INL/EXT-15-36749

Light Water Reactor Sustainability Program

Progress on the Industry Application External Hazard Analyses Early Demonstration

Curtis L Smith, Steven Prescott, Justin Coleman, Emerald Ryan, Bishwo Bhandari, Daniel Sludern, Chad Pope, Ram Sampath

September 2015

DOE Office of Nuclear Energy

INL/EXT-15-36749

Light Water Reactor Sustainability Program

Progress on the Industry Application External Hazard Analyses Early Demonstration

Curtis L Smith – INL
Steven Prescott – INL
Justin Coleman – INL
Emerald Ryan – Idaho State University
Bishwo Bhandari – Idaho State University
Daniel Sludern – Idaho State University
Chad Pope – Idaho State University
Ram Sampath – Centroid PIC

September 2015

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov/lwrs

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

This report describes the current progress and status related to the Industry Application #2 focusing on External Hazards. For this industry application within the Light Water Reactor Sustainability (LWRS) Program Risk-Informed Safety Margin Characterization (RISMC) Pathway, we focus on a Risk-Informed Margin Management approach to provide event scenarios and consequences by using an advanced 3D facility representation that evaluates external hazards. We evaluate hazards such as flooding and earthquakes in order to:

- Identify, model and analyze the appropriate physics that needs to be included to determine plant vulnerabilities related to external events.
- Manage the communication and interactions between different physics modeling and analysis technologies.
- Develop the computational infrastructure through tools related to plant representation, scenario depiction, and physics prediction.

One of the unique aspects of the RISMC approach is how it couples probabilistic approaches (the scenario) with mechanistic phenomena representation (the physics) through simulation. This simulation-based modeling allows decision makers to focus on a variety of safety, performance, or economic metrics. In this report, we describe the evaluation of various physics toolkits related to flooding representation. Ultimately, we will integrate the flooding representation with other events such as earthquakes in order to provide coupled physics analysis for scenarios where such interactions exist.

CONTENTS

FIGURES

TABLES

ACRONYMS

AEC	Atomic Energy Commission
ASCE	American Society of Civil Engineers
ASCII	American Standard Code for Information Interchange
ASPRA	Advanced Seismic Probabilistic Risk Assessment
CDF	Core Damage Frequency
CFD	Computational Fluid Dynamics
CFEL	Component Flooding Evaluation Laboratory
CFR	Code of Federal Regulation
CPU	Central Processing Unit
CSV	Comma Separated Values file
CUDA	Compute Unified Device Architecture
DAT	Data file
DBE	Design Basis Earthquake
DPD	Dissipative Particle Dynamics
DPI	DualSPHysics Pre-processing Interface
DOE	Department of Energy
EPRI	Electric Power Research Institute
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IA	Industry Application
IDE	Integrated Development Environment
INL	Idaho National Laboratory
ISRS	In-Structure Response Spectrum
ISU	Idaho State University
LWRS	Light Water Reactor Sustainability
MD	Molecular Dynamics
MOOSE	Multiphysics Object Oriented Simulation Environment
NLSSI	Nonlinear Soil-Structure Interaction
NPP	Nuclear Power Plant
NRC	Nuclear Regulatory Commission
NTTF	Near-Term Task Force
NUREG	Nuclear Regulatory Report

PFHA	Probabilistic Flood Hazard Assessment
PGA	Peak Ground Acceleration
PLY	Polygon file format
PNG	Portable Network Graphics file
PRA	Probabilistic Risk Assessment
PSHA	Probabilistic Seismic Hazard Assessment
RAVEN	Risk Analysis and Virtual Control Environment
R&D	Research and Development
RG	Regulatory Guide
RIMM	Risk-Informed Margin Management
RISMC	Risk-Informed Safety Margin Characterization
RPM	RPM Package Manager
SASSI	System for Analysis of Soil Structure Interaction
SCDF	Seismic Core Damage Frequency
SDK	Software Development Kit
SPH	Smooth Particle Hydrodynamics
SPID	Screening, Prioritization and Implementation Details
SPRA	Seismic Probabilistic Risk Assessment
SSCs	Structures, Systems, and Components
SSE	Safety Shutdown Earthquake
SPH	Smoothed Particle Hydrodynamics
SSI	Soil-Structure Interaction
STL	Stereolithography file
TXT	Text file
VTK	Visualization Toolkit file
VTU	Visual Toolkit for Unstructured grids file
XML	Extensible Markup Language

Progress on the Industry Application External Hazard Analyses Early Demonstration

1. INTRODUCTION

1.1 Background

Design of nuclear power plant (NPP) facilities to resist external hazards has been a part of the regulatory process since the beginning of the NPP industry in the United States (US), but has evolved substantially over time. The original set of approaches and methods were entirely deterministic in nature and focused on a traditional engineering margins-based approach. In this approach, design is undertaken for each structure, system, and component (SSC) individually based on achieving a capacity that is expected to provide a minimum margin over some specific design load of interest. However, neither the risk significance of the SSC nor its role within the facility is considered. The traditional approach also does not account for operator action, redundancy and other risk-related element.

Over time probabilistic and risk-informed approaches were also developed and implemented in US Nuclear Regulatory Commission (NRC) guidance and regulation. A defense-in-depth framework was also incorporated into US regulatory guidance. As a result, the US regulatory framework incorporates deterministic and probabilistic approaches for a range of different applications and for a range of natural hazard considerations. This framework will continue to evolve as a result of improved knowledge and newly identified regulatory needs and objectives, most notably in NRC activities initiated in response to the 2011 Fukushima event in Japan.

Although the US regulatory framework has continued to evolve over time, the tools, methods and data available to the US nuclear industry to meet the changing requirements have largely remained static. Notably, there is room for improvement in the tools and methods available for external event probabilistic risk assessment (PRA), which is the principal assessment approach used in risk-informed regulations and risk-informed decision-making. Development of a new set of tools and methods that incorporate current knowledge, modern best practice, and state-of-the-art computational resources would lead to more reliable assessment of facility risk and risk insights (e.g., the SSCs and accident sequences that are most risk-significant), with less uncertainty, and reduced conservatisms.

For the evaluation of industry applications within the Light Water Reactor Sustainability (LWRS) Program Risk-Informed Safety Margin Characterization (RISMC) R&D Pathway, we will create the RIMM approach to represent meaningful (i.e., realistic facility representation) event scenarios and consequences by using an advanced 3D facility representation that will:

- Identify, model and analyze the appropriate physics that needs to be included to determine plant vulnerabilities related to external events.
- Manage the communication and interactions between different physics modeling and analysis technologies.
- Develop the computational infrastructure through tools related to plant representation, scenario depiction, and physics prediction.

External hazards of interest have a primary impact on the nuclear facility that may also lead to secondary phenomena. Examples of external hazards that cause a primary impact are seismic shaking, flooding, and high winds. Examples of secondary phenomena induced by a seismic scenario are dam and levy failure, landslide, internal flood, and internal fire.

A notional depiction of this 3D representation approach is shown in Figure 1. As shown in this figure, we "layer" the different analyses that play a role in a particular scenario.

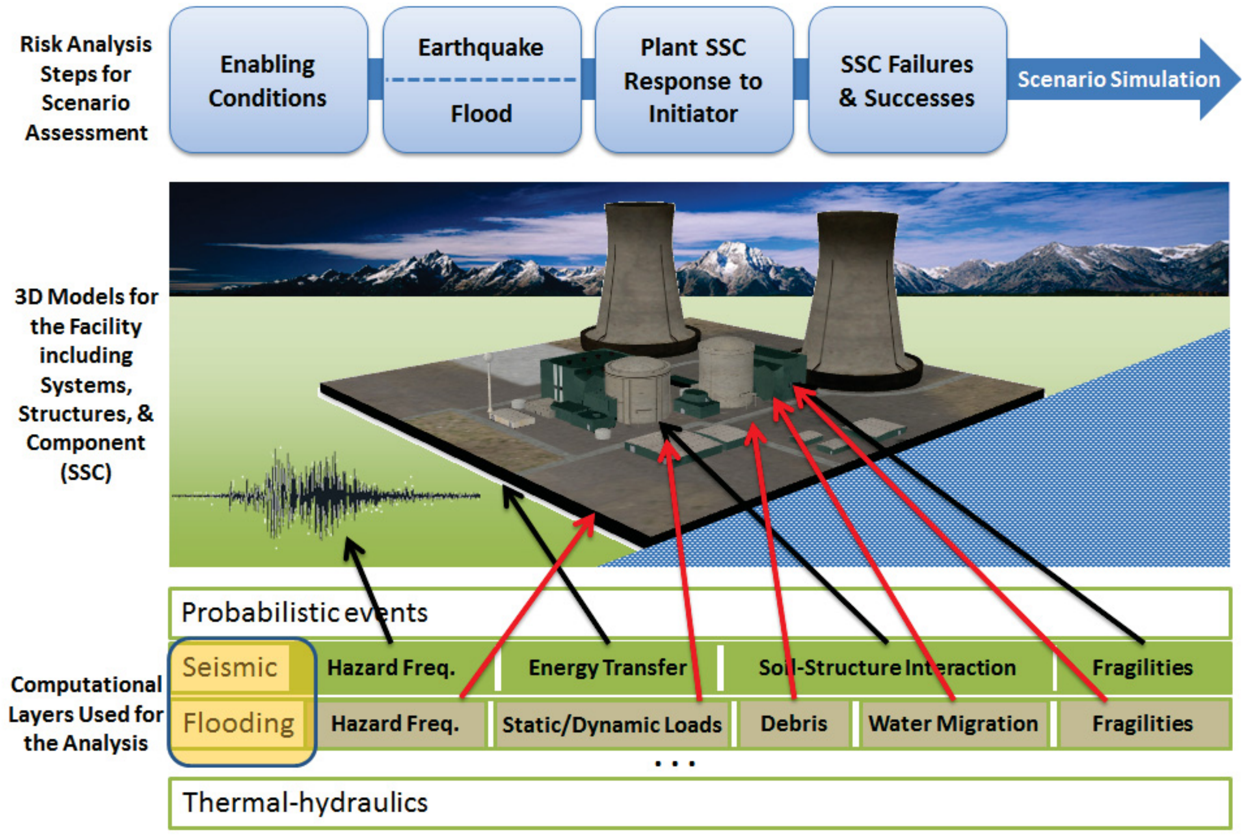


Figure 1. High-Level Features of the External Events Analysis Approach.

In order to enable probabilistic aspects of NPP external events modeling, we are using event simulation as the quantification method. Successfully linking probabilistic simulation to external events physics is a key facet of advanced methods and will directly address problems such as highly time-dependent flooding scenarios.

One of the unique aspects of the RISMC approach is how it couples probabilistic approaches (the scenario) with mechanistic phenomena representation (the physics) through simulation. This simulation-based modeling allows decision makers to focus on a variety safety, performance, or economic metrics. For example, while traditional risk assessment approaches for external hazards attempt to quantify core damage frequency (CDF), RIMM approaches may instead wish to consider other metrics such as:

• Magnitude of the hazard – for example, the height of water on buildings, or the height of water inside strategic rooms. The “magnitude” might be measured (during the simulation) by metrics such as water height, seismic energy, water volume, water pressure, etc.

• Damage to the plant (but not core damage) – for example, we may be interested in scenarios in which the facility does not see core damage, but would still experience extensive (or even minor) damage. The “damage” might be measured (again during the simulation) by metrics such as total number of components failed, cost of components destroyed, structures rendered unusable, the length of time the facility is impacted (ranging from hours to months), etc.

The defining difference between these new RIMM metrics and traditional ones such as CDF is that they represent observable quantities (e.g., the number of components failed, the costs related to the event, the height of water in a room, the duration of the event) rather than just the statistics of an event frequency. We believe these new metrics that are provided by the RISMC simulation will yield enhanced decision-making capabilities for nuclear power plants.

1.2 RIMM Industry Applications

Advanced safety analysis focuses on modernization of nuclear power safety approaches using verified and validated methods and tools; implementing state-of-the-art modeling techniques; taking advantage of modern computing hardware; and combining probabilistic and mechanistic analyses to enable a risk-informed safety analysis process. The modernized tools will maintain the current high level of safety in our nuclear power plant fleet, while providing an improved understanding of safety margins and the critical parameters that affect them. Thus, the set of tools will provide information to inform decisions on plant modifications, refurbishments, and surveillance programs, while improving economics.

Risk-informed approaches provide a technical basis for understanding and managing hazards (i.e., safety risks). In addition, risk-informed approaches can be used to estimate costs (i.e., economic risks) to support safety decisions. While the focus of advanced safety analysis is on “facility” safety, it should be noted that these facilities are managed by diverse organizations (i.e., the nuclear industry, the Department of Energy (DOE), and associated oversight organizations).

The primary purpose of industry applications in RISMC is to demonstrate advanced risk-informed decision making capabilities in relevant, realistic industry applications. The end goal of these activities is the full adoption of the RISMC tools by industry applied to their decision making process.

The four elements of the industry applications are further explored below:

(a) Demonstrate

- Provide confidence and a technical maturity in the RISMC methodology (essential for broad industry adoption)
- Strong stakeholder interaction required
- Address a wide range of current relevant issues (see also item (d))
- Three phase approach
    (1) Problem definition (3-6 months)
    (2) Early Demonstration (eDemo) (limited scope) (6-12 months)
    (3) Complete Application and Validation (Long Term- Methods, Tools, Data) (1-5 years)

(b) Advanced

- Analyze multi-physics, multi-scale, complex systems
- Use of a modern computational framework
- A variety of Methods, Tools, and Data can be utilized (e.g. use of legacy tools and state-of-the-art tools)
- Be as realistic as practicable (with the use of appropriate supporting data)
- Consider uncertainties appropriately and reduce unnecessary conservatism when warranted

(c) Risk-Informed decision making capabilities

- Use of an integrated decision process
- Integrated consideration of both risks and deterministic elements of safety

(d) Relevant industry applications

- The industry application of focus in this report is IA2 – Enhanced External Hazard Analyses (multi-hazard)

2. EXTERNAL EVENTS ANALYSIS

The early demonstration that IA2 will solve includes two external hazards, seismic and flood. The flooding at the NPP may be caused by either seismically-induced failure of an adjacent levy and seismically-induce internal flooding as a result of pipe breaks within the NPP. This report focuses primarily on the status of the flooding simulation technology – as the IA2 proceeds other hazards will be included. Figure 2 visually shows the problem definition.

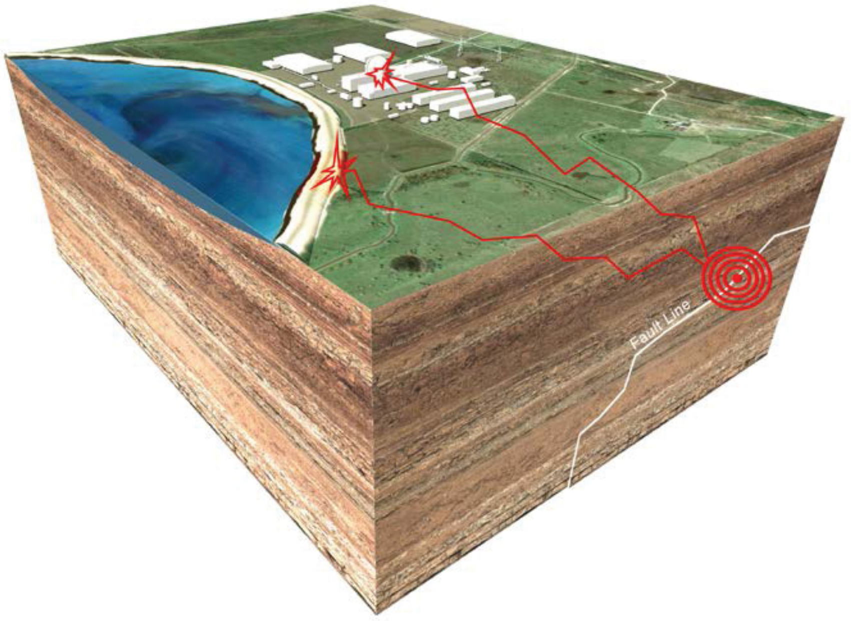


Figure 2. Illustration of the Industry Application External Hazard Analyses Problem Scope.

2.1.1 Seismic Analysis

Nonlinear soil-structure interaction (NLSSI) seismic analysis will be run to determine NPP response during multiple earthquake scenarios. NLSSI will also be performed to calculate dynamic response of the levy. Ground motion input for the NLSSI analysis will be developed from site-specific seismic hazard curves. Hundreds of scenarios, fit to the seismic hazard curve, will be run to determine probability of failure of internal safety class systems and the levy. These probabilities of failure of piping systems and the levy will then drive the assessment of the impact resulting from these secondary flooding phenomena.

An example of a recent NLSSI analysis shows that the generic NPP (Figure 3) in-structure response is different when calculated using linear and nonlinear SSI codes – the curves in Figures 4 and 5 show a comparison of linear and nonlinear SSI calculations at two different locations in the generic NPP (locations are identified in Figure 3). The results show a reasonable match at low levels of ground motion as expected since at low levels of ground motion the coupled soil structure response is linear. The curves show increasing divergence at high levels of ground motion. These plots show the maximum acceleration values on the response spectrum versus the applicable multiple of the site specific Design Basis Earthquake (DBE) (i.e. 0.5, 1,1.5, 2, 3). These figures show a nonlinear effect that is mainly produced by the ability to model gaping and sliding between the soil and structure. The observation here is that as the seismic hazard increases in magnitude, the larger the potential conservatism we see in the NLSSI portion of the seismic scenario modeling.

