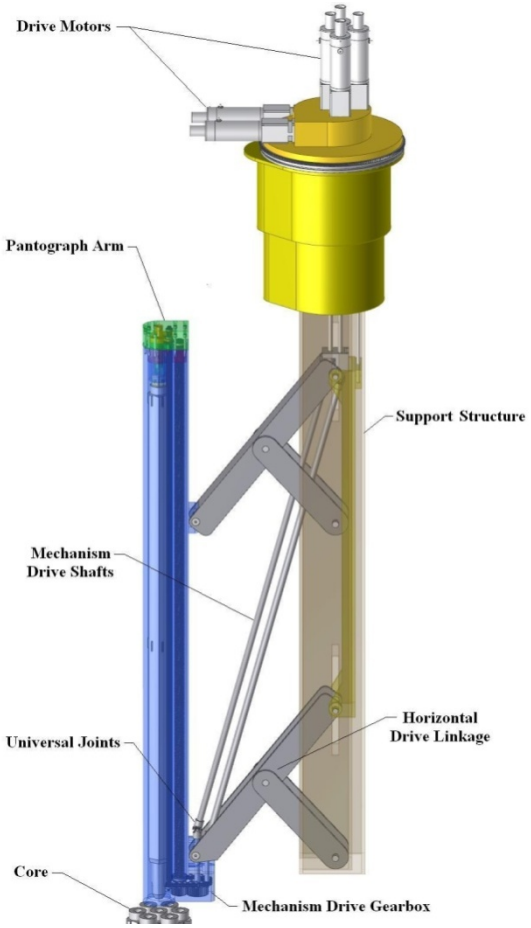


Figure 6.1 In-Vessel Fuel Handling Machine (Pantograph-Type)

7 Platform Implementation

The software architecture for running the above codes and the implementation of the different simulator features are described in this section.

7.1 Architecture

The Python scripting language is used to schedule execution of software tasks on the visualization platform and to process input and output associated with software tasks. A rudimentary database has been built with Python for housing data related to updating the display of SASSYS and GPASS data for visualization.

There simulator program is made up of a network of Python modules that form the three GUI panels. The *Plant Control Station* and *Real Time Status Station* panels are hooked up to SASSYS_RealTime, a version of SASSYS that waits for input data on external timer, simulating a “live” run. The *Safety Test Station* panel is connected to SASSYS_Safety, another

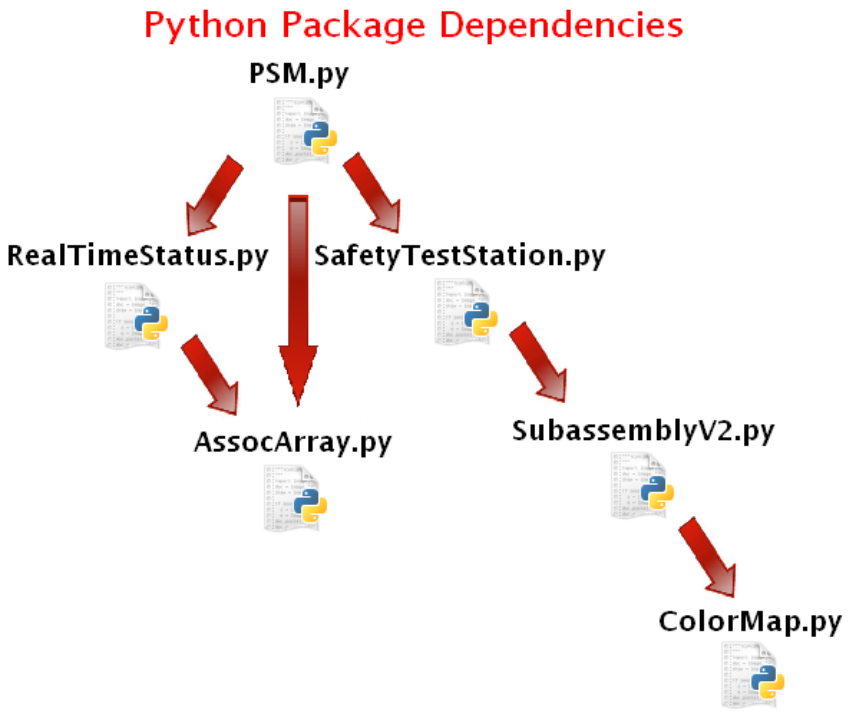


Figure 7.1 Python Module Dependencies

SASSYS variant that takes unprotected accident data as input, runs instantaneously, and returns formatted simulation run data. The existing SASSYS libraries are compiled with distinct sets of new Fortran subroutines to form these two SASSYS versions. Images of the three GUI panels appear in Section 8.

The simulator program architecture is hierarchical with a main Python script (or module) controlling overall execution through the calling of five lower level Python scripts. This organization is shown in Figure 7.1.

The main script PSM.py shown in Figure 7.1 contains the GUI application loop that launches the program. It also contains the source code for the Plant Control Station window and all its associated subclasses. One of these subclasses, called SASSYS, acts as a mediator between the PCS and the SASSYS_Realtime process. All of the variables shared back and forth between the interface and the simulator are stored in the Python SASSYS class in a series of AssocArray2 objects. The AssocArrays all share a common set of keys - corresponding to the set of SASSYS variables the interface uses - and they store everything from the current value of the variable in the simulation to the color of the line on its plot as shown in Figure 7.2. The SASSYS variables themselves are classified as one of two types: control variables or process variables. Control variables are those which are fed to SASSYS as inputs from the GUI. As of this writing, these include rod reactivity and normalized primary and intermediate pump torques. Control variables can be changed via the Manual Control window, which presents the user with a series of slider bars that can be dragged to increment or decrement the variables'

```
# The Control Variables
self.controlVars = AssocArray(('reactivity',0.0),
                              ('primaryTorque',0.0),
                              ('intermediateTorque',0.0))

# The Process Variables
self.processVars = AssocArray(('power',0.0),
                              ('primaryFlowRate',0.0),
                              ('inletTemp',0.0),
                              ('outletTemp',0.0),
                              ('coldTemp',0.0),
                              ('hotTemp',0.0),
                              ('intermediateFlowRate',0.0),
                              ('primaryPumpSpeed',0.0),
                              ('intermediatePumpSpeed',0.0))
```

```
# The line color on the plot of each SASSYS variable as a hexidecimal
# color string. These are arbitrary and can be changed/swapped around
# at the user's discretion.
self.lineColors = AssocArray(('reactivity','#FF0000'),
                             ('primaryTorque','#FF9900'),
                             ('intermediateTorque','#FFFF00'),
                             ('power','#00FF00'),
                             ('primaryFlowRate','#009900'),
                             ('inletTemp','#009999'),
                             ('outletTemp','#00FFFF'),
                             ('coldTemp','#0000FF'),
                             ('hotTemp','#330099'),
                             ('intermediateFlowRate','#FF00FF'),
                             ('primaryPumpSpeed','#663300'),
                             ('intermediatePumpSpeed','#FFFFFF'))
```

Figure 7.2 AssocArrays. The AssocArrays containing the control and process variables (top) and the supplemental AssocArray for plot line colors (bottom) using the same set of keys.

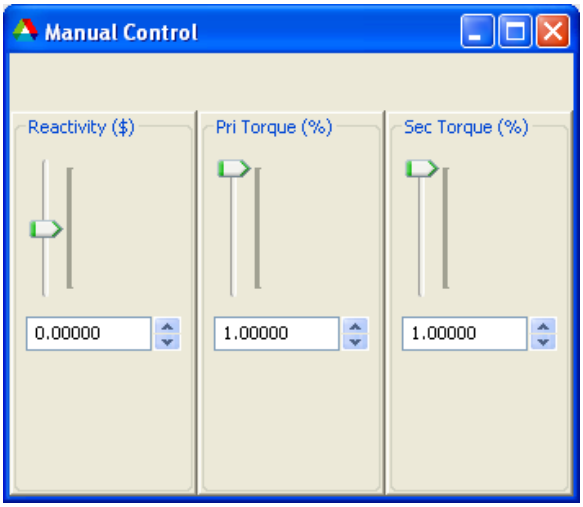


Figure 7.3 Manual Control Window. Screenshot of the Manual Control window. Changing the sliders for each control variable has an immediate effect on the SASSYS simulation.

values as shown in Figure 7.3. Process variables are those that SASSYS calculates at each time step and sends back to the GUI to be displayed. These include: reactor power, primary and intermediate flow rates, primary and intermediate pump speeds, reactor inlet and outlet temperatures, and intermediate heat exchanger cold side inlet and outlet temperatures.