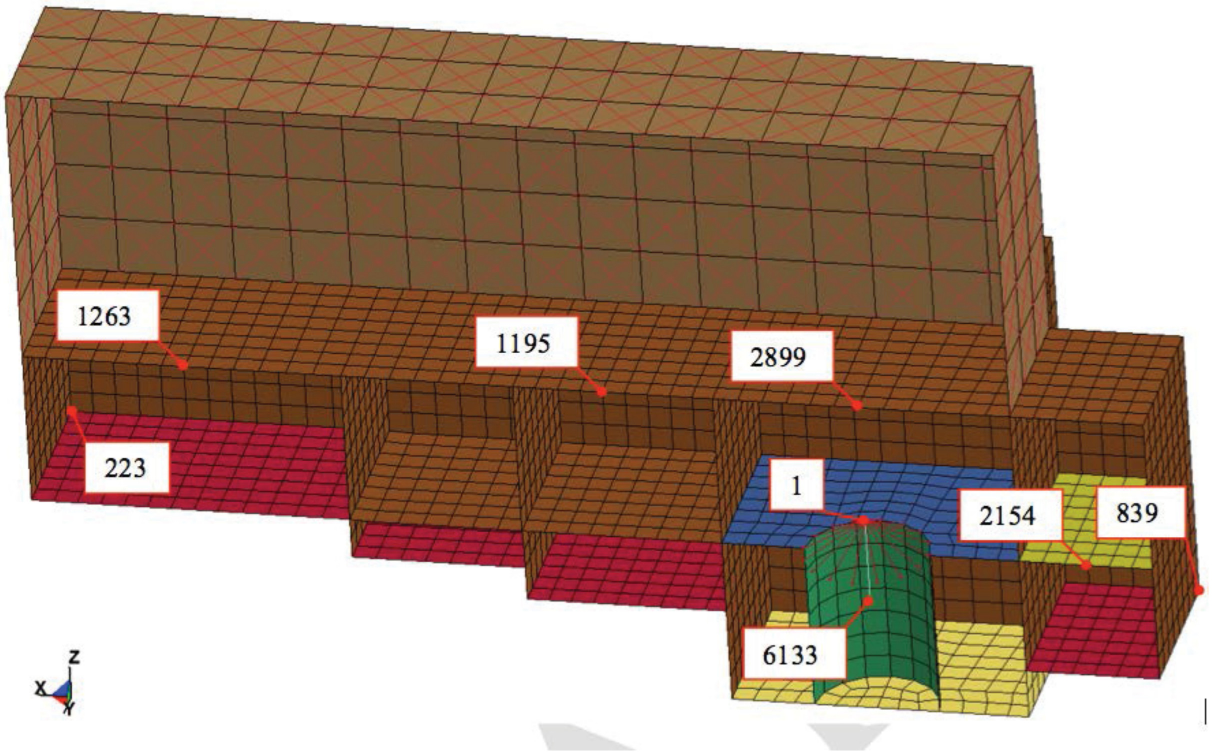


Figure 3. Structural Model of a Generic NPP.

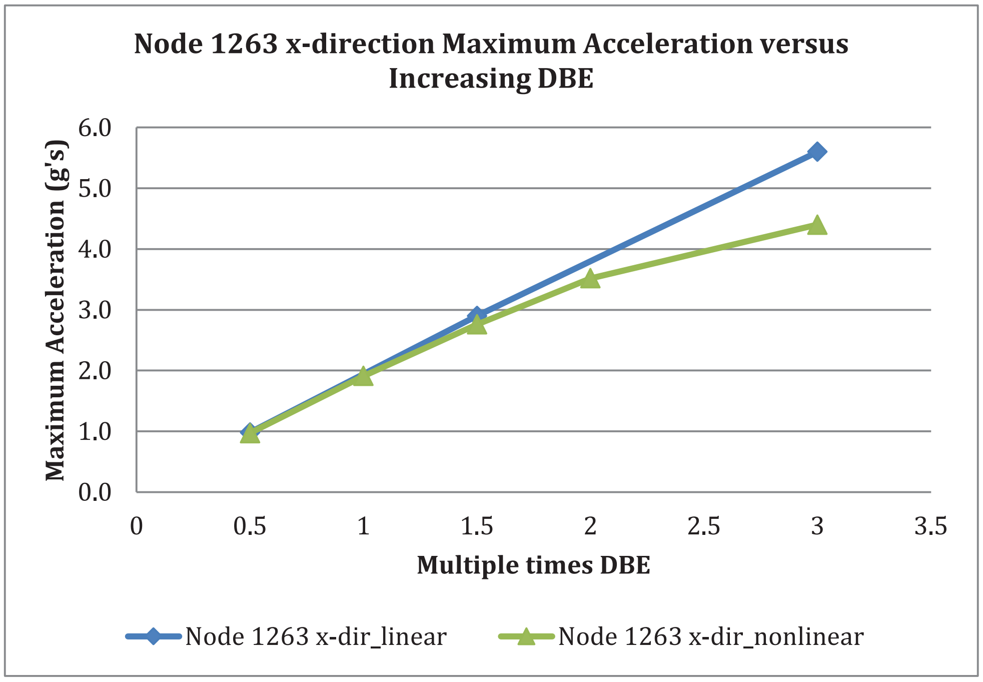


Figure 4. Maximum Response Spectrum Acceleration at Increasing Levels of Ground Motion at INL Site at In-Structure Location, Node 1263.

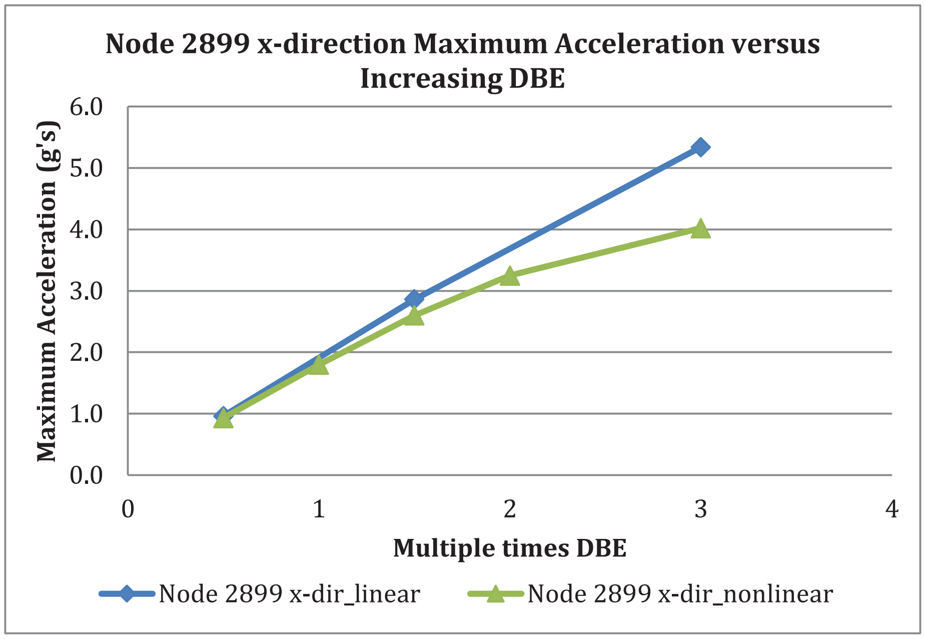


Figure 5. Maximum Response Spectrum Acceleration at Increasing Levels of Ground Motion at INL Site at In-Structure Location, Node 2899.

2.1.2 3D Plant and Site Representation

When simulating accident scenarios as part of RISMC, we may require multiple physics-based modules that must be run for one or more scenarios directly as part of the analysis. A subset of these simulation modules might be run "offline" and their results stored in whatever format is native to that particular application for retrieval during the analysis. Alternatively, we may be able to translate these mechanistic calculations into what are called “emulators” (or reduced-order models) wherein the emulator mimics the more complicated analysis but is able to run orders of magnitude faster. Let us describe a possible approach that would be used for a realistic plant representation to better understand how physics-based simulation is used in an IA.

First, we need to construct a model representing the topography of the site (and surrounding areas) and various structures at the NPP. An example of this 3D model is shown in Figure 6 (for the external structures). Then, as part of the simulation, we are going to represent a flooding event (which occur stochastically and with different magnitudes) and look at implications to the on-site structures and follow the path of the water. Also note that we can include debris in the model (in Figure 6, automobiles are represented that may be moved by the force of water) since the flooding physics tools we are exploring have the capability to represent this phenomena.

For a given flood that is simulated in the virtual NPP model by the RISMC Toolkit, we query the results of the physics related to the water. The simulation then continues by translating the physics-based mechanistic calculation into an impact in the accident scenario. For example, if the structure is cracked due to hydrostatic pressure (say a tank failure), this state would be applied to the component in the model using another stochastic model (in this case, a cracking model). Once the component state is specified, then the scenario would continue since the cracked component may experience a dislocation (the crack grows) or further damage. If the component were a tank containing water, then we might experience additional flow out of the pipe at the point of the crack. In a later section, we describe modeling for internal flooding.

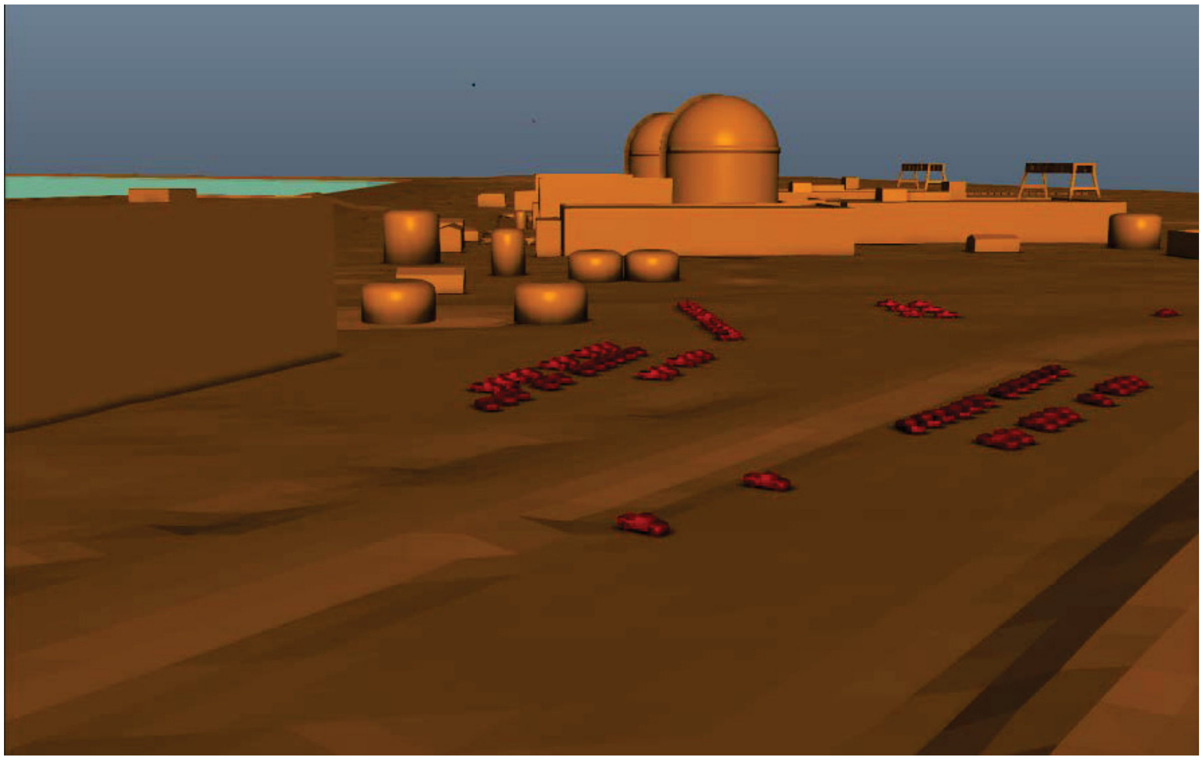


Figure 6. Example of Site Topography and 3D Models to Be Used for the Flooding Simulation.

3. INDUSTRY APPLICATION PROGRESS

In this section we describe the current research investigation results for the external events considered in IA2. Combining the tools developed for each event with a risk-informed methodology encompassing multi-hazard analysis will assist the stakeholder(s) in decision making for future plant modifications or improvements, if necessary.

3.1 Flooding Research and Development

Given that flooding is a potential hazard for nuclear power plants, it is desirable to advance knowledge of how components within these facilities fail when they come into contact with water. Experiments accurately simulating flood and tsunami conditions will be important to expand current knowledge. To complement experimental data, computational models and simulations of such events will be needed. There currently exists programs that simulate fluid motion using a computational technique called Smoothed Particle Hydrodynamics (SPH). Others have used SPH to analyze the durability of coastal structures [1] [2]. None of these programs, however, explore failure potential of relevant power plant components. In this section, we describe the SPH-based tools that were evaluated.

Twenty-two available SPH codes were investigated. Eleven of these are open source and designed for fluid simulation. Note that many of the tools were found to be impractical for use within IA2. For example, they had little documentation, were no longer available for download, or were proprietary. Several codes were chosen to receive further investigation. From there, their operation procedures and output files were detailed and an example simulation and parametric study was performed.

3.2 SPH Theory

Computational Fluid Dynamics (CFD) is a numerical method of analyzing fluid behavior. In general, the two numerical ways of describing fluid behavior are Eulerian and Lagrangian. Both methods use the Navier-Stokes equation deployed by different methods of finite elements to compute a fluid's behavior. The Navier-Stokes equation governs the fluid dynamics and is dependent on density (ρ), velocity (v), body forces such as gravity (F), pressure (P), and dynamic viscosity (μ).

$$\rho\left(\frac{\partial \vec{v}}{\partial t} + (\vec{v}\cdot\nabla)\vec{v}\right) = \vec{F} - \nabla P + \mu\nabla^2\vec{v}$$

Equation 1: Navier-Stokes equation

The Eulerian method uses the equation at fixed nodes of a mesh. The Lagrangian method solves the equation without a mesh, calculating from varied positions. SPH is one of the meshless techniques used to solve complex scenarios.

SPH was developed by Gingold, Monaghan, and Lucy in 1977 for astrophysics [3]. It has been widely used in the field of astrophysics, oceanography, and volcanology. SPH is a meshless Lagrangian method where the fluid is divided into a set of discrete particles. In other words, SPH discretizes the Navier-Stokes equation such that the discrete particles move with the fluid coordinates and carry the fluid information throughout the motion. Particles interact with other particles that are within a certain distance. The particular distance is called the smoothing distance (h). However, the strength of this interaction needs to be varied over distance. The further a particle moves from another, the less it should affect it. A smoothing kernel (W) is used to determine this interaction. It is a normalizable, probabilistic function that approaches zero as the distance between the two particles approaches h.

SPH is a grid based method where a computational frame is developed. A computational frame is a group of nodes where the fluid variables are calculated. After the fluid variables are calculated, the node will adopt a physical meaning. The node represents fluid material carrying physical properties. Since each particle represents a small volume, SPH defines a scalar quantity with mass and density terms. However, the density of each particle is not constant in SPH. The density can be calculated from the following equation.

$$\rho_j = \sum_j m_j \frac{\rho_j}{\rho_j} . W(|\vec{r} - \vec{r_j}|, h) \ = \sum_j m_j W(|\vec{r} - \vec{r_j}|, h)$$

Equation 2: SPH equation for non-constant density

The pressure gradient, in turn, can be estimated with the following equation.

$$\rho_a \nabla P_a = \sum_b m_b (P_b - P_a) \nabla_a W_{ab}$$

Equation 3: Pressure gradient

The equation above implies that the force is zero when the pressure is constant. However, Equation 3 does not conserve linear and angular momentum. A more realistic way to estimate the pressure gradient is shown in Equation 4.

$$\frac{\nabla P}{\rho} = \nabla\left(\frac{P}{\rho}\right) + \left(\frac{P}{\rho^2}\right)\nabla\rho$$

Equation 4: Realistic pressure gradient

The acceleration due to the pressure gradient is shown in Equation 5.

$$\frac{dv_a}{dt} = -\sum_b m_b \left(\frac{P_a}{{\rho_a}^2} + \frac{P_b}{{\rho_b}^2}\right) \nabla_a W_{ab}$$

Equation 5: Acceleration due to pressure gradient

Where the pressure terms are calculated using the following equation.

$$P_i = K\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1$$

Equation 6: Pressure equation

γ is typically 7 and K is chosen such that the speed of sound $\left(\sqrt{\gamma P \rho^{-1}}\right)$ is 10 to 100 times greater than the maximum speed in the simulation. If the smoothing kernel is Gaussian, then the force exerted by particle **a** onto particle **b** is given by Equation 7.

$$F_{ab} = \frac{2 m_a m_b}{h^2}\left(\frac{P_a}{{\rho_a}^2} + \frac{P_b}{{\rho_b}^2}\right)(r_a - r_b) W_{ab}$$

Equation 7: Force exerted between particles

Where the pressure gradient produces symmetrical force between a pair of particles, and hence the linear and angular momentum are conserved.

3.2.1 Smoothing Kernels

SPH is an interpolation method which allows any function to be expressed in terms of its values at a set of disordered points, the particles [3]. The function A(r) is integrated over the space with an interpolating kernel, W.

$$A(r) = \int A(\acute{r})W(r - \acute{r}, h)d\acute{r}$$

Equation 8: Integrating function

The interpolating kernel, W has two properties [4].

$$\int W\,(r - \acute{r}, h)d\acute{r} = 1$$

Equation 9: Interpolating kernel property 1

and

$$\lim_{h\to 0} W(r - \acute{r}, h) = \delta(r - \acute{r})$$

Equation 10: Interpolating kernel property 2

The limit is expressed as the limit of the corresponding integral interpolants. The choice of smoothing kernel is very important in SPH. It must be selected such that the kernel tends to the delta function as the maximum length tends to 0 (h →0) [4]. The kernel must be normalized so that the area under the curve is unity (Equation 9). To calculate the gradient of the kernels, the first derivative must be continuous and well defined.

The Gaussian Function, given by the following equation, was first selected in SPH simulations because of its spherical symmetry [4].

$$W(r, h) = \frac{1}{h^3\pi^{3/2}}e^{-x^2}$$

Equation 11: Gaussian function

Where x = r/h, and W > 0 for all r, meaning that all of the particles in the domain contribute. Therefore, cubic spline kernels are often used so that the contribution of the particles can be limited to a certain smoothing length. The cubic spline kernel is defined in Equation 12 [3],

$$W(r, h) = \frac{1}{\pi h^3}\begin{cases} 1 - \frac{3}{2}x^2 + \frac{3}{4}x^3 & 0 \leq x \leq 1 \\ \frac{1}{4}(2 - x)^2 & 1 \leq x \leq 2 \\ 0 & x \geq 2 \end{cases}$$

Equation 12: Cubic spline kernel

For a numerical approach of this technique, the integral is expressed as the summation of the interpolant, which is shown in Equation 13.

$$A\,(\vec{r}) = \sum_j m_j \frac{A_j}{\rho_j} . W(|\vec{r} - \vec{r_j}|\,, h)$$

Equation 13: Numerical approach for calculating A

$A\,(\vec{r})$ is the weighted summation of all the particles at point r. A_j is the value of A at particle j, m_j is the mass of particle j, ρ_j is the density of particle j, $W(|\vec{r} - \vec{r_j}|$, h) is the weighting factor, and h is the maximum distance. The interpolating kernel is also called the smoothing kernel. If the neighboring points are closer, the influence on each other will be higher. Therefore, the smoothing kernel will be 0 at the maximum distance (h). After taking second derivative of Equation 13, the following equation is obtained,

$$\Delta^2 A(\,\vec{r}) = \sum_j m_j \frac{A_j}{\rho_j} . \Delta^2 W(|\vec{r} - \vec{r_j}|$$

Equation 14: Second derivative of the numerical approach of A

3.2.2 Time Stepping

In SPH, the particles are allowed to move in time, thereby representing the motion of fluids such as water. The time step method involves the force term, viscous diffusion term, and the Courant condition. They can be used to numerically integrate the ordinary differential equations for the physical variable of each particle [3]. The time step δt can be calculated by first calculating δt_f and δt_{cv}. These quantities are defined by the following equations.

$$\delta t_f \;=\; min_a(\frac{h_a}{|f_a|})$$

Equation 15: Force based time step equation

$$\delta t_{cv} = min_a \frac{h}{c_a + .6(\propto c_a + \beta max_b \mu_{ab})}$$

Equation 16: Courant and viscous diffusion time step equation

$$\delta t \;=\; .25\, min\big(\delta t_f\,, \delta t_{cv}\big)$$

Equation 17: Time step equation

δt_f is based on the force per unit mass f and δt_{cv} combines the Courant and the viscous time-step controls. The time step is a very important factor in choosing the number of particles. If the time step is chosen correctly, the total energy should be conserved to within 0.5% for over 400 time steps [3].

3.3 Available SPH Codes

A web search was conducted to determine what SPH codes are available. After conducting the search, there were 3 proprietary codes, 11 open source codes, 2 astrophysics codes, and 6 codes that require special conditions for use. Note that the NEUTRIO code currently in use at the INL was not included in this evaluation since the focus was on the potential for additional codes to be used in the RISMC Toolkit. The 3 proprietary codes are:

- IMPETUS Afea Solver [5]
- RealFlow [6]
- LY-DYNA [7]

The 11 open source codes are:

- SPHysics – Serial [8]
- SPHysics – Parallel [9]
- DualSPHysics [10]
- Fluids v.3 [11]
- PySPH [12]
- GPUSPH [13]
- Sibernetic [14]
- AQUAgpusph [15]
- SPH Distributed Fluid Simulator [16]
- COULWAVE [17]
- Phys X Fluid Sandbox [18]

The 2 astrophysics codes are:

- GADGET [19]
- NDSPMHD [20]

The 6 codes that require a special conditions for use are:

- Fluidix [21]
    - Free for non-commercial use
- GRLab [22]
    - Requires a request for a serial number
- SPH-flow [23]
    - Must join consortium
- Pasimodo [24]
    - Not available for download, but may be able to contact
- PHANTOM [25]
    - Collaborative basis only
- ADCRIC [26]
    - May not use SPH theory

Once a list of SPH codes was compiled, specific codes were chosen to be evaluated. The chosen codes were open source or astrophysics. The following list is the codes that were chosen for further evaluation along with salient features:

- DualSPHysics
    - Most developed of the SPHysics codes
    - Windows or Linux operating system
    - Performs 3D simulations
    - Allows for rigid bodies
- Fluids v.3
    - Windows operating system
    - Performs 3D simulations
    - No rigid bodies
- PySPH

  - Windows, Linux, or Mac operating systems
  - Performs 3D simulations
  - Allows for rigid bodies
- GPUSPH
  - Linux operating system
  - Performs 3D simulations
  - Allows for rigid bodies
- Fluidix
  - Windows, Linux, or Mac operating systems
  - Performs 3D simulations
  - Allows for rigid bodies
- Sibernetic
  - Linux or Mac operating systems
  - Performs 3D simulations
  - Allows for rigid bodies
- AQUAgpusph
  - Linux operating system
  - Performs 3D simulations
  - Allows for rigid bodies
- SPH Distributed Fluid Simulator
  - Linux operating system
  - Performs 3D simulations
  - No rigid bodies
- COULWAVE
  - Windows or Linux operating systems
  - Performs 2D simulations
  - Allows for 2D rigid bodies
- Phys X Fluid Sandbox
  - Windows or Linux operating systems
  - Performs 3D simulations
  - Allows for rigid bodies
- GADGET
  - Linux operating system
  - Performs 3D simulations
  - No rigid bodies
- NDSPMHD
  - Linux operating system
  - Performs 3D simulations
  - No rigid bodies

The following sections provide more detail about each of the codes. They give a description of the code, how to run the code, the output each code provides, as well as a parametric study of the code.

3.4 Investigated Codes

Each of the codes listed above were evaluated to determine which code would best fit the need of incorporating flooding simulation. The first step in evaluating the codes was to download, install, and get the code running. If the code was able to run, the next step was to simulate a simple model. The model that was attempted to be simulated in all of the codes is shown in Figure 7 and Figure 8.

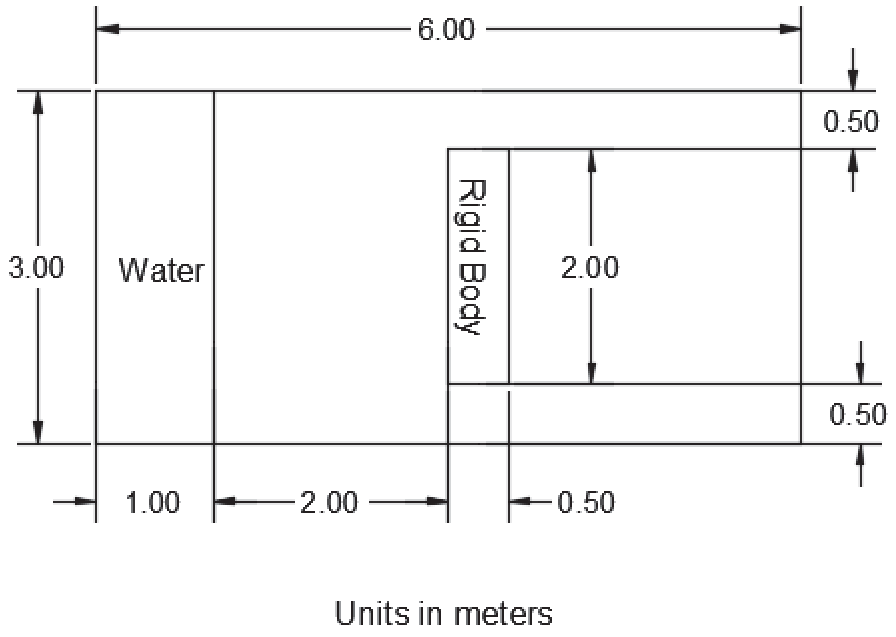


Figure 7: Comparison model, top view

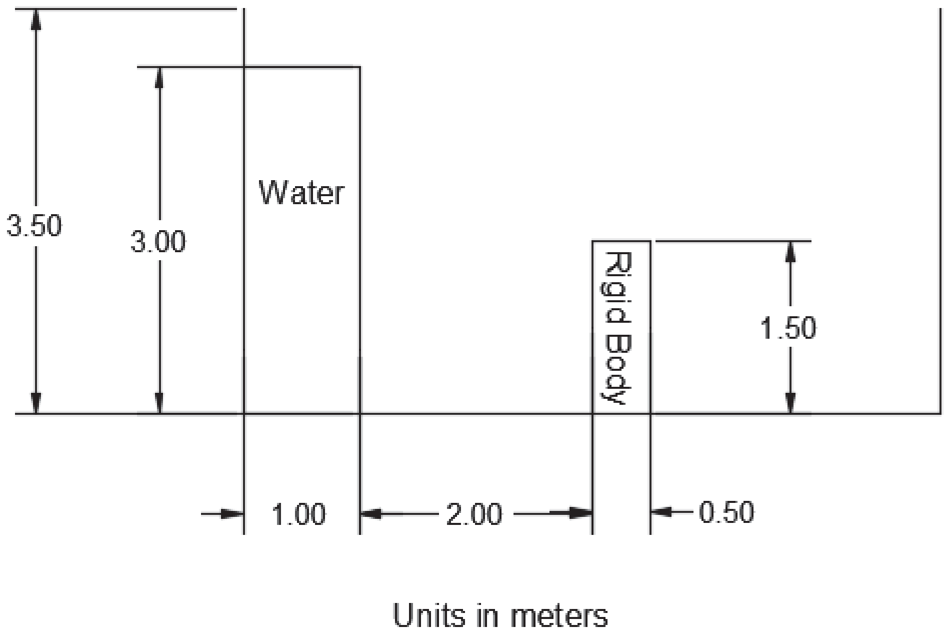


Figure 8: Comparison model, side view

In addition to the provided dimensions, the following additional criteria were added to the simulation:

- No initial velocity
- Gravity only
- Let the simulation run until the water is calm

The above model was selected because it is fairly simple, it incorporates a rigid body, and allows for all of the tested codes to be compared.

3.4.1 Fluids v.3

Fluids v.3 is a large-scale, open source fluids simulator [11]. It was developed by *Rama Hoetzlein*. The code is implemented in C++ and CUDA and has the ability to run using CPU or GPU.

Fluids v.3 can simulate up to 8 million particles. However, it does have limitations. The biggest limitation being that it does not support rigid bodies. The limitation of not supporting rigid bodies would require additional modification in order to get Fluids v.3 up to a required standard.

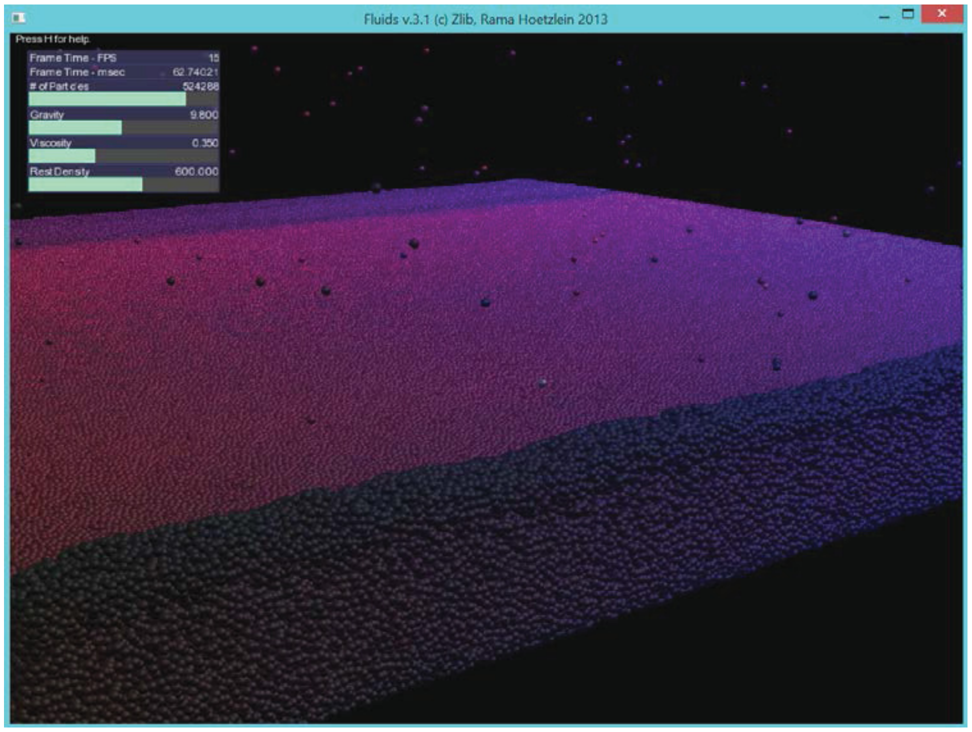
3.4.1.1 Running the Code

In order to run Fluids v.3, the computer operating system must be Windows. The code can then be downloaded from the Fluids v.3 main website [11]. Once everything is downloaded and extracted, the user can then start running the code. If the user wants to modify any of the settings, the scene.xml file must be edited.

After the XML file has been edited by the user, the user can then double click on the Fluids v.3 application in order to show the simulation.

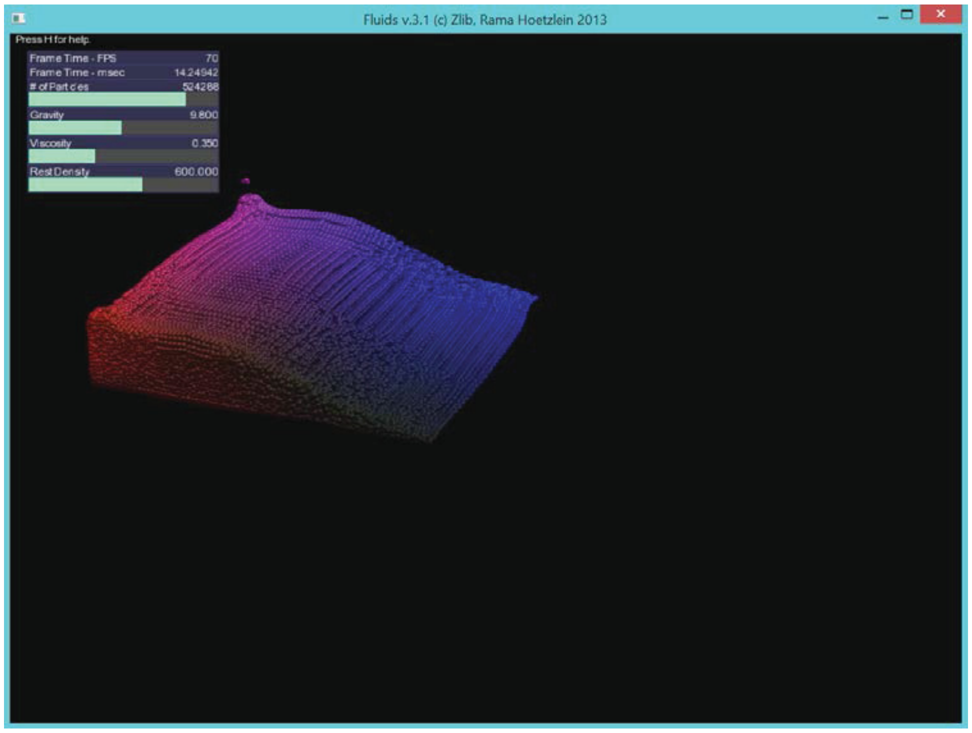
3.4.1.2 Code Output

The output for Fluids v.3 is a saved video of the simulation. Figure 9 through Figure 12 show screenshots of the Fluids v.3 simulation at each of the different scenes provided in the scene.xml file.