At the start of a time step in the Plant Control Station, the program traverses the AssocArray of control variables and writes their values to SASSYSInput.dat. On the back end of the time step, after SASSYS_RealTime has run its calculations, the Plant Control Station reads the data in SASSYSOutput.dat into the AssocArray of process variables. It then traverses these two AssocArrays and plots each control and process variable in turn, using the shared set of keys to efficiently access the display name, units, and line color of the plot in order to draw it as shown in Figure 7.4.

RealTimeStatus.py is imported as a module by PSM.py. It contains the source code for the RealTimeStatus class, or the Real Time Status Station GUI frame. The RealTimeStatus class must be spawned as a child of the PlantControlStation class, since the data which it uses to configure the reactor diagram display is read directly from the AssocArrays in the

```
def handshake(self):
    """Coordinate communication between SASSYS and the GUI on timestep."""

    # Initialize timestep variables
    self.now = self.refreshRate * self.iters / 1000.0
    self.iters += 1

    # Call SASSYS functions to send input file
    self.SASSYS.configure()
    self.SASSYS.write_SASSYS_input()

    # Poll SASSYS output file until new data is found
    tries = 0
    while 1:
        # If we are able to get output data...
        try:
            self.SASSYS.get_SASSYS_output()
            # ... plot that data and break out of the loop.
            for key in self.SASSYS.controlVars:
                self.plots[key].plot_point(self.now,
                        self.SASSYS.controlVars[key])
            for key in self.SASSYS.processVars:
                self.plots[key].plot_point(self.now,
                        self.SASSYS.processVars[key])
            break
        # If we fail to get output data...
        except NoDataError as e:
        ...
```

Figure 7.4 Handshake() Function. The PlantControlStation handshake() function, which coordinates the data exchange with SASSYS. It calls write_SASSYS_input() to send the data, then loops while attempting the get_SASSYS_output() function. Once it succeeds, it

redraws all the plots with the new data points.

PlantControlStation's SASSYS subclass. The RealTimeStatus class uses the wxPython DeviceContext tool to freehand draw a nuclear reactor diagram. Some parts of the drawing code are dependent on the actions going on in the Plant Control Station, which cause the display to animate by highlighting relevant reactor components in red and. The display is refreshed at every time step of the Plant Control Station, using buffers to prevent flickering on redraws.

SafetyTestStation.py, the source code for the Safety Test Station GUI frame, is also imported by PSM.py and spawned as a child of the PlantControlStation class. Though the SafetyTestStation class largely operates independently of the PlantControlStation class and does not read the values of control and process variables, it still makes use of some of the original SASSYS AssocArray data, such as display names and line colors of variables. Because it interfaces with SASSYS_Safety and not SASSYS_RealTime, it calls on its own separate SASSYS-mediator class. This class writes accident simulation data to SASSYS_Safety, waits until the simulation has finished, then parses the reactor outlet and core temperature data files that come back.

In addition to the primary modules, there are two more supplementary modules used by SafetyTestStation. SubassemblyV2.py is used to generate the hexagonal reactor core diagrams, making heavy use of the DeviceContext drawing tool. The SubassemblyV2 class in turn imports the ColorMap.py to generate color spectrums. The ColorMap class maps a range of numbers to a color spectrum, which defaults to blue-cyan-green-yellow-red unless otherwise specified. Blue represents the low end of the value range, while red represents the high end. ColorMap takes in a number in this range and returns a color in the spectrum relative to its position in the range. The result is that the segments of the subassembly diagram get painted a color corresponding to their temperature, with low temperatures appearing closer to blue and high temperatures closer to red.

7.2 Inter-Process Communication

The sharing of data between these two processes involved the following considerations. Both processes can share the standard input and output (I/O) buffers that store the data written to the shell. Two processes running in the same shell share the same standard I/O, so theoretically, the python and Fortran programs could use these buffers to exchange data. However, SASSYS already uses the standard output buffer to print large amounts of data to the shell, effectively claiming this line of communication. The use of text files, or "flat files", is an alternate means for inter-process communication. Both Fortran and Python can easily handle input and output with flat files, so any file written by one language can be read by the other. This means was implemented in the simulator because of its efficiency and the relatively minor modification to the original code (only a few lines in either language) it requires.

Though this approach solves the problem of where to temporarily store the data, it does require it for proper coordination. Access to these shared flat files must be carefully synchronized,

since two processes cannot read a file at the same time. For the Safety Test Station this is a minor issue. Since SASSYS_Safety does not accept user input during runtime, the GUI can simply wait for the SASSYS_Safety process to terminate before attempting to read and process the

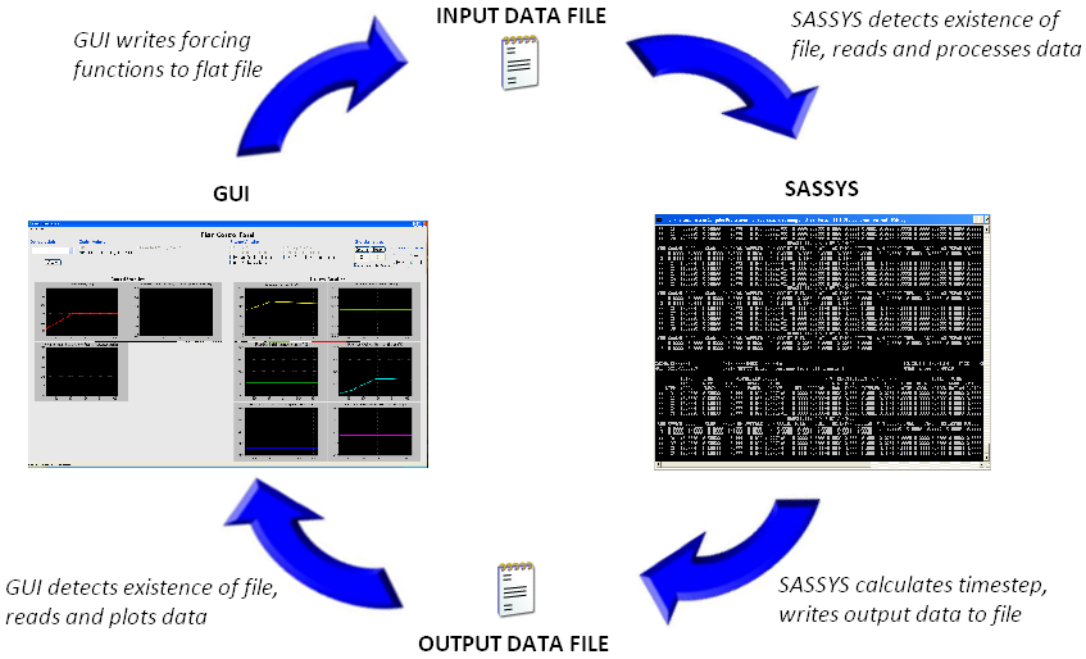


Figure 7.5 Inter-Process Communication Flow Diagram. A flow diagram of the inter-process communication system between SASSYS and the GUI.

output data. However, this same strategy cannot be applied to SASSYS_RealTime and the PCS. Instead, a master timer in the Python code that ticks every half a second coordinates a regular exchange of control flow between the two processes. When a time step starts, the PlantControlStation GUI writes the current values of the control variables to a data file. Until it receives a response from SASSYS_RealTime, the PlantControlStation enters a while-loop that repeatedly checks for the existence of the response data file. The program breaks out of the loop once the file is detected, but otherwise it continuously calls the operating system sleep function, which stalls the program for 50 milliseconds between checks for the file's existence. Meanwhile, the SASSYS_RealTime code contains a similar loop based on the operating system sleep function, waiting to calculate a simulation time step until it receives input data. The resulting effect is a linear transfer of program flow back and forth between the two processes, with one always sleeping while the other is working as depicted in Figure 7.5.

7.3 Code Parallelization

Balance of plant simulations that involve large property changes such as in the Rankine cycle and the super critical Brayton cycle are computationally intensive. Large property changes

require finer nodalizations while property calculations are numerically intensive to begin with. The balance-of-plant in the present version of the simulator is a split-flow recuperated supercritical carbon dioxide Brayton cycle. On a typical single core the simulation of this cycle using the G-PASS code runs about one-third real time.

To achieve real-time simulation performance the G-PASS code was parallelized. The following considerations entered into the choice of the method of parallelization. First, the sequence of code execution is dependent on the loop configuration being simulated, an important consideration in how the code is parallelized. A common feature across all potential balance of plant applications is the appearance of a small library of plant components. At a low level in the code is a sequence of calls to the active components by the solver. The numerical solution scheme is fully implicit allowing for this sequence to be parallelized outright by component since each evaluation of a component's equations is independent of the other components. Further, the allocation of component memory is by a class-like definition using Fortran 90 data-type statements. This facilitates ensuring thread-safe execution of each component's equations.