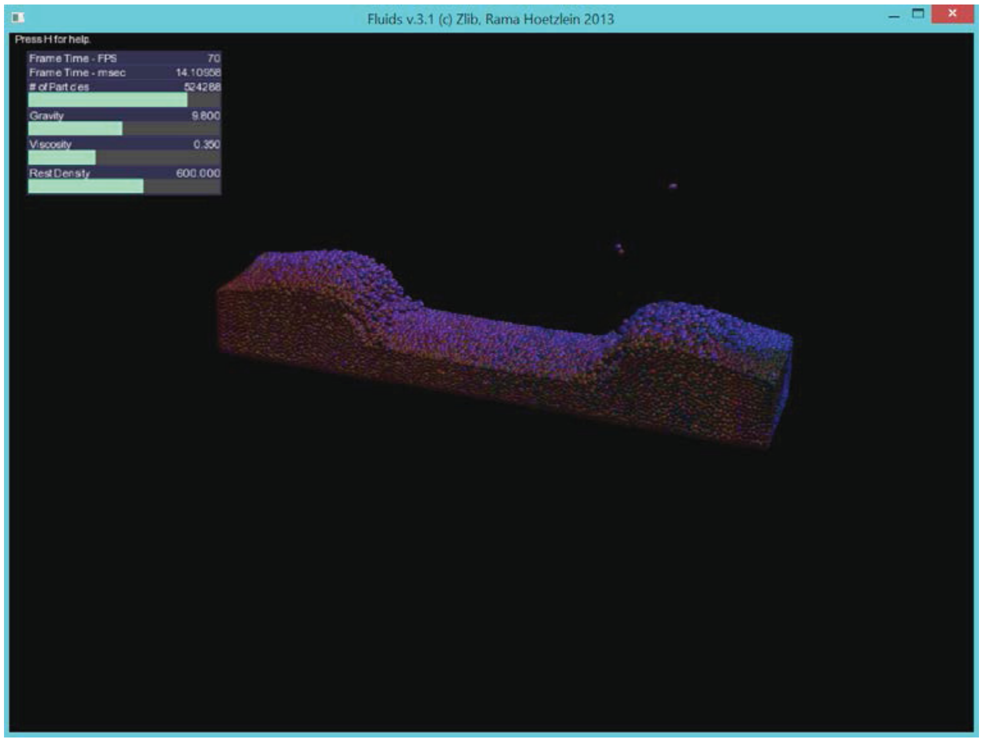
```
Press H for help.
Frame Time - FPS          15
Frame Time - msec   62.74021
# of Particles        524288
Gravity                9.800
Viscosity              0.350
Rest Density         600.000
```

Figure 9: Fluids v.3 simulation, scene 1



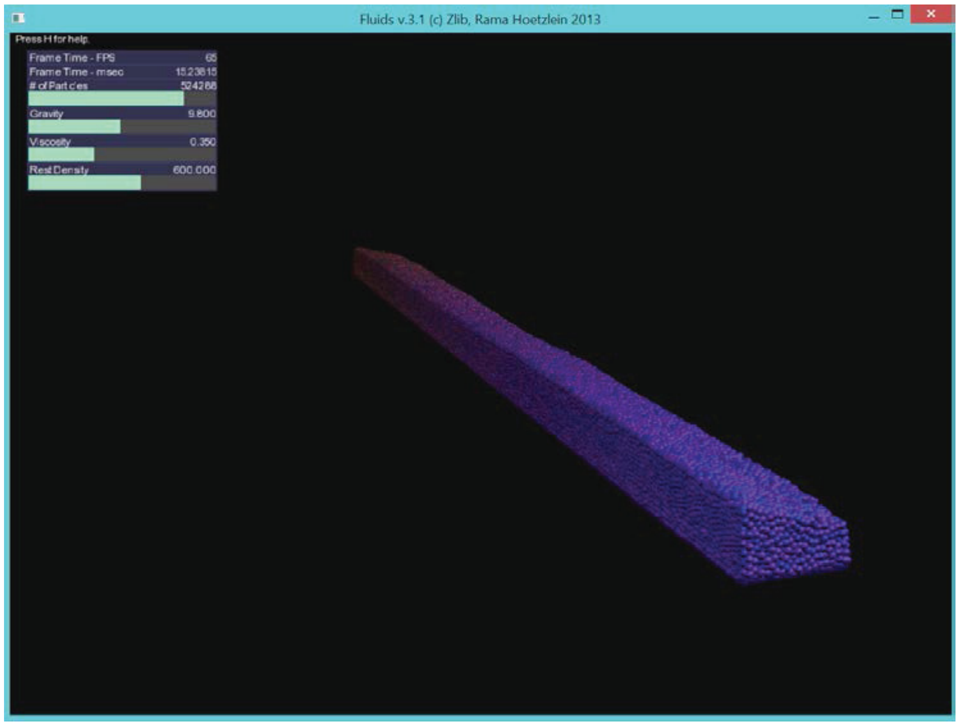
```
Frame Time - FPS          70
Frame Time - msec   14.24942
# of Particles        524288
Gravity                9.800
Viscosity              0.350
Rest Density         600.000
```

Figure 10: Fluids v.3 simulation, scene 2



```
Frame Time - FPS          70
Frame Time - msec   14.10958
# of Particles        524288
Gravity                9.800
Viscosity              0.350
Rest Density         600.000
```

Figure 11: Fluids v.3 simulation, scene 3


Fluids v.3.1 (c) Zlib, Rama Hoetzlein 2013

Press H for help.

Frame Time - FPS	65
Frame Time - msec	15.23815
# of Particles	524288
Gravity	9.800
Viscosity	0.350
Rest Density	600.000

Figure 12: Fluids v.3 simulation, scene 4

No specific particle output could be found during the investigation of Fluids v.3. There may be a way to output individual particle information by modifying the source code, but this would be another feature that must be added that other codes may already have.

3.4.1.3 Comparison Model Simulation

Since Fluids v.3 cannot support rigid bodies, the comparison model could not be performed.

3.4.1.4 Parametric Study

From the Fluids v.3 website, a parametric study was performed by the developer [11]. The data from this study is provided in Table 1 and graphed in Figure 13 and Figure 14.

Table 1: Fluids v.3 parametric study results

# Particles	msec / frame	Hardware Efficiency (particles per second)
4,096	0.68	6,113,432
8,192	1.30	6,301,538
16,384	2.30	7,123,478
32,767	4.20	7,801,666
65,536	8.80	7,447,272
131,072	18.21	7,197,803
262,144	42.30	6,197,257
524,288	98.00	5,349,877
1,048,576	234.00	4,481,094
2,900,800	1085.00	2,673,548
8,388,608	4433.00	1,892,309

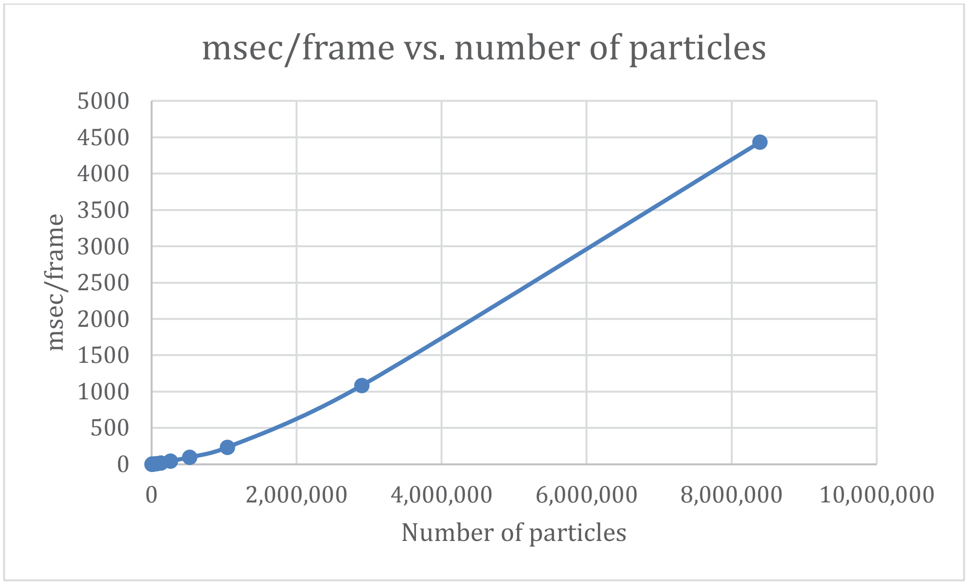


Figure 13: Fluids v.3 msec/frame vs. number of particles

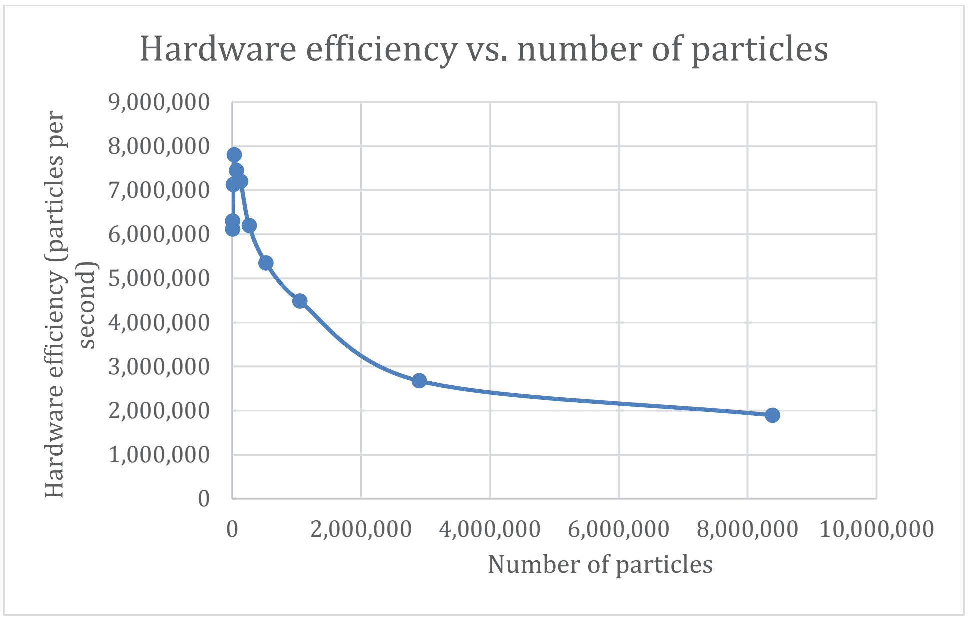


Figure 14: Fluids v.3 hardware efficiency vs. number of particles

From the results above, it can be seen that as the number of particles increase, the time it takes per frame also increases. However, at about 32,000 particles, Fluids v.3 has the best hardware efficiency. The results show that after 32,000 particles the hardware efficiency is going to drop as the number of particles increase.

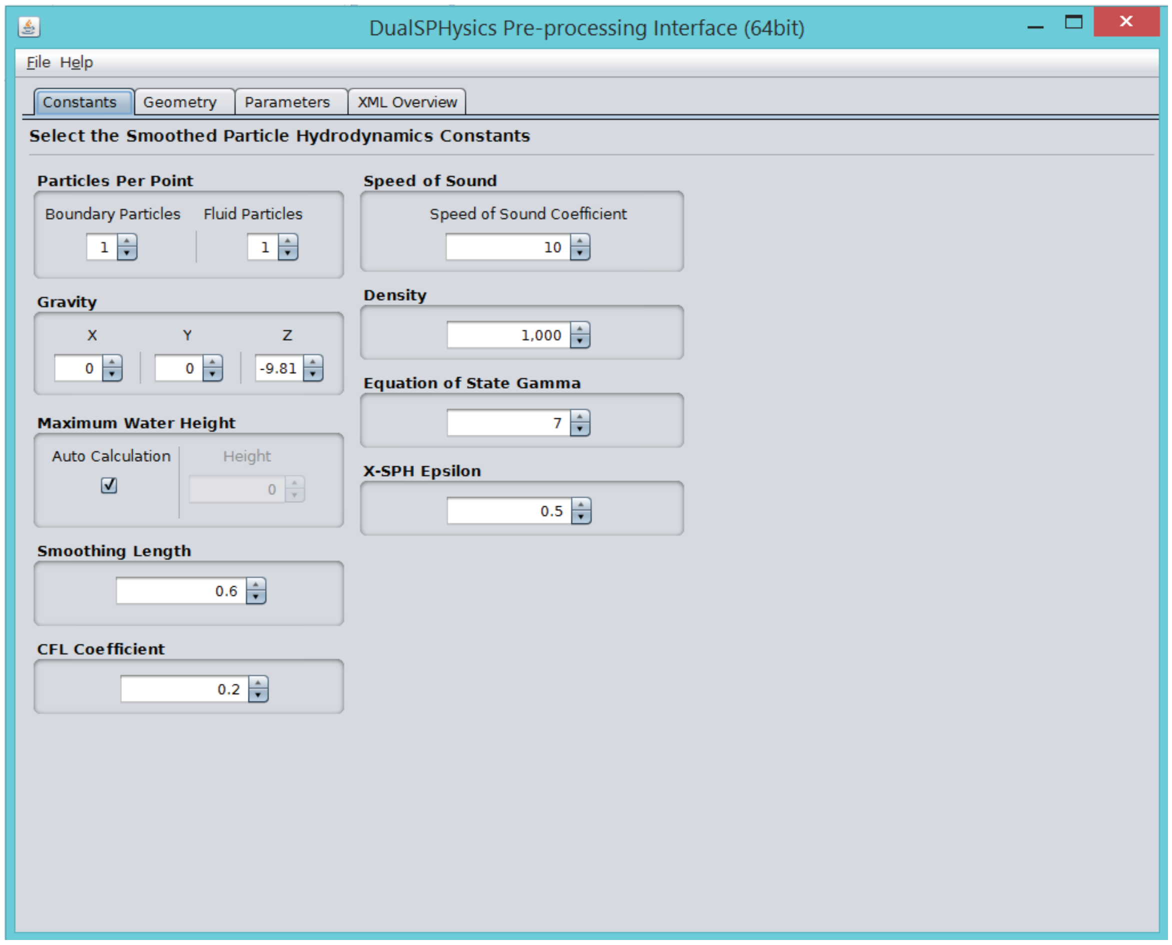
3.4.2 DualSPHysics

DualSPHysics is an open source, SPH code that is based on the SPHysics model [27]. It was developed by the University of Vigo, University of Manchester, University of Parma, Science & Technology Facilities Council, and CEHIDRO, Instituto Superior Tecnico. The code is implemented in C++ and CUDA and has the ability to run using CPUs or GPUs.

DualSPHysics provides documentation as well as a support team that will answer questions as needed. Additionally, it provides multiple examples that allow a first time user to become familiar with the code. The provided examples include floating objects, solid objects, and importing in 3D geometries, such as SolidWork files. Lastly, there has been validation studies performed on DualSPHysics.

3.4.2.1 Running the Code

In order to run DualSPHysics, the computer operating system must either be Windows or Linux based. The code can then be downloaded from the DualSPHysics main website after filling out a short form [27]. Once everything is downloaded and extracted, the user can then start running the code. In order to begin, DualSPHysics requires an XML input file. The XML input file can be created two ways: by hand or by using the DualSPHysics Pre-processing Interface (DPI). The DPI is a GUI that allows the user to create their model. The DPI will then create the XML file based on what the user entered in the GUI. Figure 15 through Figure 17 show the different tabs that the user will use of the DPI.



Select the Smoothed Particle Hydrodynamics Constants

Particles Per Point: Boundary Particles 1; Fluid Particles 1

Gravity: X 0; Y 0; Z -9.81

Maximum Water Height: Auto Calculation ✓; Height 0

Smoothing Length: 0.6

CFL Coefficient: 0.2

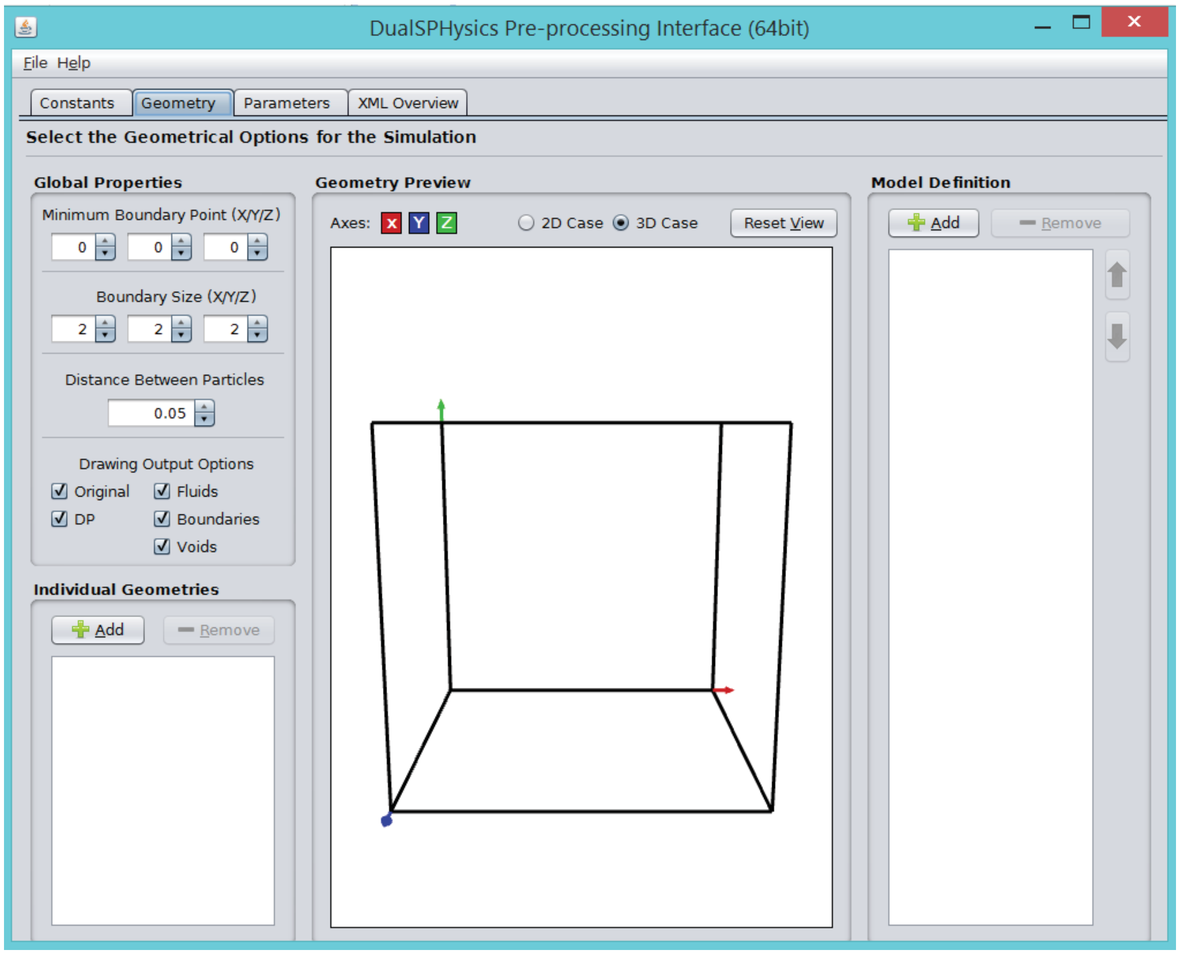
Speed of Sound: Speed of Sound Coefficient 10