The above properties led naturally to the selection of OpenMP for G-PASS parallelization. The well defined by-component memory structures are suited to the shared-memory multiple-processor concept supported by OpenMP. The use of MPI in this environment would lead to inefficiencies of code and memory copying. Further, the direction of machine hardware is toward increasing number of cores which will allow for improved future performance under OpenMP.

7.4 Augmented Reality

The simulator includes a visual-haptic model of the fuel handling mechanisms. Implementation of this model is composed of three parts: visual-haptic environment, haptic feedback for fuel handling, and virtual fixture as operation guidance. Appendix B provides a detailed description of how the augmented Reality capability is implemented.

7.5 Remote Imaging

A capability for simulating the physical process of imaging a surface using ultrasonic sensors has been implemented. The method uses a method known as *ray-casting* and makes use of graphical model representation of the equipment being inspected. An interface for ray-casting is built into the Coin3d graphics library.

The imaging algorithm is as follows. An object of type SoRayPickAction takes arguments of start position and ray direction, and returns the nearest point intersected by the ray. SoRayPickAction traverses the scene graph (or sub-scene graph) looking for transformation and geometry nodes. For each geometry node, all intersection points are calculated and are added to a list. This list is sorted by proximity, and the closest point is returned. This is analogous to an ultrasound ray hitting the nearest object.

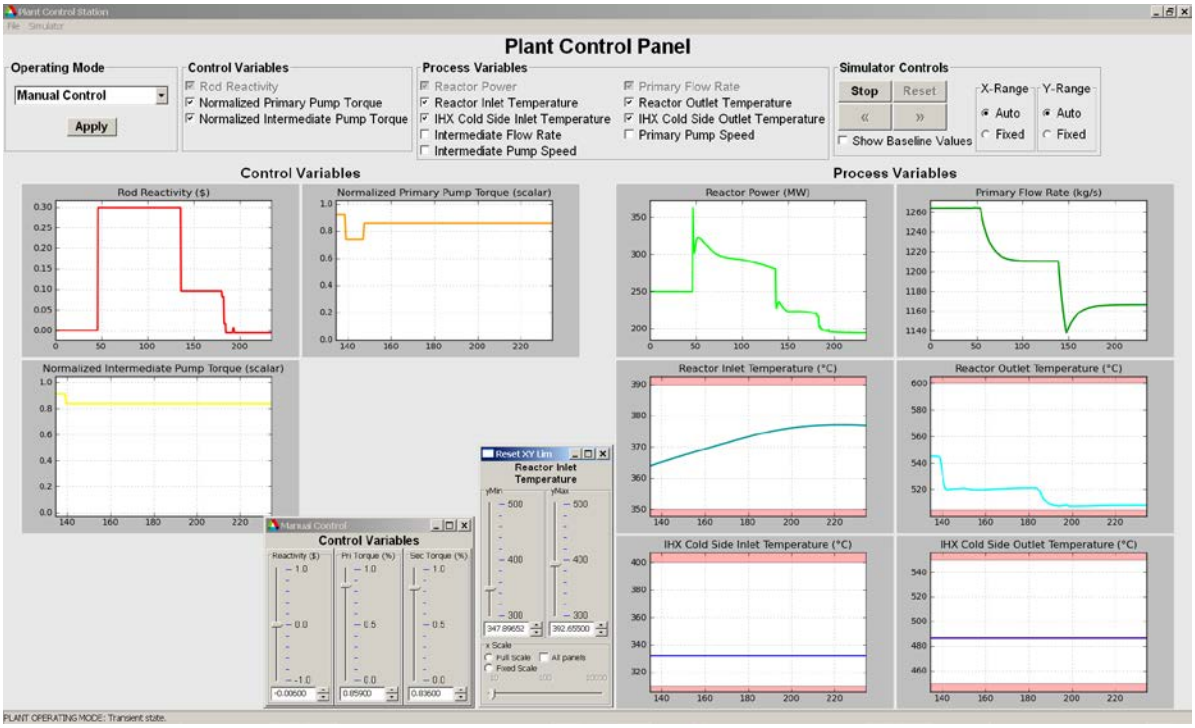


Figure 8.1 Plant Control Station Window

8 Current Status

8.1 At-Power Operation

The Plant Control Station shown in Figure 8.1 provides an interactive session for manually controlling plant actuators in a SASSYS-GPASS simulation synchronized to real time. The user can choose to manipulate rod reactivity, normalized primary pump torque, normalized intermediate pump torque, and generator demand power by adjusting a corresponding slider bar. The immediate effects of these changes and the response of the plant's safety systems can be observed in real time through plots of many different reactor variables, including: reactor power, primary and intermediate flow rates, primary and intermediate pump speeds, reactor inlet and outlet temperatures, and intermediate heat exchanger inlet and outlet temperatures.

The Real Time Status Station shown in Figure 8.2 displays the instantaneous values of plant process variables against a schematic of the plant. All of the major plant components are displayed on the screen. Temperatures of various components are updated as they are

calculated in real time. A blinking effect also highlights these reactor components in red when a relevant reactor variable is modified.

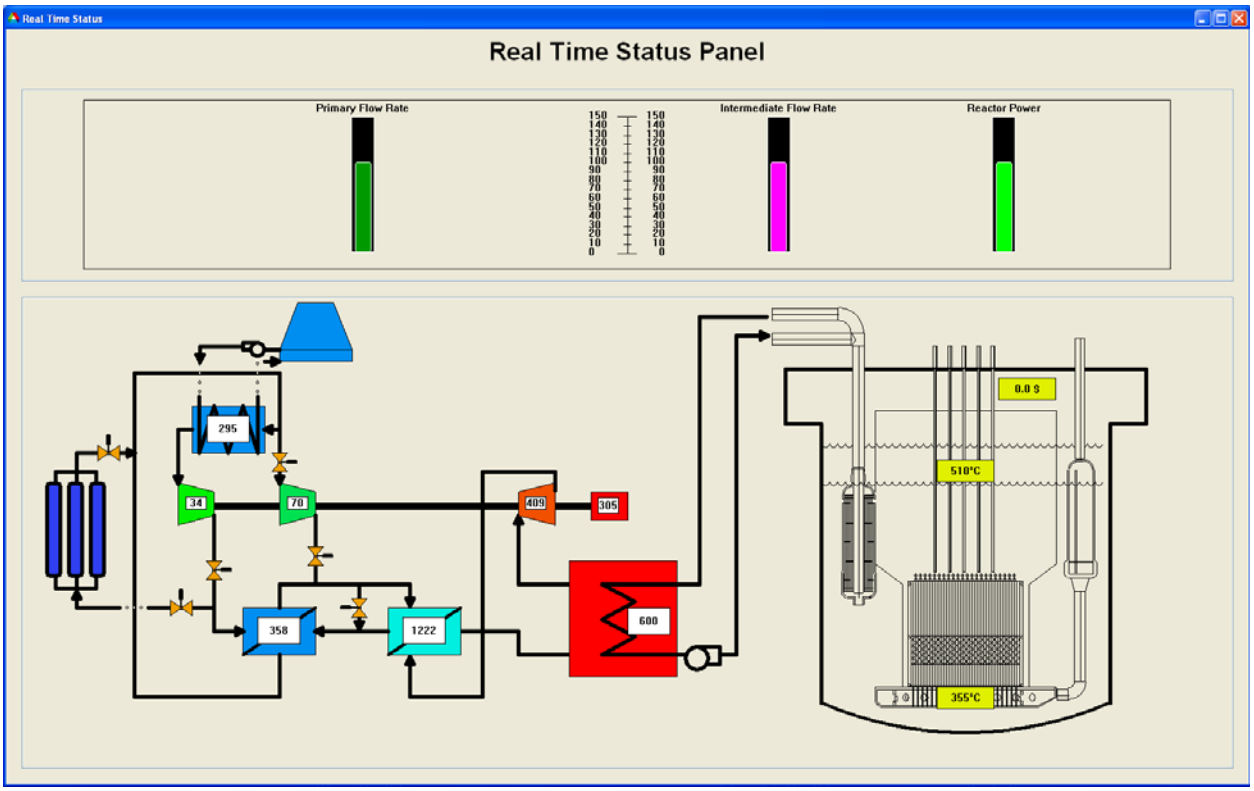


Figure 8.2 Real Time Status Station Window

8.2 Accident Response

The Safety Test Station shown in Figure 8.3, unlike the other two panels, does not run an interactive simulation. Instead, the user chooses from a set of common unprotected accidents: loss of flow, loss of heat sink, and a rod run out. The simulator takes several seconds to execute the accident simulation in batch mode, and then loads the data into a diagram of the reactor core regions color-coded by their temperatures. The simulation also produces a plot of the bulk reactor outlet temperature over time and a brief textual summary of the run data. The user can then replay the simulation at faster speeds or scroll through this data manually using a slider bar. This allows the user to examine the plant's behavior at any point in time during the simulation, with the display updating itself to reflect the data recorded at that point.