Density: 1,000

Equation of State Gamma: 7

X-SPH Epsilon: 0.5

Figure 15: DualSPHysics DPI constants tab



Select the Geometrical Options for the Simulation

Minimum Boundary Point (X/Y/Z): 0 0 0

Boundary Size (X/Y/Z): 2 2 2

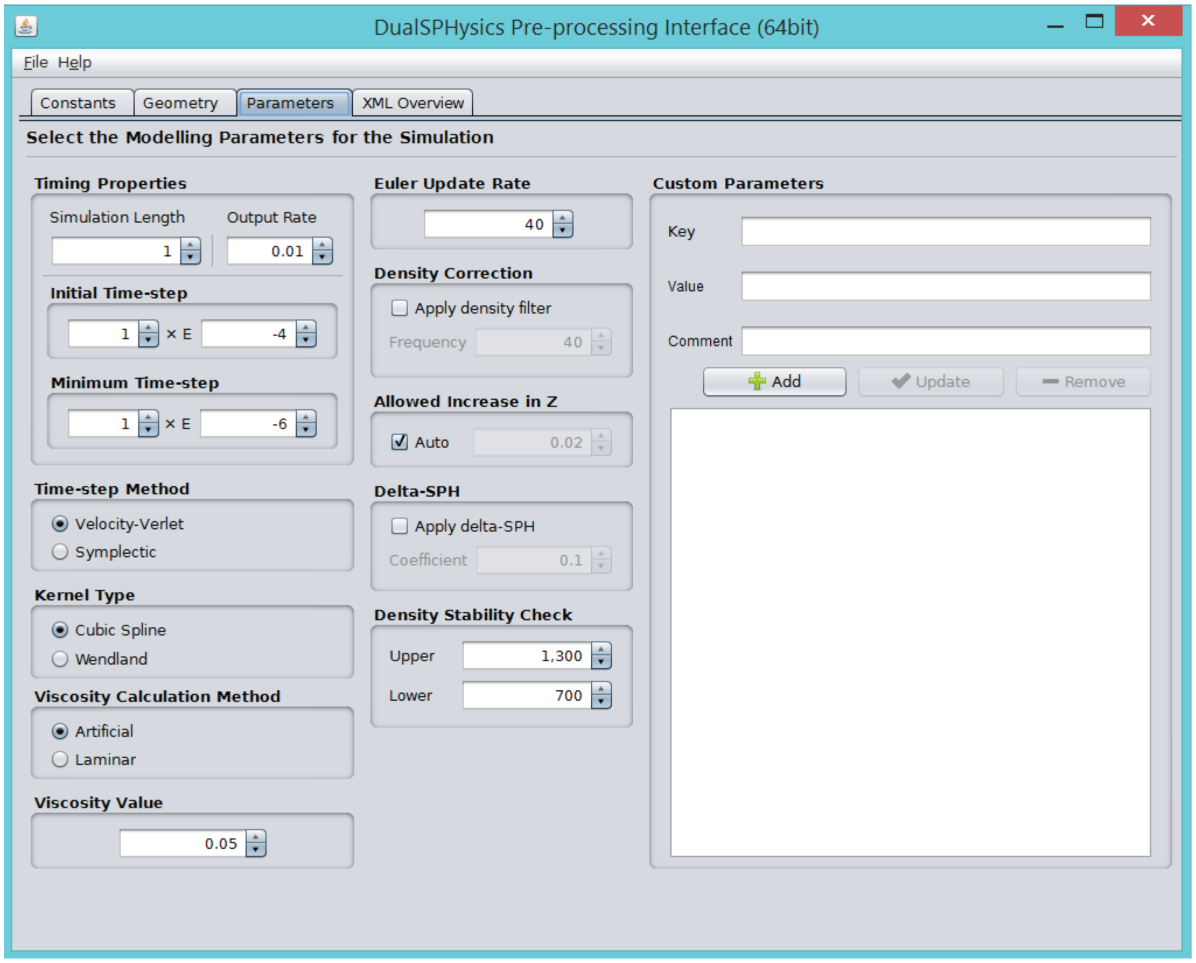
Distance Between Particles: 0.05

Drawing Output Options

- [✓] Original
- [✓] Fluids
- [✓] DP
- [✓] Boundaries
- [✓] Voids

2D Case [✓ 3D Case]

Figure 16: DualSPHysics DPI Geometry tab



Select the Modelling Parameters for the Simulation

Timing Properties

Simulation Length 1

Output Rate 0.01

Initial Time-step 1 × E -4

Minimum Time-step 1 × E -6

Time-step Method

[✓ Velocity-Verlet]

Symplectic

Kernel Type

[✓ Cubic Spline]

Wendland

Viscosity Calculation Method

[✓ Artificial]

Laminar

Viscosity Value 0.05

Euler Update Rate 40

Density Correction

Apply density filter

Frequency 40

Allowed Increase in Z

[✓ Auto] 0.02

Delta-SPH

Apply delta-SPH

Coefficient 0.1

Density Stability Check

Upper 1,300

Lower 700

Custom Parameters

Key

Value

Comment

Figure 17: DualSPHysics DPI parameters tab

Once the user has created the input XML file, a pre-processing executable must be ran. The executable is the GenCase.exe. The GenCase executable creates the required input in order to run the DualSPHysics code. After that is executed, the DualSPHysics executable can be ran. The DualSPHysics executable runs the actual SPH code. Next, a series of post-processing executables can be ran. The post-processing executables are as follows:

- PartVTK.exe – Used to visualize particle data output
- MeasureTool.exe – Used for comparing experimental and numerical values
- BoundaryVTK.exe – Allows for the boundary shapes formed by the boundary particles to be visualized
- IsoSurface.exe – Allows for the simulation to be represented by the surface rather than individual particles

All of the executables have different options associated with them. For a more detailed look at how to run the code or more information about the different executable options, please refer to the DualSPHysics documentation [10].

3.4.2.2 Code Output

The DualSPHysics output will depend on what executables the user runs as well as what options the user specifies. The following is a brief outline of what each executable will output:

- GenCase.exe

    - XML file (input for DualSPHysics.exe)
    - Binary file (input for DualSPHysics.exe)
    - VTK files of geometry (used to visualize the initial setup)
- DualSPHysics.exe
    - Binary files (includes particle information at different times during the simulation)
    - Run.out file (brief summary of the simulation)
    - Optional output files with particle information
        - CSV
        - VTK
        - ASCII
- PartVTK.exe
    - VTK files (used for plotting in ParaView)
- MeasureTool.exe
    - VTK, CSV, or ASCII (includes the desired information at the specified point)
- BoundaryVTK.exe
    - VTK, STL, or PLY (includes loaded information and boundary information)
- IsoSurface.exe
    - VTK (used for plotting in ParaView)

After the user has run the code, another code is provided by DualSPHysics in order to extract more information from the binary files that are outputted from the DualSPHysics executable. The provided code is called ToVTK. When ToVTK is ran, it will extract information from the binary files, such as position, velocity, pressure, density, etc…, and then create CSV files that the user can then open in Excel. The CSV file makes it easier for the user to read the information that was provided by the binary file. It also provides more information than the CSV files that are outputted by the DualSPHysics executable.

3.4.2.3 Comparison Model Simulation

The executables that were ran included the following:

- GenCase.exe
- DualSPHysics.exe
- ParkVTK.exe
- IsoSurface.exe

Figure 18 through Figure 22 show several screenshots of the simulation at different points in time. The left side is the PartVTK output which shows the individual particles and the right side is the IsoSurface output which shows the surface.

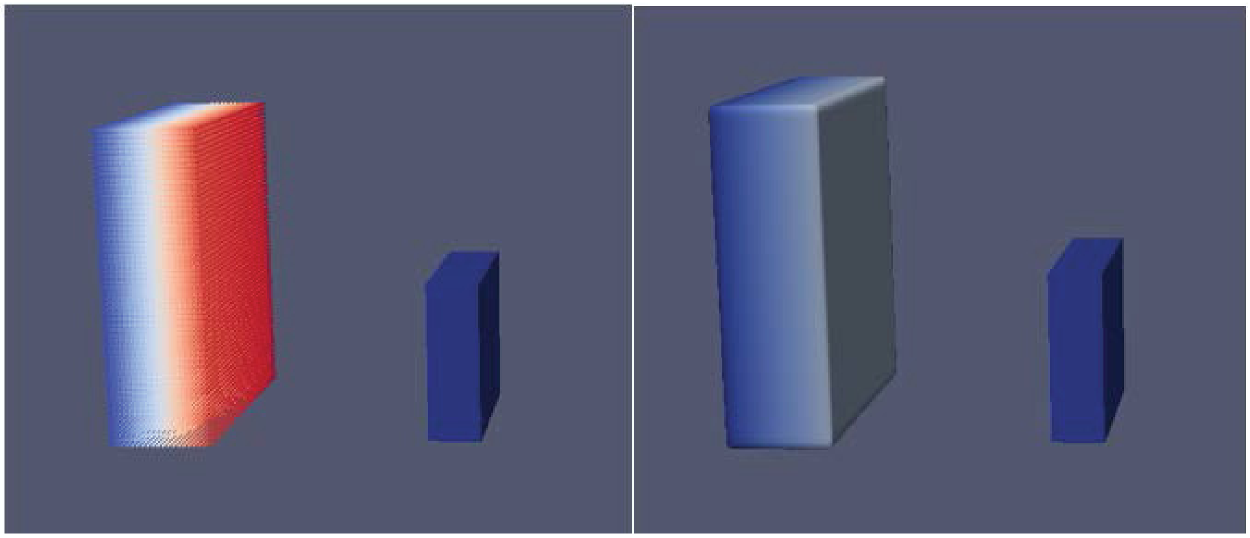


Figure 18: DualSPHysics comparison model initial setup

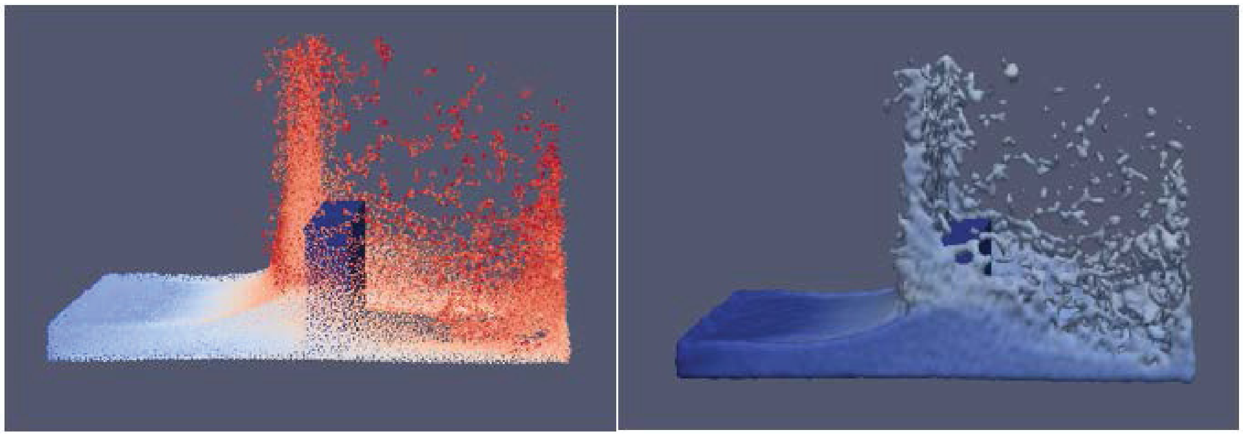


Figure 19: DualSPHysics comparison model 25% complete

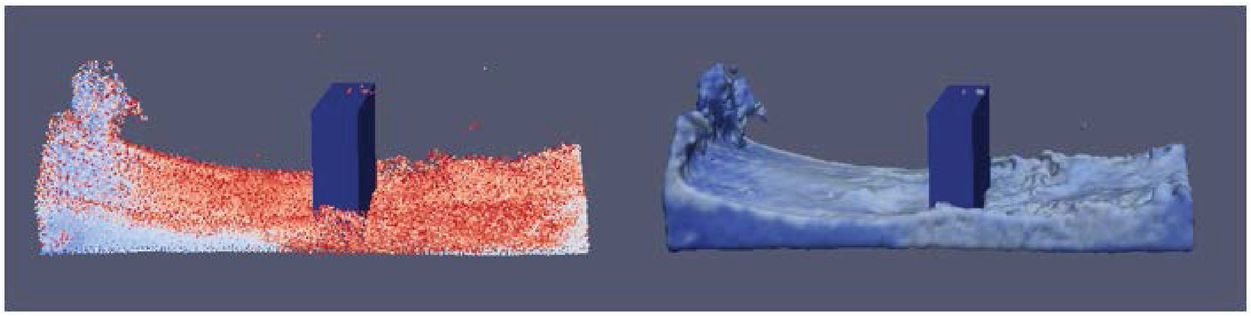


Figure 20: DualSPHysics comparison model 50% complete

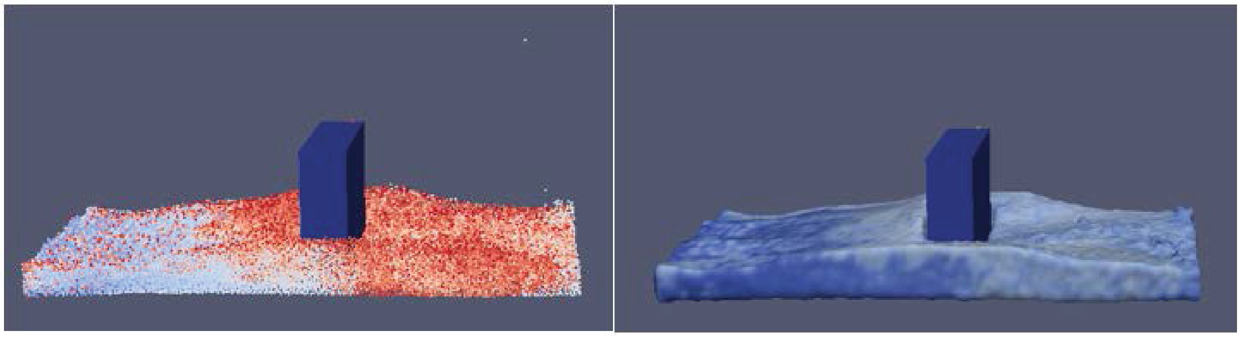


Figure 21: DualSPHysics comparison model 75% complete

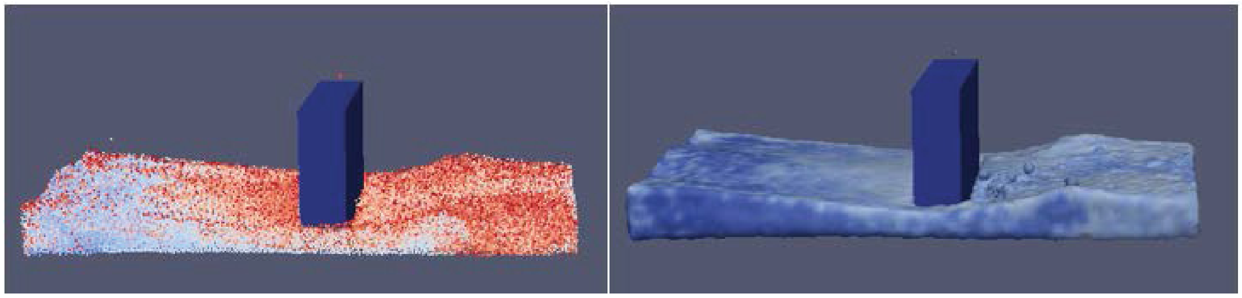


Figure 22: DualSPHysics comparison model 100% complete

3.4.2.4 Parametric Study

Several parametric studies where performed in order to determine runtimes on different computers as well as how the number of particles affects the runtime. For the first parametric study, the simple comparison model was ran on two different computers. The details of the computers are as follows:

- Xi computer
  - 32GB RAM
  - 1TB Hard drive
  - 2 NVidia GeForce GTX Titan X 12GB graphics cards
- HP Envy
  - 16GB RAM
  - 2TB Hard drive
  - 1 NVidia GeForce GTX 850M 4GB graphics card

The simple comparison model was ran on each of these computers both using CPU and GPU options. Table 2 **s**hows the results of the runtimes on each of these computers.

Table 2: DualSPHysics computer runtime comparison results

Computer	Runtime
Xi – GPU	11.8 minutes
HP Envy – GPU	23.9 minutes
Xi – CPU	49.8 minutes
HP Envy – CPU	110.4 minutes

Based on the results from above, the Xi computer using GPU provides the quickest runtimes for DualSPHysics. The next study was to determine how the number of particles affects runtime. Since DualSPHysics sets the distance between the particles, this is what was evaluated knowing that there will be more particles as the distance between particles becomes smaller. Figure 23 shows the plot of runtime vs. distance between particles using GPU and CPU. Table 3 shows all of the plotted data. All of these simulations were run on the Xi computer.