8.3 Refueling

Mechanical operation of the fuel handling machine is demonstrated for a part of the refueling process – fuel rod extraction. Key issues in the operation are trade-offs between precision, safety and efficiency. Four modes of refueling operations are demonstrated: automatic operation, manual operation, and guided operation. The interactive simulation will allow the operators to experience the trade-offs.

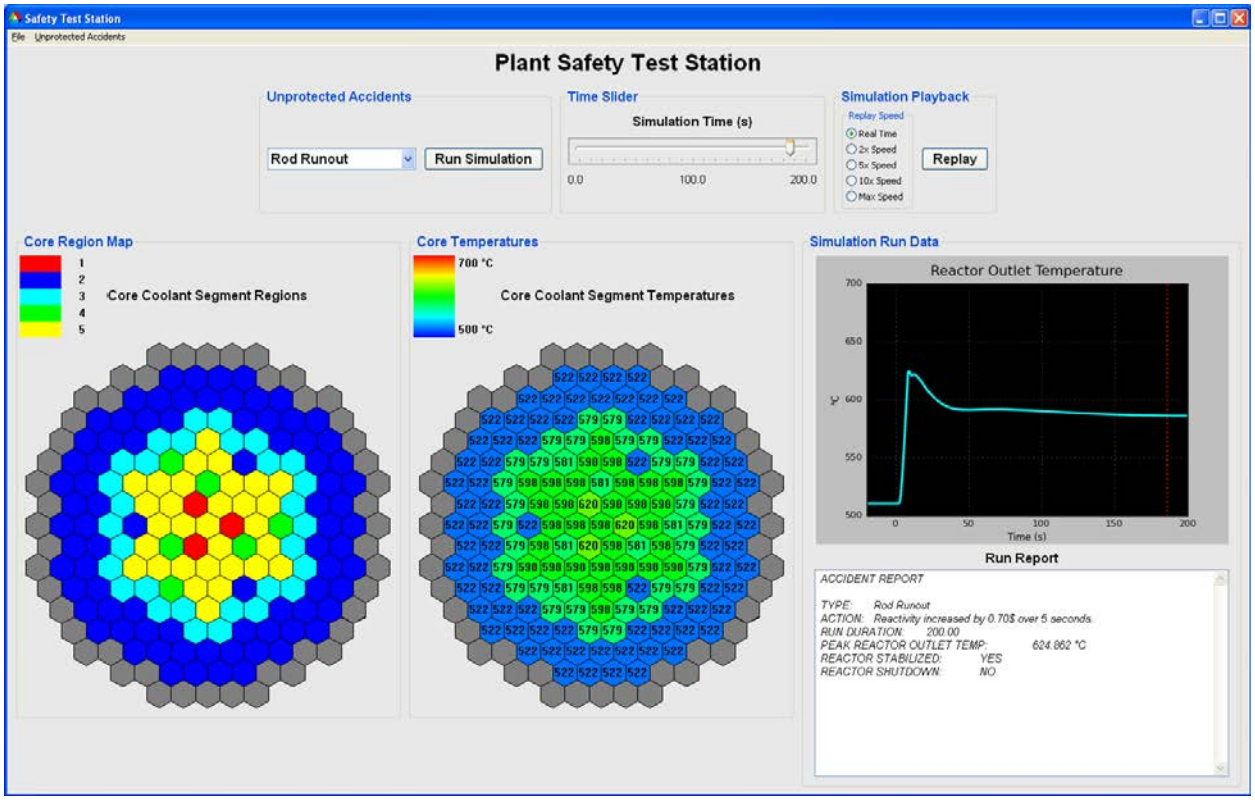


Figure 8.3 Safety Test Station Window

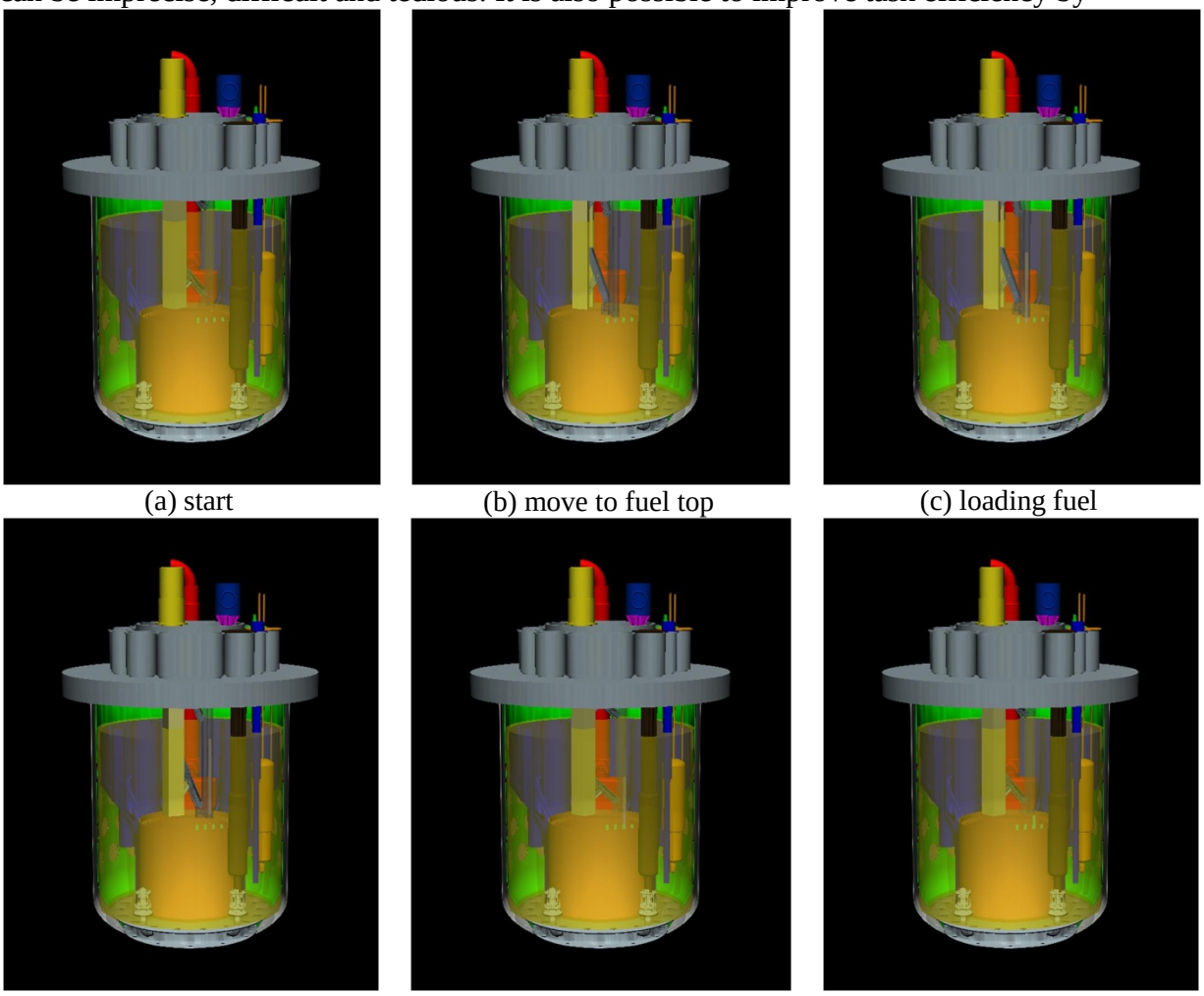
Automatic Operation - The fuel extraction process includes four steps: Move the rod to the place above the rod to be extracted, load the rod, move to the interim storage rack; unload the rod. In automatic operation mode, the operator triggers the process and the four step operations are sequenced based on interlocks. The automatic mode is triggered by keyboard event in keyboard_callback() function in myWindows.cpp file. Four Flags are set, and the autoFlag is used to enable the mode, and iteratively turn on the flag of the next phase when the previous procedure is properly finished.

Figure 8.4 presents a time sequence of screen captures recorded during execution of the automatic operation mode.

Automatic operation is the most efficient means for transferring fuel. However, it is prone through equipment failures to incidents of misalignment as reported in the literature.