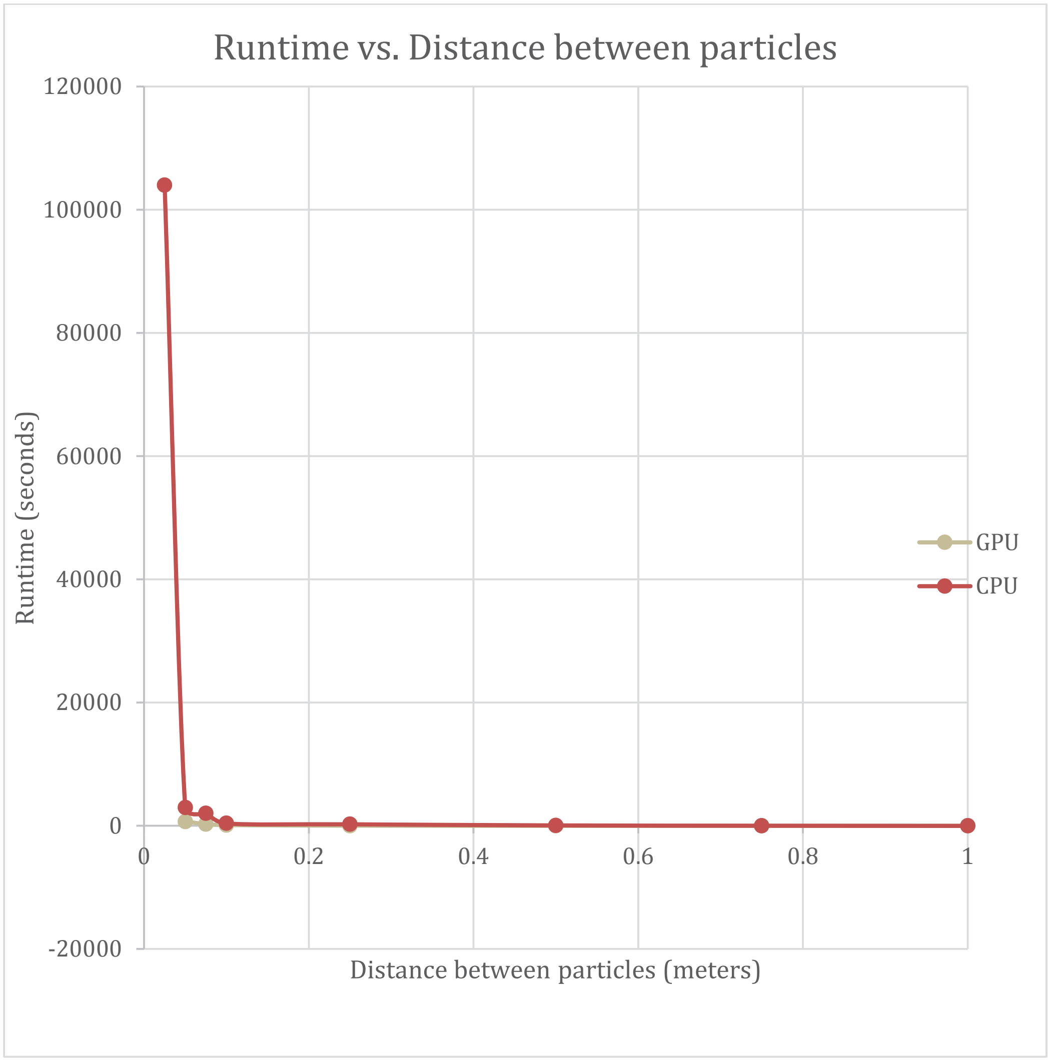


Figure 23: DualSPHysics runtime vs. distance between particles graph

Table 3: DualSPHysics runtime vs. distance between particles results

Distance between particles	Total particles	Fluid Particles	GPU Runtime	CPU Runtime
1	96	6	7.2 sec	12.2 sec
0.75	165	12	9.4 sec	35.9 sec
0.5	433	60	15.1 sec	87 sec
0.25	1,988	528	37.9 sec	265.9 sec
0.1	17,717	8,700	143.1 sec	404.2 sec
0.075	36,351	20,280	277.3 sec	2,057 sec
0.05	106,732	70,800	708 sec	2,988 sec
0.025	714,662	571,200	Cannot run – error	104,019 sec

From the above results, it can be shown that the runtime exponentially increases as the distance between particles decreases. It also shows that the GPU computation works faster than the CPU computation in all working cases. The DualSPHysics team has been notified of the error when running large number of particles on GPU and hopefully a solution can be found.

From the results of the above simulations, it was determined that a specific distance between the particles must be obtained in order to get a reasonable output. The simulation where the distance was greater than 0.1 did not provide realistic results. Based on the results, Equation 18 provides a guideline for the required distance between particles and the volume of the fluid.

$$Distance\ between\ particles = 0.007 * Volume\ of\ fluid\ (in\ m^3)$$

Equation 18: DualSPHysics distance between particles recommendation equation

3.4.3 Fluidix

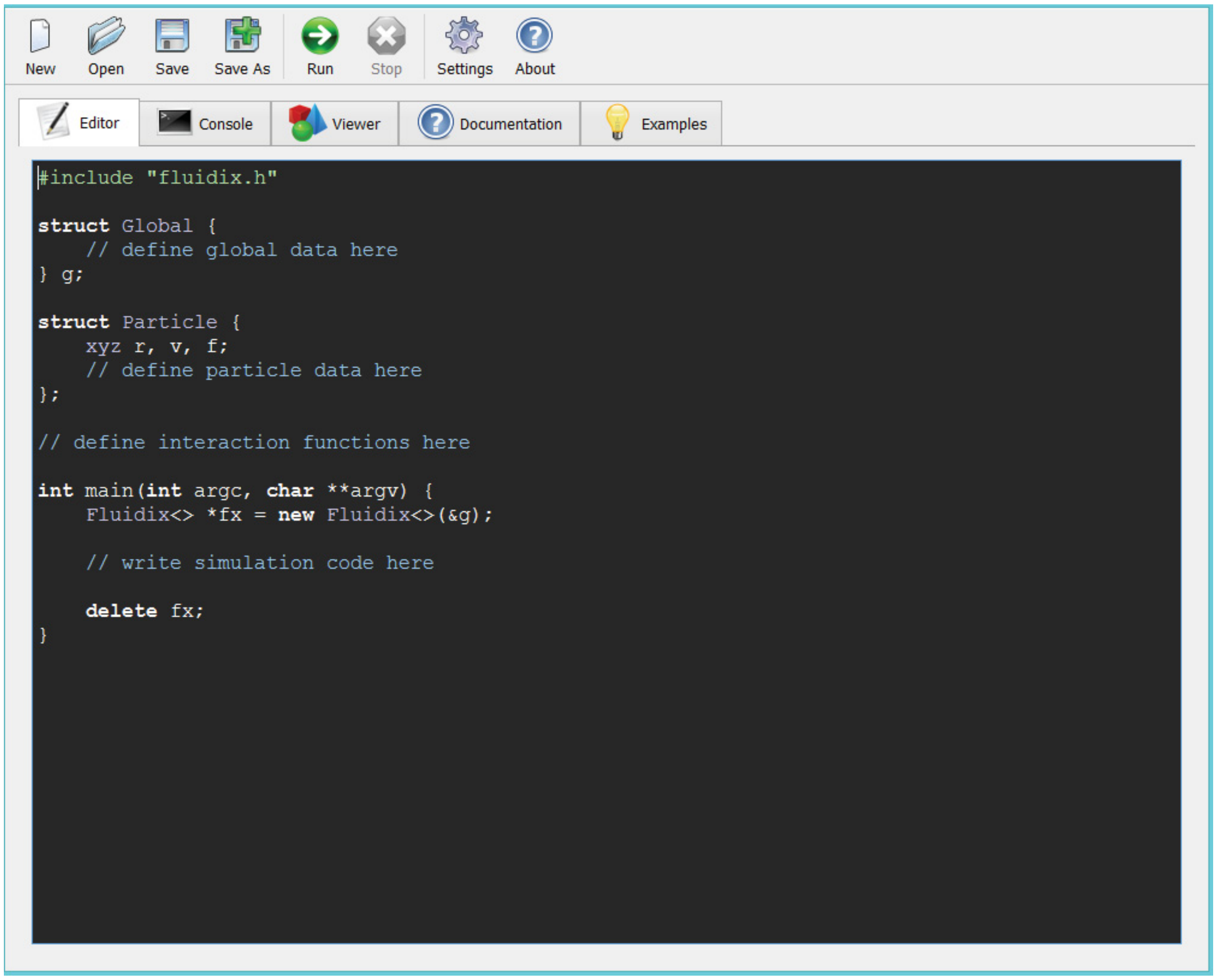
Fluidix is a CUDA-based parallel particle simulation platform which can be applied to practically any particle-based model, such as Molecular Dynamics (MD), Dissipative Particle Dynamics (DPD), SPH, or peridynamics, from large-scale scientific systems to real-time visual effects [21]. It was developed by OneZero Software and is a C++ library. It has the ability to run using CPUs or GPUs.

Fluidix provides documentation as well as contact information if there are any questions. Additionally, it provides multiple examples that allow a first time user to become familiar with the code. The code also allows for 3D modeling files (.STL files) to be imported in. The main difference with Fluidix from other SPH codes is that the user can import the functionality into their own CUDA code.

Fluidix is available for free for non-commercial, personal, and academic use. It may not be redistributed unless OneZero Software has given written permission for a commercial license.

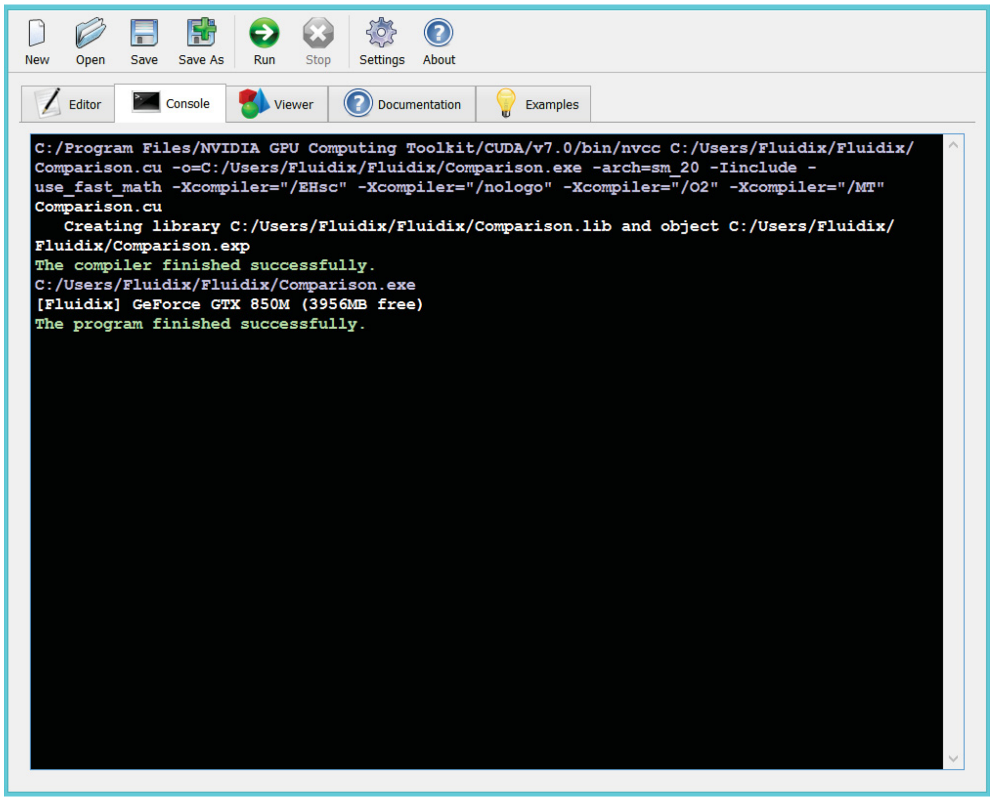
3.4.3.1 Running the Code

There are two ways to run Fluidix. One way is to use Fluidix's IDE. The IDE allows the user to write their code, run the code, and visualize the simulation all in one GUI. The IDE option is the option that was used for the investigation of Fluidix. The other option is to import Fluidix as a library into the users own CUDA code. The option of using Fluidix as a library allows for more flexibility to create one's own program. Figure 24 through Figure 28 show the different tabs of the IDE.



```
#include "fluidix.h"

struct Global {
    // define global data here
} g;

struct Particle {
    xyz r, v, f;
    // define particle data here
};

// define interaction functions here

int main(int argc, char **argv) {
    Fluidix<> *fx = new Fluidix<>(&g);

    // write simulation code here

    delete fx;
}
```

Figure 24: Fluidix IDE editor tab



```
C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v7.0/bin/nvcc C:/Users/Fluidix/Fluidix/
Comparison.cu -o=C:/Users/Fluidix/Fluidix/Comparison.exe -arch=sm_20 -Iinclude -
use_fast_math -Xcompiler="/EHsc" -Xcompiler="/nologo" -Xcompiler="/O2" -Xcompiler="/MT"
Comparison.cu
   Creating library C:/Users/Fluidix/Fluidix/Comparison.lib and object C:/Users/Fluidix/
Fluidix/Comparison.exp
The compiler finished successfully.
C:/Users/Fluidix/Fluidix/Comparison.exe
[Fluidix] GeForce GTX 850M (3956MB free)
The program finished successfully.
```

Figure 25: Fluidix IDE console tab

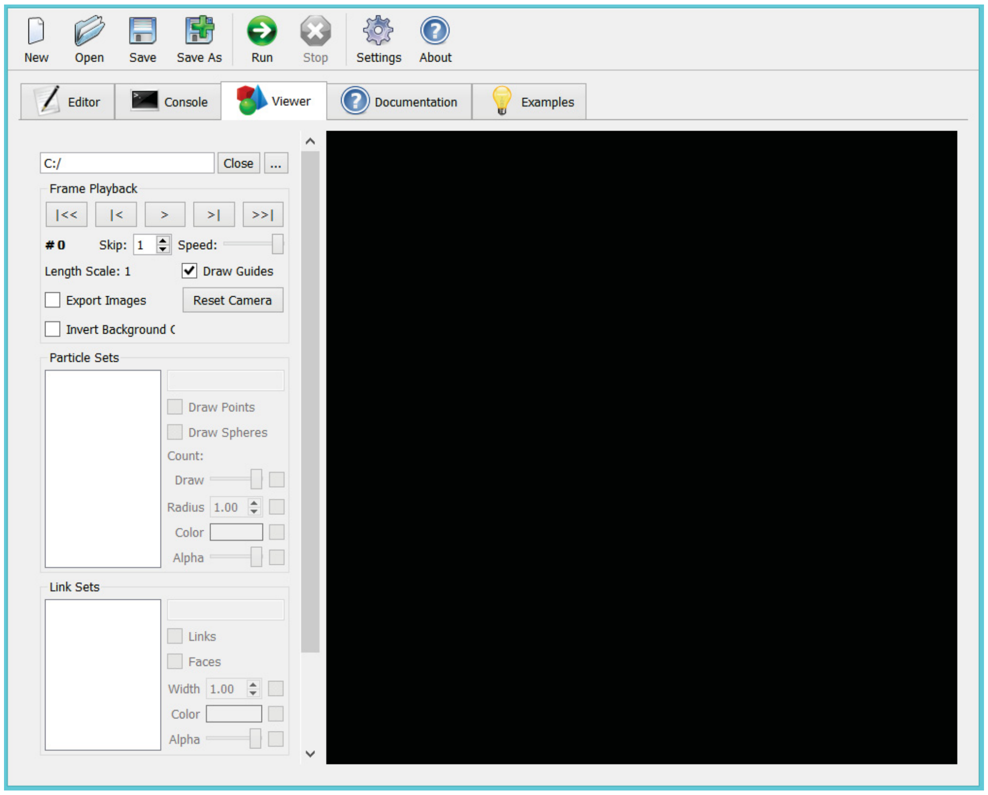
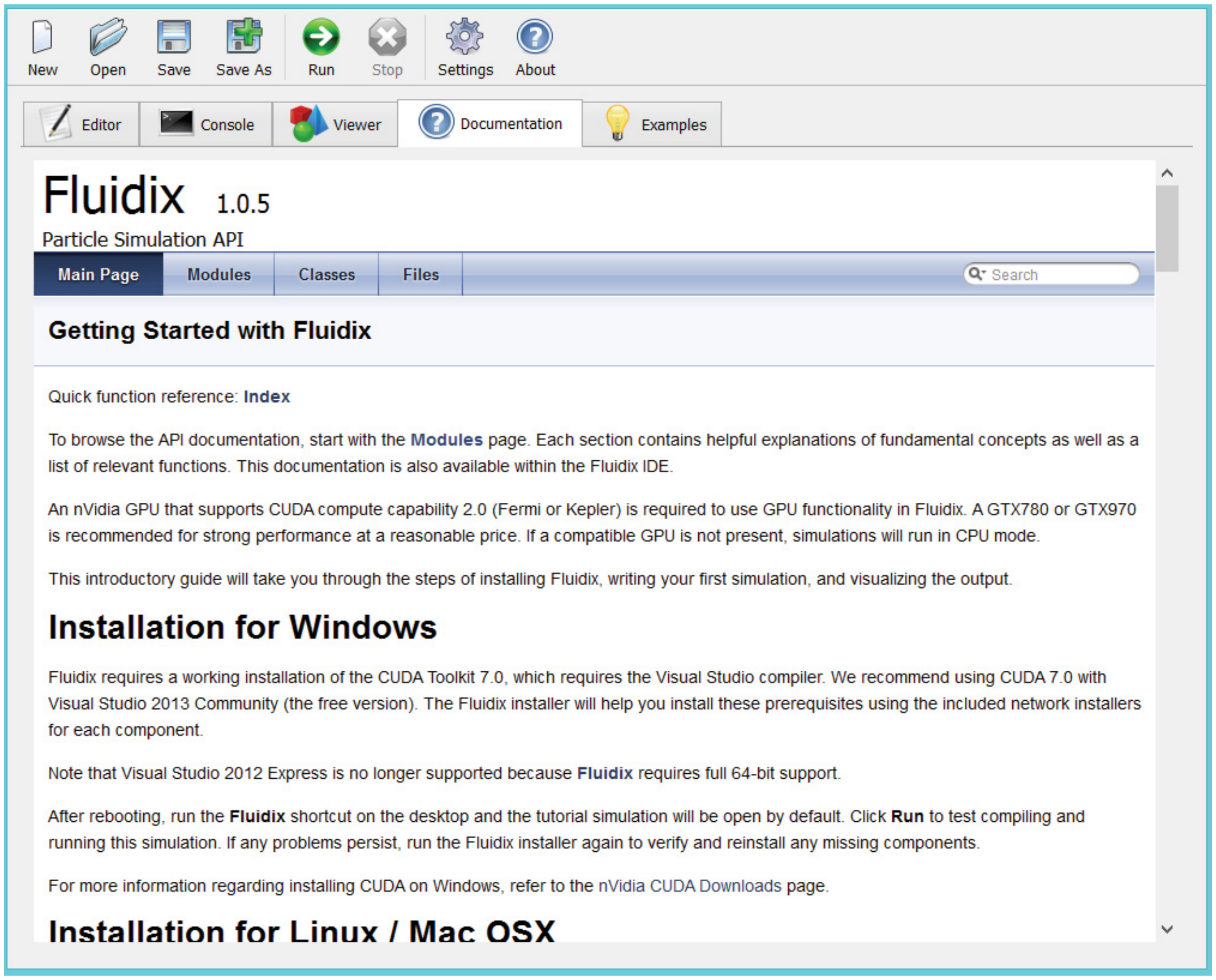


Figure 26: Fluidix IDE viewer tab



Fluidix 1.0.5

Particle Simulation API

Getting Started with Fluidix

Quick function reference: **Index**

To browse the API documentation, start with the **Modules** page. Each section contains helpful explanations of fundamental concepts as well as a list of relevant functions. This documentation is also available within the Fluidix IDE.