Manual Operation - Manual operation is another possible mode where the operator controls every motion of the mechanism based on perceptual feedback. This mode of operation is demonstrated as shown in Figure 8.5 where the operator controls the graphic mechanism via a hand-controller. Both visual and haptic (sense of touch) feedback are provided. For safety critical operations and/or in unstructured environment, such a man-in-the-loop control

approach is preferred. In general, however, due to the imperfect perceptual feedback, this task can be imprecise, difficult and tedious. It is also possible to improve task efficiency by



(a) start (b) move to fuel top (c) loading fuel

(d) moving with fuel (e) unloading fuel (f) after

Figure 8.4 Time Sequence Showing Fuel Handling Machine in Automatic Operation Mode

incorporating semi-automatic operation which blends operator's gross motion and automated precision motion.

Guided Operation - In guided operation, virtual fixtures are introduced to assist the human operator to achieve safe and efficient operation. Guided operation is demonstrated for both unilateral and bilateral virtual fixtures.

The use of a unilateral virtual fixture is demonstrated for aiding in alignment of the fuel handling machine onto the fuel to be extracted. This is a tedious task in manual operation, yet full automation can result in misalignment. When invoked by the operator, cone shaped virtual surfaces are drawn from the current end-effector location to the target fuel top. By invoking the haptic mesh, this surface can serve as a unilateral virtual surface which the operator can use to

feel and follow to reach the alignment position. Figure 8.6 shows a snapshot of guided operations. Figure 8.7 shows the guided operation under various opacities of under-sodium environment. The benefit of the virtual fixture is more apparent when demonstrated in an opaque sodium environment.

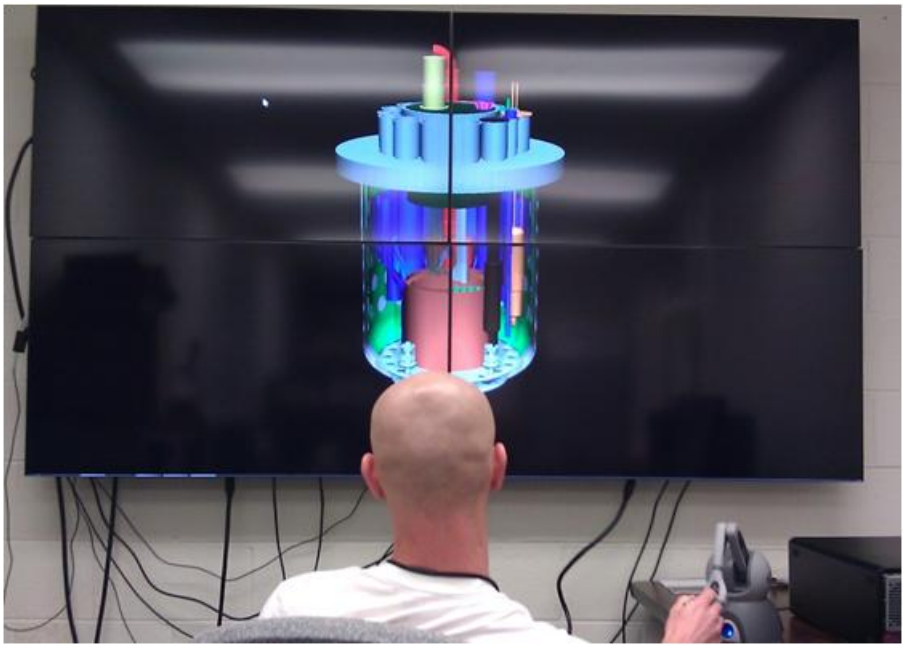


Figure 8.5 Fuel Handling Machine in Manual Control Mode

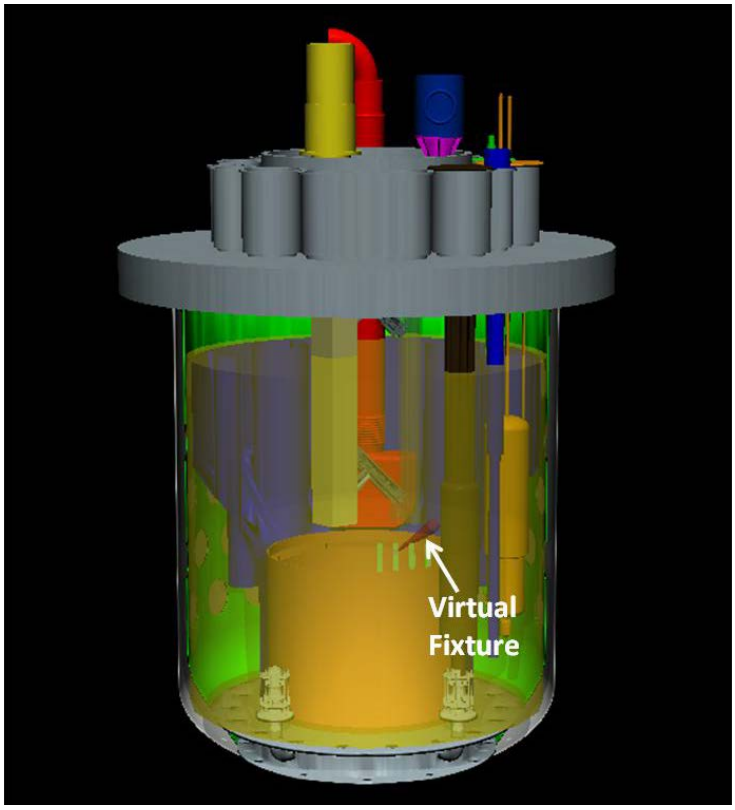


Figure 8.6 Fuel Handling Machine in Guided Operation Mode

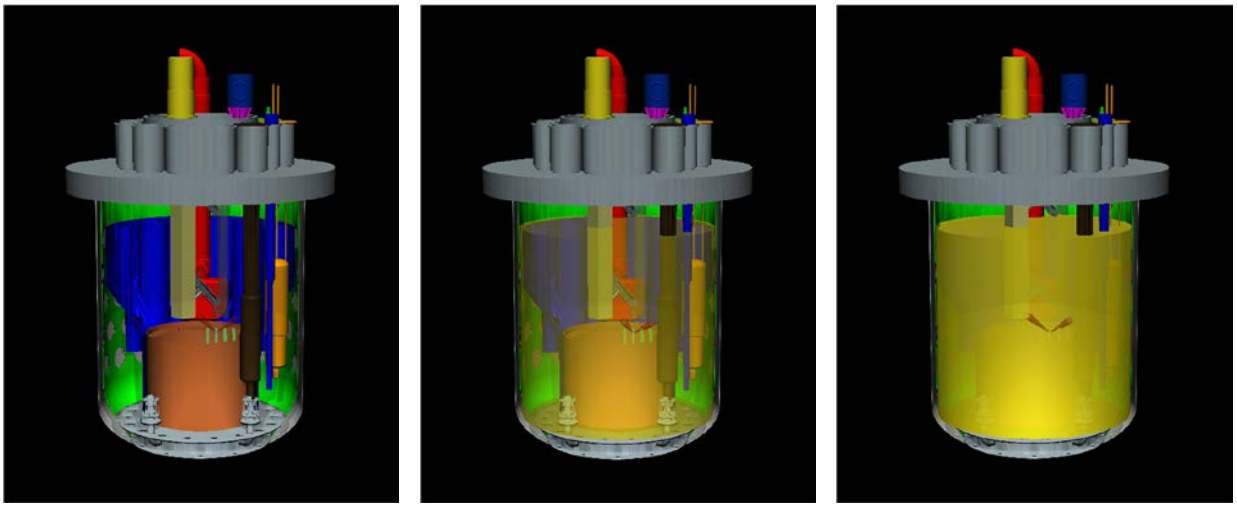


Figure 8.7 Guided Operation Mode Shown Under Three Different Sodium Opacities

The use of a bilateral virtual fixture is demonstrated for a fuel subassembly pulling operation. This operation is simulated and is performed with a virtual fuel handling machine having unconstrained end-effector vertical mobility. [A virtual fixture is a spatial construct across

whose boundary motion is prohibited or impeded. Appendix A presents a technical description of the virtual fixture concept.] Unlike the previous type of mechanism, this additional mobility makes it difficult to pull the rod in a vertical direction, or at regulated speed/force. A bilateral virtual fixture is invoked which constrains the end-effector motion in the vertical direction. The general theory of virtual fixture was used to design the guidance virtual fixture. The objective was to achieve a particular operation, the withdrawing of a fuel rod straight upward. This is shown in Figure 8.8 where the up direction is represented by $+y$. Given this coordinate system the associated value of the $\boldsymbol{D}$ matrix is $\boldsymbol{D} = [0\ 1\ 0\ 0\ 0\ 0]^T$. In this particular example the scaling of haptic device stylus movement to actual gripper movement was specified through the scalar $c = 40$ in the above equation. Additionally, to achieve a relatively hard fixture (i.e. to avoid damaging core structures), a large impedance to moving out of the preferred direction was achieved by specifying $c_\tau = 0.025$, a relatively small value for the attenuation factor in the above equation. This example illustrates some of the benefits of virtual fixtures. First, compared to manual teleoperation which is difficult to perform and results in imprecise motion, a virtual fixture aids in easy and precise operation. Second, compared to open-loop operation with the possibility of inadvertently forcing a non-compliant structure, it allows the user to sense the operation and to control it based on this feedback, making for safer safety-critical operations.