An nVidia GPU that supports CUDA compute capability 2.0 (Fermi or Kepler) is required to use GPU functionality in Fluidix. A GTX780 or GTX970 is recommended for strong performance at a reasonable price. If a compatible GPU is not present, simulations will run in CPU mode.

This introductory guide will take you through the steps of installing Fluidix, writing your first simulation, and visualizing the output.

Installation for Windows

Fluidix requires a working installation of the CUDA Toolkit 7.0, which requires the Visual Studio compiler. We recommend using CUDA 7.0 with Visual Studio 2013 Community (the free version). The Fluidix installer will help you install these prerequisites using the included network installers for each component.

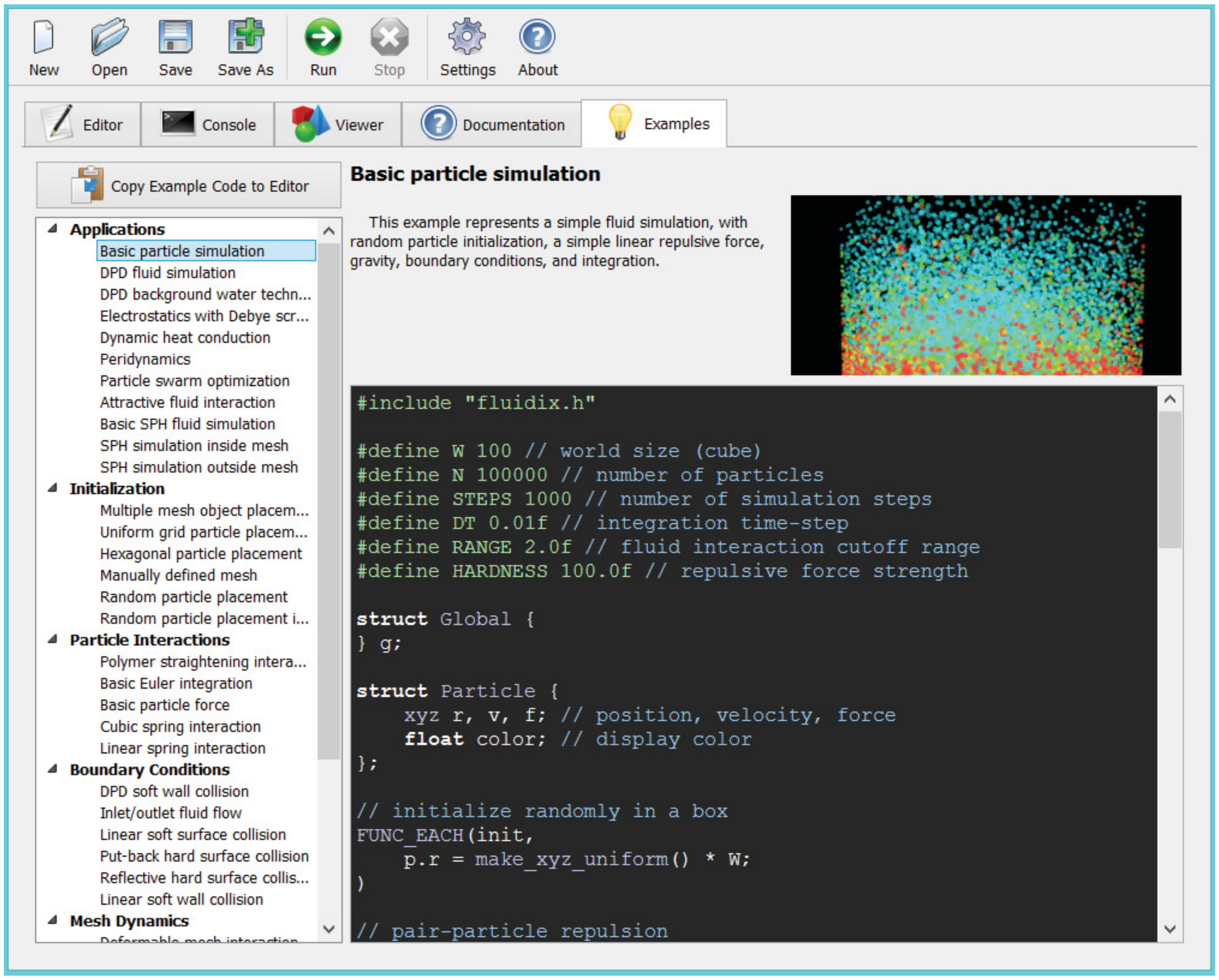
Note that Visual Studio 2012 Express is no longer supported because **Fluidix** requires full 64-bit support.

After rebooting, run the **Fluidix** shortcut on the desktop and the tutorial simulation will be open by default. Click **Run** to test compiling and running this simulation. If any problems persist, run the Fluidix installer again to verify and reinstall any missing components.

For more information regarding installing CUDA on Windows, refer to the nVidia CUDA Downloads page.

Installation for Linux / Mac OSX

Figure 27: Fluidix IDE documentation tab


New
Open
Save
Save As
Run
Stop
Settings
About
Editor
Console
Viewer
Documentation
Examples
Copy Example Code to Editor
Applications
Basic particle simulation
DPD fluid simulation
DPD background water techn...
Electrostatics with Debye scr...
Dynamic heat conduction
Peridynamics
Particle swarm optimization
Attractive fluid interaction
Basic SPH fluid simulation
SPH simulation inside mesh
SPH simulation outside mesh
Initialization
Multiple mesh object placem...
Uniform grid particle placem...
Hexagonal particle placement
Manually defined mesh
Random particle placement
Random particle placement i...
Particle Interactions
Polymer straightening intera...
Basic Euler integration
Basic particle force
Cubic spring interaction
Linear spring interaction
Boundary Conditions
DPD soft wall collision
Inlet/outlet fluid flow
Linear soft surface collision
Put-back hard surface collision
Reflective hard surface collis...
Linear soft wall collision
Mesh Dynamics

Basic particle simulation

This example represents a simple fluid simulation, with random particle initialization, a simple linear repulsive force, gravity, boundary conditions, and integration.

```
#include "fluidix.h"

#define W 100 // world size (cube)
#define N 100000 // number of particles
#define STEPS 1000 // number of simulation steps
#define DT 0.01f // integration time-step
#define RANGE 2.0f // fluid interaction cutoff range
#define HARDNESS 100.0f // repulsive force strength

struct Global {
} g;

struct Particle {
    xyz r, v, f; // position, velocity, force
    float color; // display color
};

// initialize randomly in a box
FUNC_EACH(init,
    p.r = make_xyz_uniform() * W;
)

// pair-particle repulsion
```

Figure 28: Fluidix IDE examples tab

Once the user has wrote the code in the Fluidix IDE editor tab, the user can run it by pressing the run button on the IDE. As the code is running, the console tab will state what step it is at or if the code has an error.

3.4.3.2 Code Output

The Fluidix output is DAT files, PNG files, and text output. The DAT files contain the simulation video and can be played using the Fluidix viewer in the IDE. The PNG files are pictures of each frame of the simulation. The text output is shown on the console, but could be modified to be outputted into an external text file or CSV file. The text output provides what the user has specified such as density, position, and velocity. Depending on the information and type of file that data needs to be saved to, the user can write a code to perform that task.

3.4.3.3 Comparison Model Simulation

That code was run using the Fluidix IDE. Figure 29 through Figure 33 show screenshots of the simulation at different points in time.

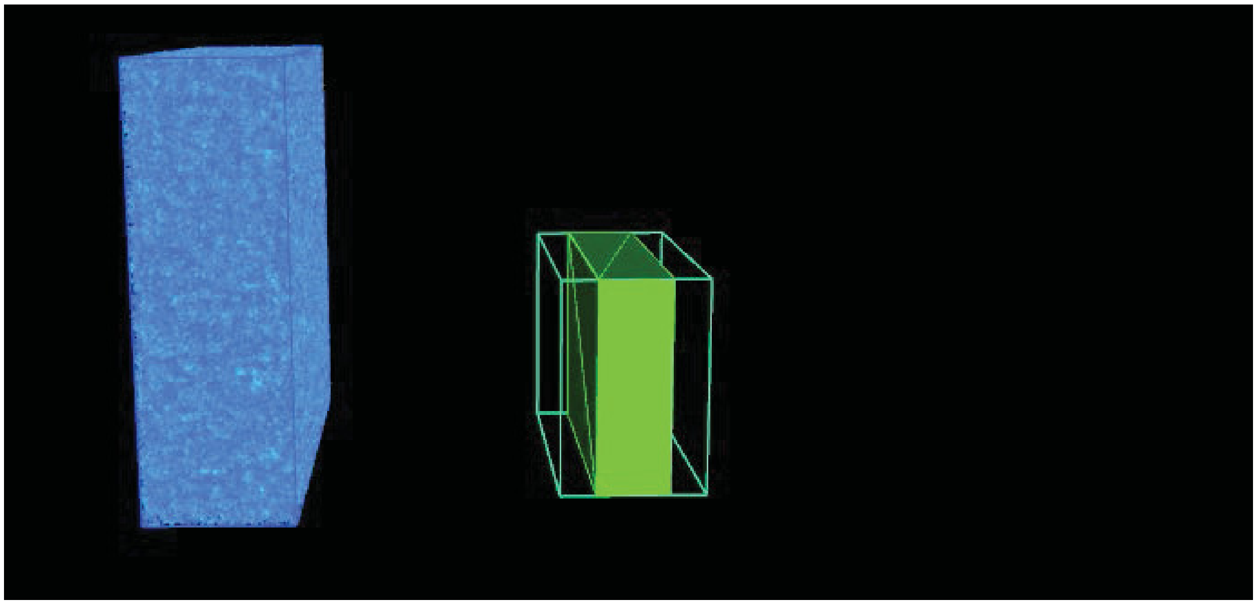


Figure 29: Fluidix comparison model initial setup

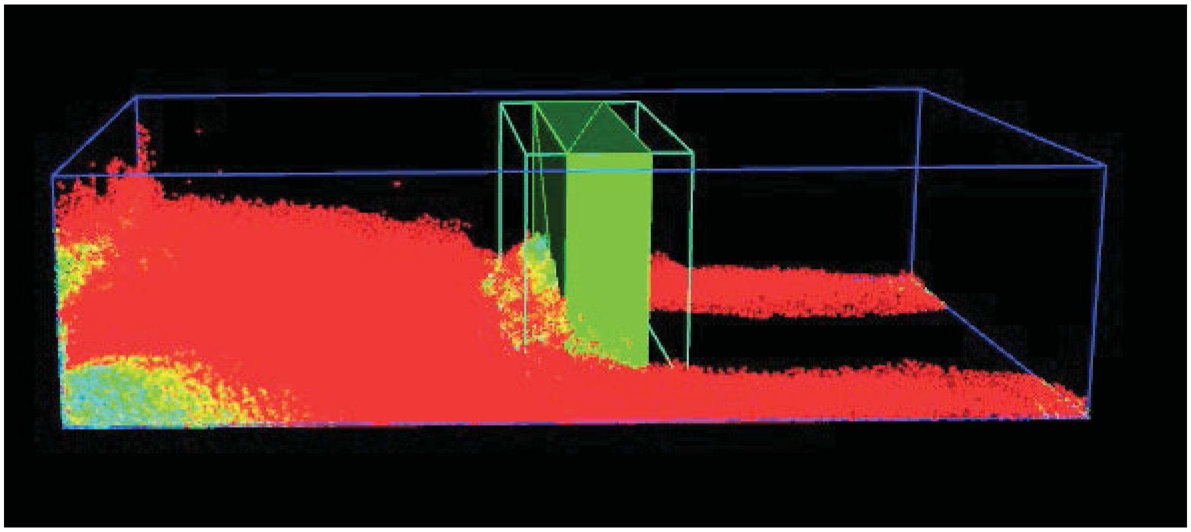


Figure 30: Fluidix comparison model 25% complete

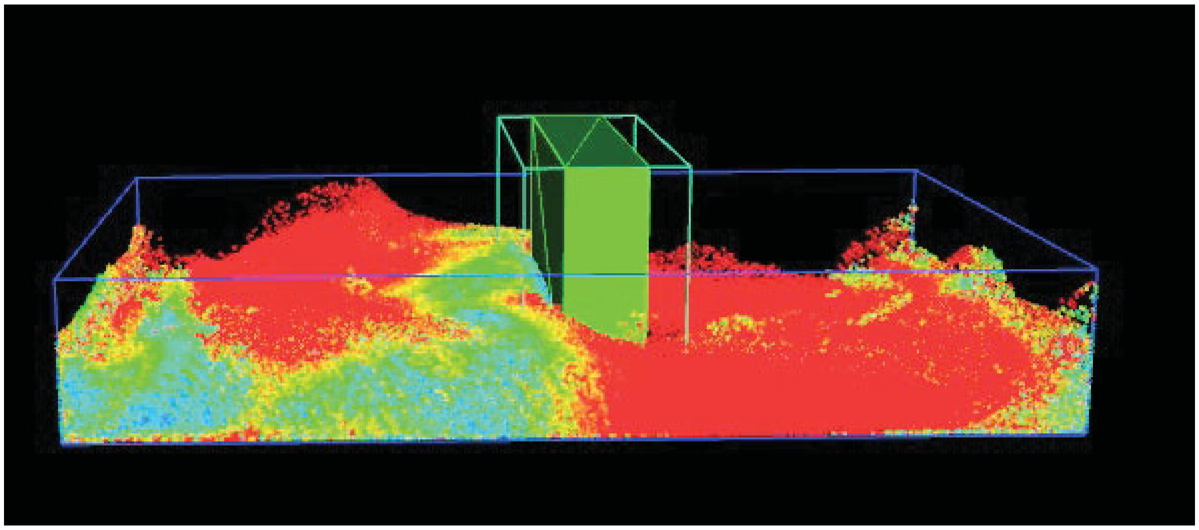


Figure 31: Fluidix comparison model 50% complete

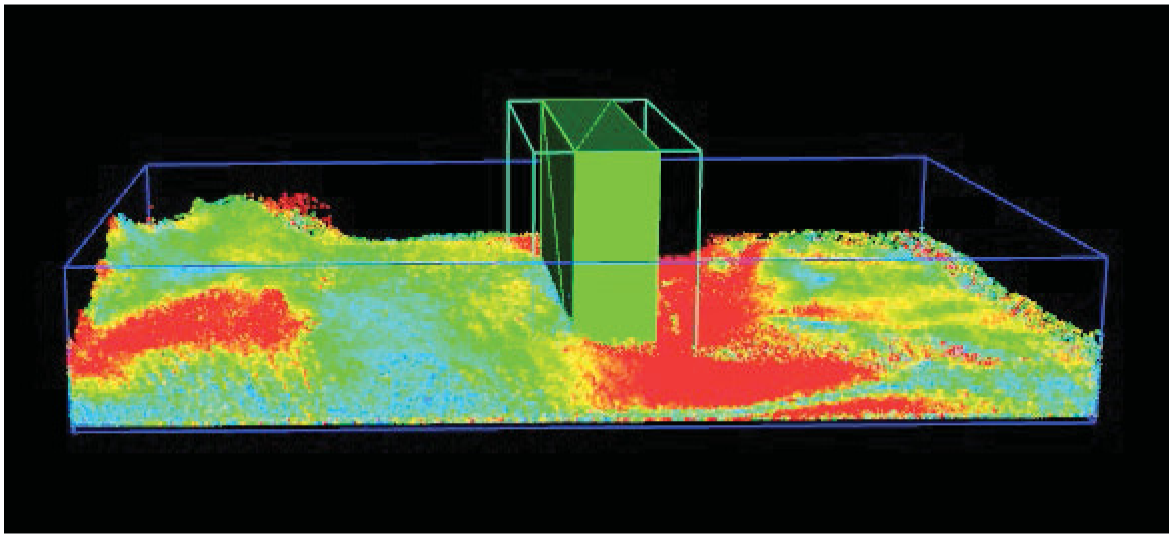


Figure 32: Fluidix comparison model 75% complete

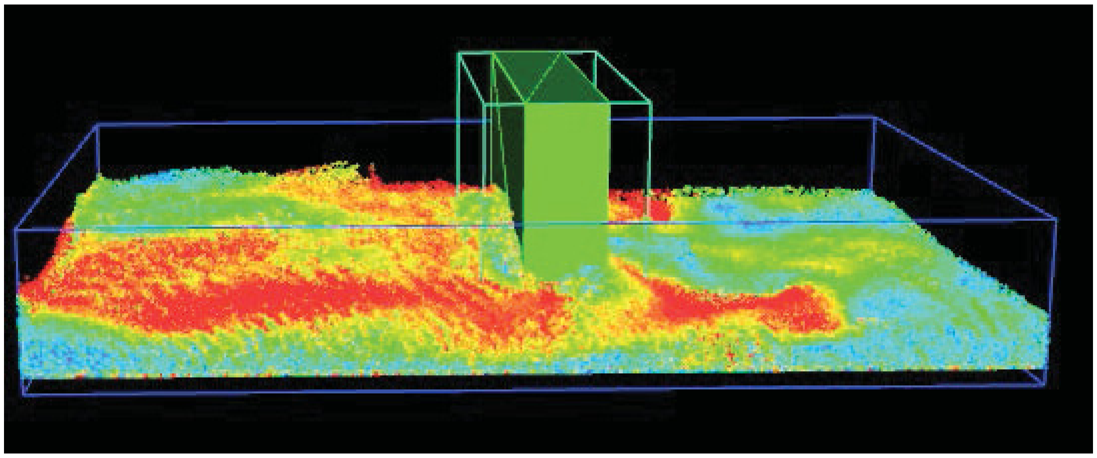


Figure 33: Fluidix comparison model 100% complete

3.4.3.4 *Parametric Study*

Several parametric studies where performed in order to determine runtimes on different computers as well as the how the number of particles affects the runtime. For the first parametric study, the simple comparison model was ran on two different computers. The details of the computers are as follows:

- Xi computer
    - 32GB RAM
    - 1TB Hard drive
    - 2 NVidia GeForce GTX Titan X 12GB graphics cards
- HP Envy
    - 16GB RAM
    - 2TB Hard drive
    - 1 NVidia GeForce GTX 850M 4GB graphics card

The simple comparison model was ran on each of these computers both using CPU and GPU options. There were 50,000 particles used in order to reduce the runtime associated with using CPU. Table 4 shows the results of the runtimes on each of these computers.