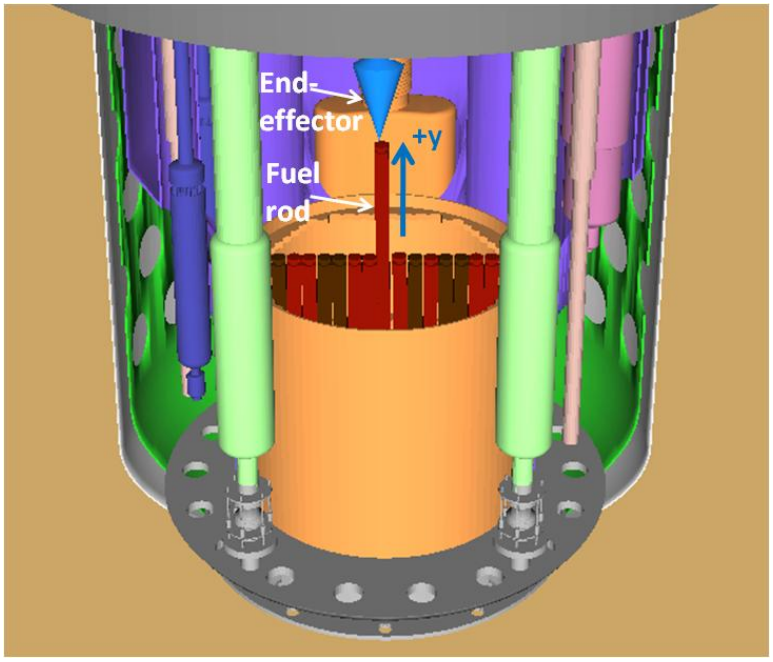


Figure 8.8 Snapshot of Fuel Subassembly Pulling Operation Guided by Virtual Fixture

8.4 Under-Sodium Viewing

The reactor simulator is capable of live ultrasound imaging. Automatic and manual modes are available. The automatic mode, activated by the 'z' key, casts rays as the robotic arm moves along a pre-defined path. The manual mode, activated by the 'x' key, casts rays from the current robotic arm position. Each ray takes about half a second, and the returned intersection points are written to a file to be plotted as a point cloud. Gnuplot is used for plotting. A Gnuplot script replots the point cloud data every second. This allows live data collection and visualization.

The result of scanning the IHX in the reactor graphics model in Figure 8.9 is shown in Figure 8.10. The image on the top of Figure 8.10 is the graphical representation of the component scanned. The image on the bottom is the scan result.

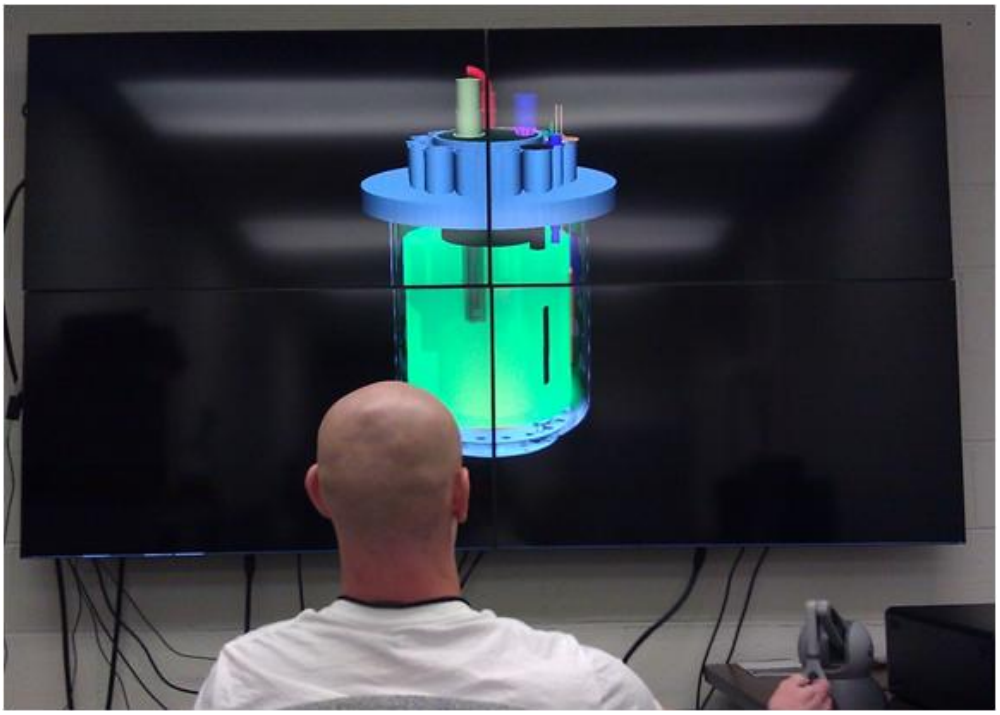


Figure 8.9 Photograph of Simulated Ultra Sonic Imaging of IHX in Progress

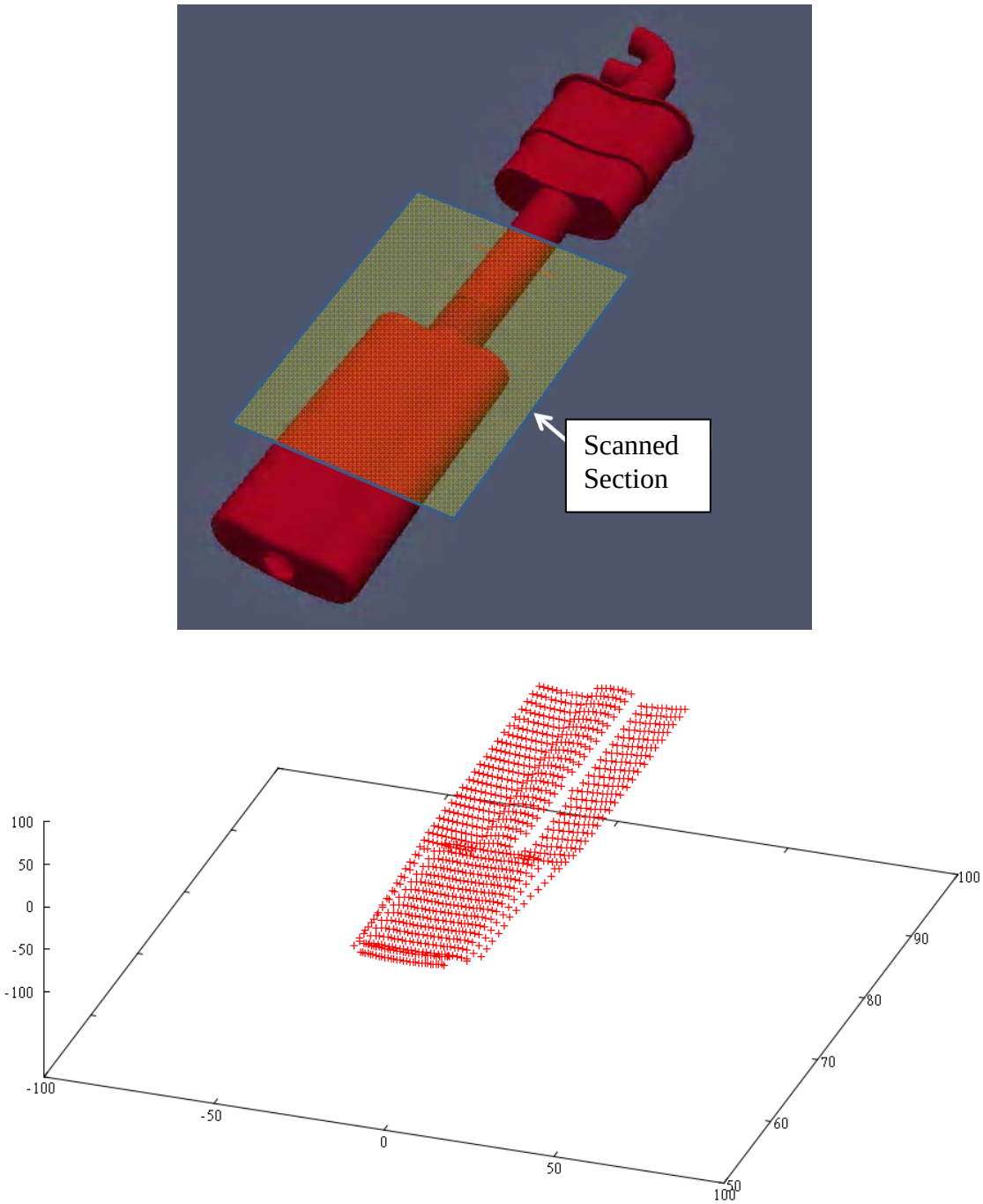


Figure 8.10 Graphics Model of IHX (top) and Simulated Imaging Result (bottom)

9 Conclusions

A simulator platform for visualization and demonstration of innovative concepts in fast reactor technology was described. The objective is to make more accessible the workings of fast reactor technology innovations and to do so in a human factors environment that uses state-of-

the art visualization technologies. In this work the computer codes in use at Argonne National Laboratory (ANL) for the design of fast reactor systems are being integrated to run on this platform. In its current state the SASSYS and GPASS codes have been linked for a whole plant simulation of a liquid metal reactor coupled to a supercritical carbon dioxide energy conversion cycle. The simulator also includes visualizations of mechanical systems related to fuel handling and in-service inspection.

10 References

1. R.B. Vilim, "Control and Safety System Design for Highly Autonomous Operation of a Modular Lead Reactor with Steam Cycle," *ICAPP 2004*, Pittsburgh, PA, June 2004.
2. Y. I. CHANG, P .J. FINCK, and C. GRANDY, "Advanced Burner Test Reactor Preconceptual Design Report," ANL-ABR-1, Argonne National Laboratory, September 2006.
3. V. DOSTAL, P. HEJZLAR, and M. J. DRISCOLL, "High Performance Supercritical Carbon Dioxide Cycle for Next Generation Nuclear Reactors", Nuclear Technology, Vol. 154, pp.265-282, June 2006.
4. R. B.VILIM, "One-Dimensional Compressor Model for Super-Critical Carbon Dioxide Applications," 2010 International Conference on Advances in Nuclear Power Plants (ICAPP'10), San Diego, CA, June 13-17, 2010.
5. D.A. CLAYTON and R. T. WOOD, "The Role of Instrumentation and Control Technology in Enabling Deployment of Small Modular Reactors," *Nuclear News* **54**(13), 42-47 (2011).
6. K. C. GROSS, R. M. SINGER, S. W. WEGERICH, and J. P. HERZOG "Application of a Model-Based Fault Detection System to Nuclear Plant Signals," Proc. 9th Intl. Conf. On Intelligent Systems Application to Power Systems, Seoul, South Korea, 1997.
7. J. REIFMAN and T.Y.C. WEI, "PRODIAG – Dynamic Qualitative Analysis for Process Fault Diagnosis," Proceedings of the 9th Power Plant Dynamics, Control and Testing Symposium, Knoxville, Tennessee, May 24-26, 1995.
8. 2010 ASME Boiler and Pressure Vessel Code, Section XI: Rules for Inservice Inspection of Nuclear Power Plant Components, Division 3: Rules for Inspection and Testing of Components of Liquid-Metal Cooled Plants, ASME, July 2010.
9. *OpenHaptics Toolkit version 3.0: Programmer's Guide*, SensAble Technology, Inc.
10. F. E. DUNN, F.G. PROHAMMER, D. P. WEBER, and R.B. VILIM, "The SASYS-1 LMFBR Systems Analysis Code," International Topical Meeting on Fast Reactor Safety, Knoxville, Tennessee, April 21-25, 1985.
11. R.B.Vilim and A. Moisseytsev, "Comparative Analysis of Supercritical CO2 Power Conversion System," 2008 International Conference on Advances in Nuclear Power Plants (ICAPP'08), Anaheim, CA, June 8-12, 2008.
12. S. Payandeh, and Z. Stanisic, "On Application of Virtual Fixtures as an Aid for Telemanipulation and Training," *Proc. Of 10th Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, IEEE, New York, pp. 18-23, 2002

13. H. Kang, Y. Park, et. al., "Visually and Haptically Augmented Teleoperation in D&D tasks Using Virtual Fixtures," *Proc. of ANS 10th International Topical Meeting on Robotics and Remote Systems*, Gainsville, FL, March 28 – 31, 2004

14. P. Marayong, M. Li, A. Okamura, G. Hager, "Spatial Motion Constraints: Theory and Demonstrations for Robot Guidance using Virtual Fixtures," *Proc. ICRA*, 2003, pp. 1954-1959

Appendix A Haptic and Visual Virtual Fixtures Concepts

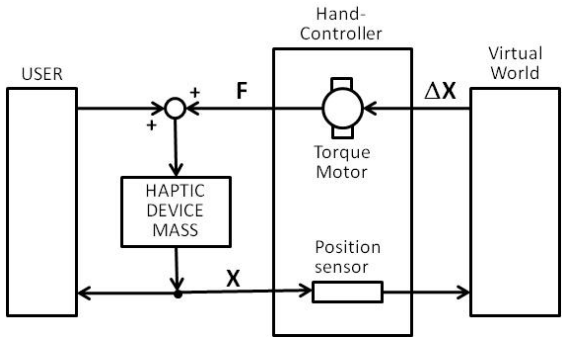
A.1 Haptic Virtual Fixture

The Phantom haptic device is an impedance controlled device (as opposed to admittance device), whereas mechanical impedance, $Z(\omega)$, is defined as

$$\mathbf{f}(\omega) = Z(\omega)\,\mathbf{v}(\omega) \qquad (1)$$

where is $\mathbf{f}(\omega)$ and $\mathbf{v}(\omega)$ are force and velocity in complex frequency domain, ω. In impedance control, the paradigm is: the user moves the device, and the device will react with a force. This is the basic interaction between the user and the control loop. Viewed from the control loop, the paradigm is: "displacement in – force out". Fig. A.1 shows the schematics of an impedance controlled device in interaction with the user and the (virtual) environment.

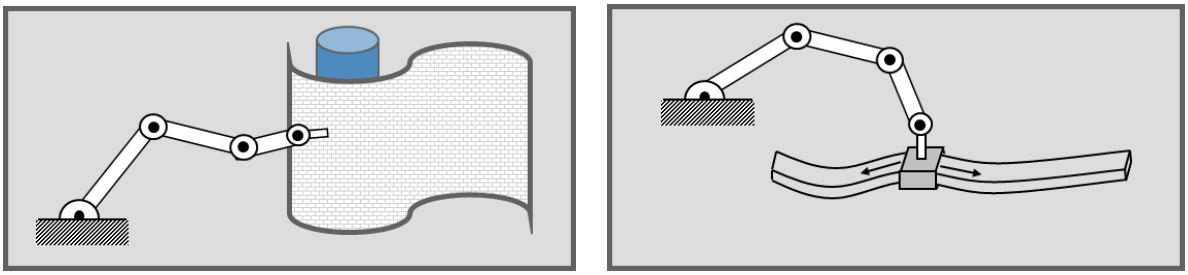
(a) Phantom Omni Haptic Device device



(b) Schematic of an impedance controlled

Figure A.1. Schematics of Impedance Controlled Haptic Device (Phantom Omni)

A unilateral virtual fixture is represented as an inequality constraint equation. It can be thought of as a virtual wall to confine the free motion space. The motion is free until it interferes with the constraint geometry. Examples of its use are defining of a forbidden region [12], or an alignment cone [13]. A bilateral virtual fixture has a bilateral constraint that is expressed with equality. The end-effector motion is forced to lie in the constraint geometry.