Table 4: Fluidix computer runtime comparison results

Computer	Runtime
Xi – GPU	55.4 sec
HP Envy – GPU	135.5 sec
Xi – CPU	1,890.8 sec
HP Envy – CPU	2,411.4 sec

Based on the results from above, the Xi computer using GPU provides the quickest runtime for Fluidix. The next study was to determine how the number of particles affects runtime. Figure 34 shows the plot of runtime vs. number of particles using GPU. Table 5 shows all of the plotted data. All of these simulations were run on the Xi computer.

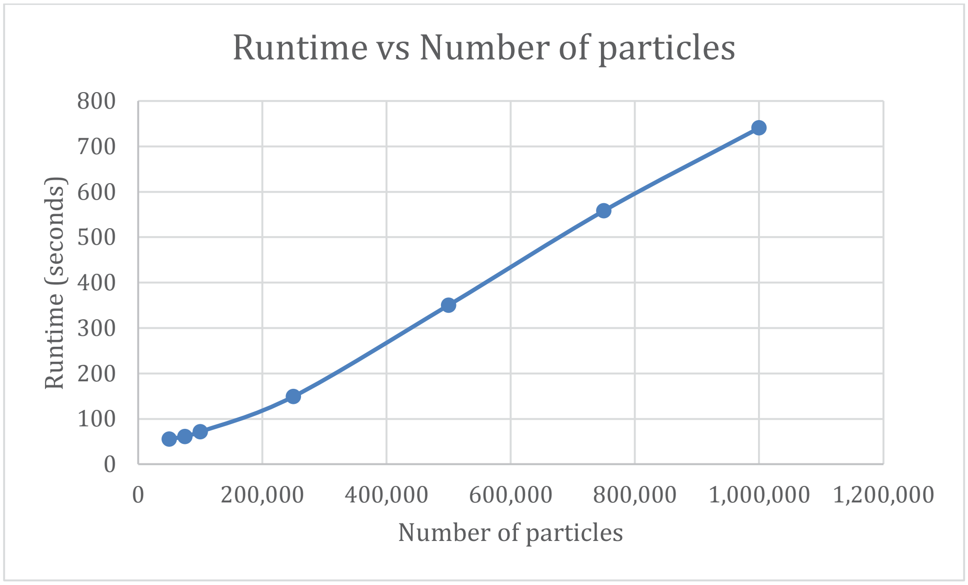


Figure 34: Fluidix runtime vs. number of particles graph

Table 5: Fluidix runtime vs. number of particles results

Fluid Particles	GPU Runtime
50,000	55.4 sec
75,000	61.1 sec
100,000	71.3 sec
250,000	149.1 sec
500,000	350.4 sec
750,000	558.2 sec
1,000,000	740.5 sec
5,000,000	Error

From the above results, it can be shown that the runtime linearly increases as the number of particles increase. At 5 million particles, the graphics card driver kept crashing. There may be a way to fix the error by altering the code. One possible solution could be to change the mass of each particle. By changing the mass, the size of each particle would decrease resulting in there being more room for more particles. Another solution might be to write the code so it is based on the size of the particle or the distance between particles rather than randomly distributing the particles in a given volume.

3.4.4 GPUSPH

GPUSPH is a CUDA enabled open source SPH code, which was first developed by Alexis Herault. GPUSPH was first developed for a lava flow project called “Realization of the lava flow invasion hazard map at Mt Etna and methods for its dynamic update”. GPUSPH is derived from a FORTRAN based code, SPHysics. GPUSPH can be run on NVidia’s Ge Force 86000, 88000, the Tesla 1060 and other GTX cards. GPUSPH contains the source code for some examples such as Dam Break, Open Channel, Test Topo, and Wave Tank. The source code is written in C++ which is compatible with CUDA. The key features of GPUSPH are listed below [13].

Supports multiple SPH formulations, smoothing kernels, viscosities, corrections and boundary conditions.

Supports planes, variable gravity and moving boundaries

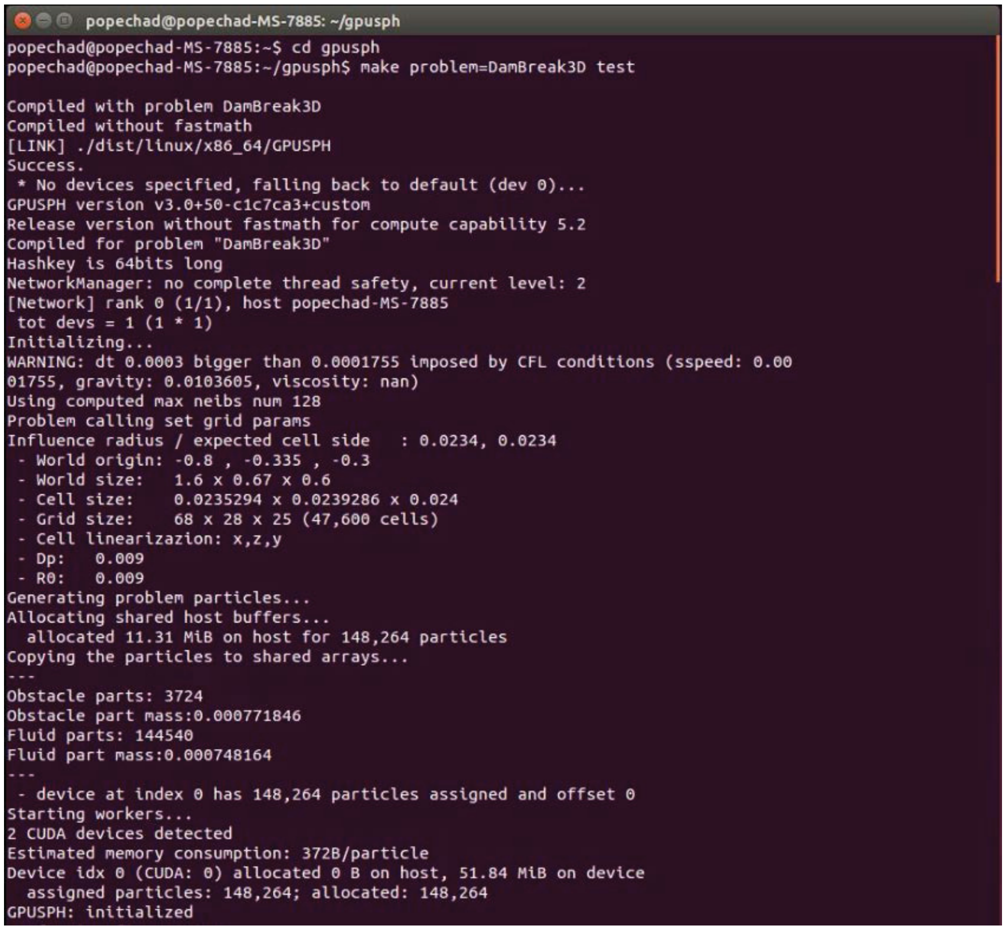
Surface objects joints

Supports multi-GPU and multi-node modes

3.4.4.1 Running the Code

The GPUSPH code can be downloaded from their main website. After the tar file has been downloaded, the files can be extracted into a GPUSPH folder. In order to run the code, GPUSPH must be compiled with the running CUDA version on the computer. To compile GPUSPH, the following command must be entered into the command line under the GPUSPH directory, **make compute=XX test**, where XX is 11, 12 or 20 for compute capability of 1.1, 1.2 or 2.0 [13]. The compute capability options only need to be specified only once, until the compute capability of the graphics card has been set to the Make file. GPUSPH uses the make file to select the problem to run. After compilation, an executable GPUSPH file will be saved into the GPUSPH directory. Now, the program GPUSPH is ready to run the simulations. A provided example can be run by entering **make Problem=DamBreadk3D test** in the terminal.

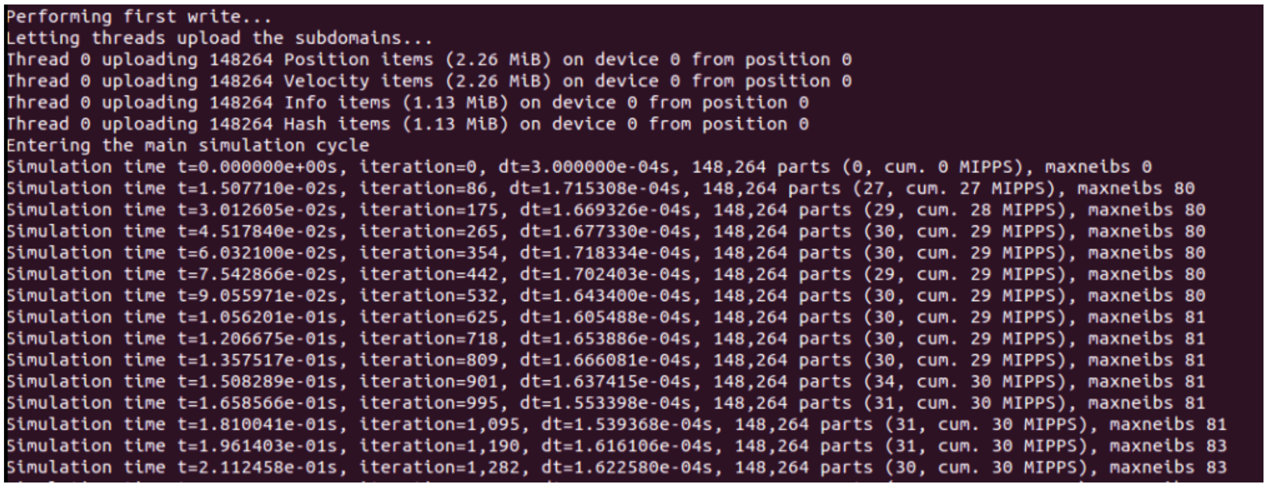
After the command has been entered in the terminal, the GPUSPH will begin with displaying the size of the simulation, the number of fluid particles, and the number of obstacle particles. Then, the problem particles will be generated which contains fluid particles and obstacle particles. After that, it will detect the CUDA version and number of CUDA devices on the computer. The initialization is then complete for the problem and the code will start the simulation calculation for the problem. The following image will be seen in the terminal display.



```
popechad@popechad-MS-7885: ~/gpusph
popechad@popechad-MS-7885:~$ cd gpusph
popechad@popechad-MS-7885:~/gpusph$ make problem=DamBreak3D test

Compiled with problem DamBreak3D
Compiled without fastmath
[LINK] ./dist/linux/x86_64/GPUSPH
Success.
 * No devices specified, falling back to default (dev 0)...
GPUSPH version v3.0+50-c1c7ca3+custom
Release version without fastmath for compute capability 5.2
Compiled for problem "DamBreak3D"
Hashkey is 64bits long
NetworkManager: no complete thread safety, current level: 2
[Network] rank 0 (1/1), host popechad-MS-7885
 tot devs = 1 (1 * 1)
Initializing...
WARNING: dt 0.0003 bigger than 0.0001755 imposed by CFL conditions (sspeed: 0.00
01755, gravity: 0.0103605, viscosity: nan)
Using computed max neibs num 128
Problem calling set grid params
Influence radius / expected cell side   : 0.0234, 0.0234
 - World origin: -0.8 , -0.335 , -0.3
 - World size:   1.6 x 0.67 x 0.6
 - Cell size:    0.0235294 x 0.0239286 x 0.024
 - Grid size:    68 x 28 x 25 (47,600 cells)
 - Cell linearizazion: x,z,y
 - Dp:   0.009
 - R0:   0.009
Generating problem particles...
Allocating shared host buffers...
  allocated 11.31 MiB on host for 148,264 particles
Copying the particles to shared arrays...
---
Obstacle parts: 3724
Obstacle part mass:0.000771846
Fluid parts: 144540
Fluid part mass:0.000748164
---
 - device at index 0 has 148,264 particles assigned and offset 0
Starting workers...
2 CUDA devices detected
Estimated memory consumption: 372B/particle
Device idx 0 (CUDA: 0) allocated 0 B on host, 51.84 MiB on device
  assigned particles: 148,264; allocated: 148,264
GPUSPH: initialized
```

Figure 35: Initialization of GPUSPH

After GPUSPH has been initialized, the code starts the calculation for each particle. The run time to do the calculation depends upon the number of particles. While the calculation is being done, the following display can be seen in the terminal.



```
Performing first write...
Letting threads upload the subdomains...
Thread 0 uploading 148264 Position items (2.26 MiB) on device 0 from position 0
Thread 0 uploading 148264 Velocity items (2.26 MiB) on device 0 from position 0
Thread 0 uploading 148264 Info items (1.13 MiB) on device 0 from position 0
Thread 0 uploading 148264 Hash items (1.13 MiB) on device 0 from position 0
Entering the main simulation cycle
Simulation time t=0.000000e+00s, iteration=0, dt=3.000000e-04s, 148,264 parts (0, cum. 0 MIPPS), maxneibs 0
Simulation time t=1.507710e-02s, iteration=86, dt=1.715308e-04s, 148,264 parts (27, cum. 27 MIPPS), maxneibs 80
Simulation time t=3.012605e-02s, iteration=175, dt=1.669326e-04s, 148,264 parts (29, cum. 28 MIPPS), maxneibs 80
Simulation time t=4.517840e-02s, iteration=265, dt=1.677330e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 80
Simulation time t=6.032100e-02s, iteration=354, dt=1.718334e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 80
Simulation time t=7.542866e-02s, iteration=442, dt=1.702403e-04s, 148,264 parts (29, cum. 29 MIPPS), maxneibs 80
Simulation time t=9.055971e-02s, iteration=532, dt=1.643400e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 80
Simulation time t=1.056201e-01s, iteration=625, dt=1.605488e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 81
Simulation time t=1.206675e-01s, iteration=718, dt=1.653886e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 81
Simulation time t=1.357517e-01s, iteration=809, dt=1.666081e-04s, 148,264 parts (30, cum. 29 MIPPS), maxneibs 81
Simulation time t=1.508289e-01s, iteration=901, dt=1.637415e-04s, 148,264 parts (34, cum. 30 MIPPS), maxneibs 81
Simulation time t=1.658566e-01s, iteration=995, dt=1.553398e-04s, 148,264 parts (31, cum. 30 MIPPS), maxneibs 81
Simulation time t=1.810041e-01s, iteration=1,095, dt=1.539368e-04s, 148,264 parts (31, cum. 30 MIPPS), maxneibs 81
Simulation time t=1.961403e-01s, iteration=1,190, dt=1.616106e-04s, 148,264 parts (31, cum. 30 MIPPS), maxneibs 83
Simulation time t=2.112458e-01s, iteration=1,282, dt=1.622580e-04s, 148,264 parts (30, cum. 30 MIPPS), maxneibs 83
```

Figure 36: GPUSPH simulation calculation display

3.4.4.2 Code Output

After compiling GPUSPH, the output file will be saved in the tests folder under the GPUSPH directory. The output file will be in .VTU format. The tests folder contains the output files which are energy.txt, wavegage.txt, vtuinp.pvd and several other .VTU files.

The output files can be imported to ParaView for the visualization. The two .txt files, energy.txt and wavegage.txt, provides information about the initial velocity and energy of the fluid particles.

3.4.4.3 Comparison Model Simulation

The comparison model was then simulated in GPUSPH. The layout of the example can be seen in Figure 7 and Figure 8. The simulation was performed with various time steps, which allowed for different numbers of particles. The simulation below was performed with 150,048 particles. Figure 37 through Figure 39 show the comparison model output at different steps in time.

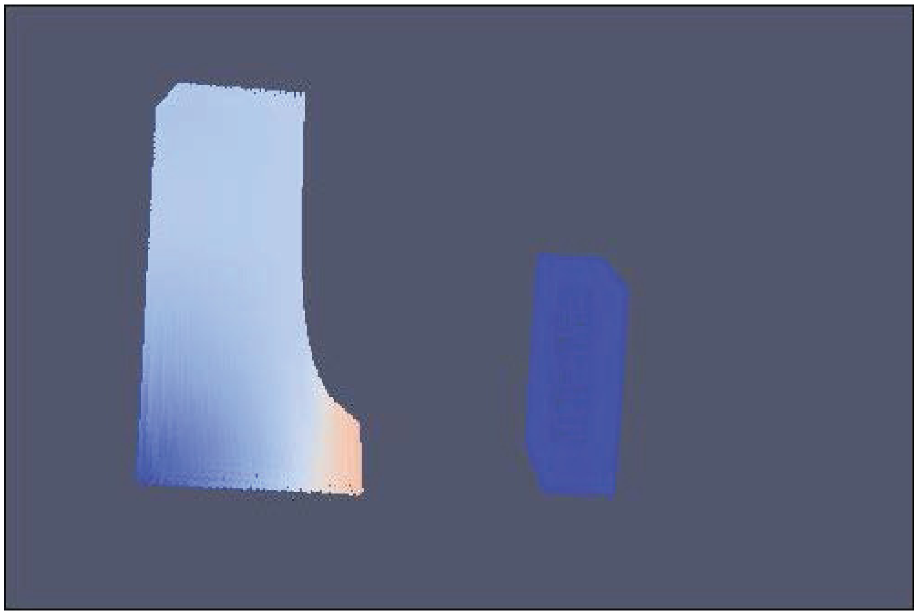


Figure 37: GPUSPH comparison model initial setup