(a) Unilateral (b) Bilateral

Figure A.2. Illustration of Virtual Fixtures

The following simplified example illustrates how a control law that implements a virtual fixture can be derived [14]. Suppose the robot's end-effector pose is $\mathbf{x} = [\mathbf{p}, \mathbf{r}]^T$, which consists of the end-effector position $\mathbf{p} = [x,y,z]^T$ and the end-effector orientation, $\mathbf{r} = [r_x,r_y,r_z]^T$. The control input to the robot is assumed to be the desired end-effector velocity, $\mathbf{v} = \dot{\mathbf{x}} = \mathbf{[\dot{p},\dot{r}]}$. In teleoperation, the control input is derived from the operator commanded velocity input, $\mathbf{v_{op}}$, such that

$$\mathbf{v} = \boldsymbol{C}\,\mathbf{v_{op}} \qquad (2)$$

where $\boldsymbol{C}$ is a diagonal matrix defining motion constraint. To illustrate a virtual fixture, all but the first two diagonal elements equal to zero creates a system that permits translational motion in the x-y plane only. This is an example of a simple *guidance virtual fixture* whereby the elements of $\boldsymbol{C}$ serve to define the degree of "hardness" of a constraint whose purpose is to limit motion. That is a 'soft' rather than 'hard' virtual fixture can be defined by allowing small compliance in the non-preferred direction. In the example above this could be achieved by setting the third and higher diagonal elements of $\boldsymbol{C}$ to small non-zero values.

The above example served to illustrate relevant aspects of virtual fixtures which are part of a more general theory and treatment described next. In the more general framework the matrix $\boldsymbol{C}$ becomes the more generalized time-varying geometric constraint matrix of dimension 6 x n represented by $\boldsymbol{D} = \boldsymbol{D}(t)$, $0<n<6$ where t is time and n is degrees-of-freedom (d.o.f.) space. By specifying the $\boldsymbol{D}$ in such a way as to constrain the mobility in an n d.o.f. space, a virtual fixture is again defined. The key concept of bilateral virtual fixture is that it is possible to choose the generalized coordinate vector in such a way to decompose the motion into one in constrained space and the other in free space. From $\boldsymbol{D}$, a projection matrix $Span(\boldsymbol{D}) \equiv [\boldsymbol{D}] = \boldsymbol{D}(\boldsymbol{D}^T\boldsymbol{D})^+\boldsymbol{D}^T$ can be defined that projects any $\mathbf{v_{op}}$ onto the column space of $\boldsymbol{D}$. And another projection matrix $Ker(\boldsymbol{D}) = \langle\boldsymbol{D}\rangle = \mathbf{I} - [\boldsymbol{D}]$ is defined which projects any $\mathbf{v_{op}}$ onto the orthogonal complement null space. By defining these projection matrixes, the master motion is easily decomposed into one in preferred directions and the other in non-preferred directions. One sees this in the following representation where the operator command motion, $\mathbf{v_{op}}$, is decomposed into two components, an allowable motion region and a region that is to a degree forbidden. The control law that achieves this is

$$\mathbf{v} = \mathbf{c}([\boldsymbol{D}] + \mathbf{c_\tau}\langle\boldsymbol{D}\rangle)\,\mathbf{v_{op}}$$

where $\mathbf{c}$ is an overall motion scaling factor and $\mathbf{c_\tau}$ is an additional scaling factor that attenuates the non-preferred component of the operator motion command. As noted above, we refer to the case $\mathbf{c_\tau} = 0$ as "hard virtual fixture", and $\mathbf{c_\tau} \neq 0$ (i.e. small non-zero value) as "soft virtual fixture".

A.2 Visual Virtual Fixture

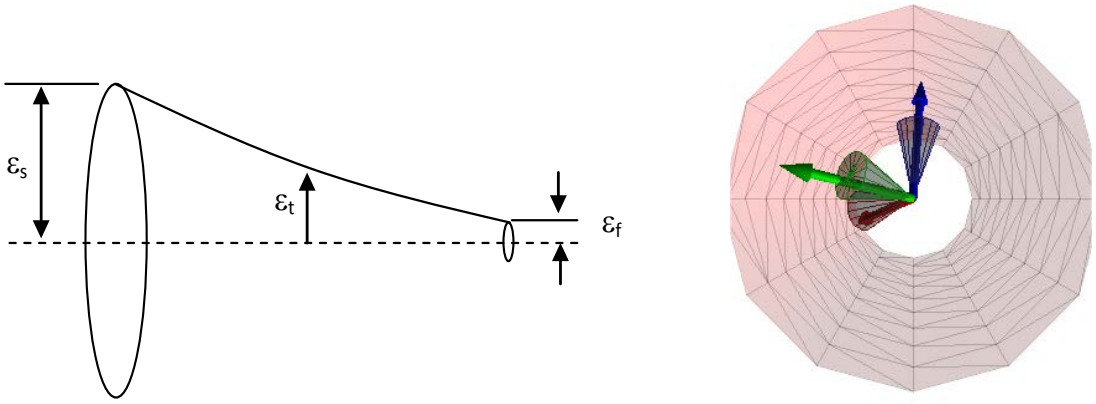
Besides the haptic virtual fixture described above, other sensory forms of virtual fixtures are also possible such as visual, auditory and tactile. The *visual* virtual fixture is described in this section. Visual virtual fixture is by nature unilateral and can be used to define a forbidden region or to guide motion. In this work, we have adopted virtual fixtures to guide alignment of

the end-effector onto the work piece, e.g. onto the fuel pin for grasping. Such a task involves alignments of both position and orientation. We adopted as appears in Fig. A.3 a virtual funnel for positional alignment and virtual aligning cones for rotational alignment [13].

Virtual funnel is defined with radius ε_s at the start position $\mathbf{p}_s$, ε_f at the end position $\mathbf{p}_f$, and the funnel axis $\mathbf{n}_t = \mathbf{p}_f - \mathbf{p}_s$. As shown in Fig. A.3 (a), the radius of the funnel along the axis is described by a cubic parametric function of $t \in [0,1]$ as

$$\varepsilon_t(t) = \varepsilon_f + (\varepsilon_s - \varepsilon_f)(2t^3 - 3t^2 + 1).$$

On approaching the target position, the virtual funnel shrinks so that the operator can easily guide the position of tool.



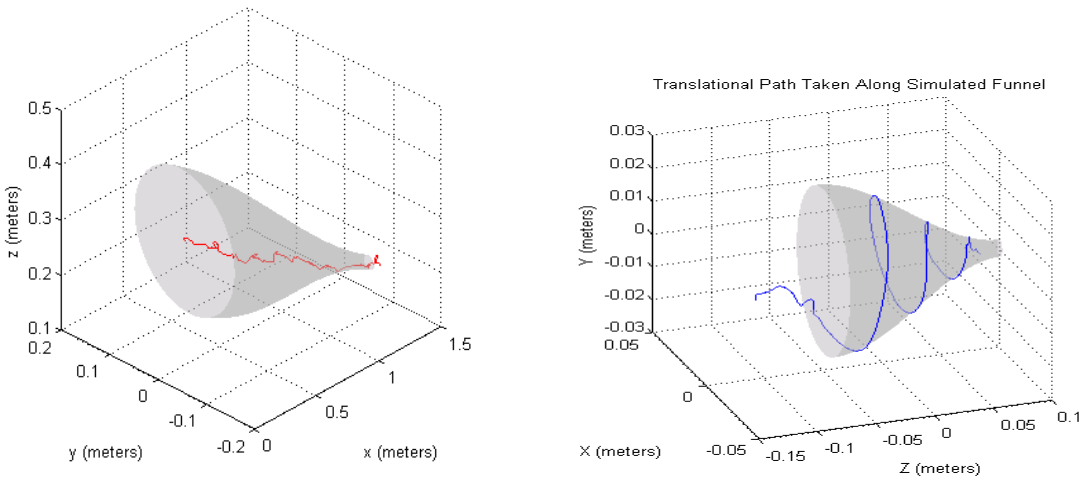
(a) Virtual funnel for positional alignment alignment

(b) Virtual cones for orientation

Figure A.3. Visual Virtual Fixtures

In addition virtual aligning cones shown in Fig. A.3 assists the user to align the tool orientation. Fig. A.3 (b) shows virtual cones with axes in the three orthonormal directions of the goal orientation. The virtual cone angle allows the operator to get intuitive perception of the desired rotational motion. The geometrical model of the virtual aligning cones can be concisely represented with the introducing quaternion notations and their spherical linear interpolation. [13]

Fig. A.4 illustrates examples of the progression of alignment tasks guided by visual virtual fixtures. Fig. A.4 (a) shows the translational path taken along the virtual funnel in an actual test. The results show that both translational and rotational alignment was achieved successfully. Fig. A.4 (b) shows the result of an alignment task guided by combined visual and haptic virtual fixtures. The tool is first visually guided by the unilateral virtual fixture until it hits the virtual surface. Once the operator hits the virtual surface, he is aided by the bilateral virtual surface and is able to slide into the target position along the surface.



(a) guided by visual virtual fixture (b) visually and haptically guided

Figure A.4. Result of alignment tasks guided by virtual fixtures

Nuclear Engineering Division
Argonne National Laboratory
9700 South Case Avenue
Argonne, IL 60439

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC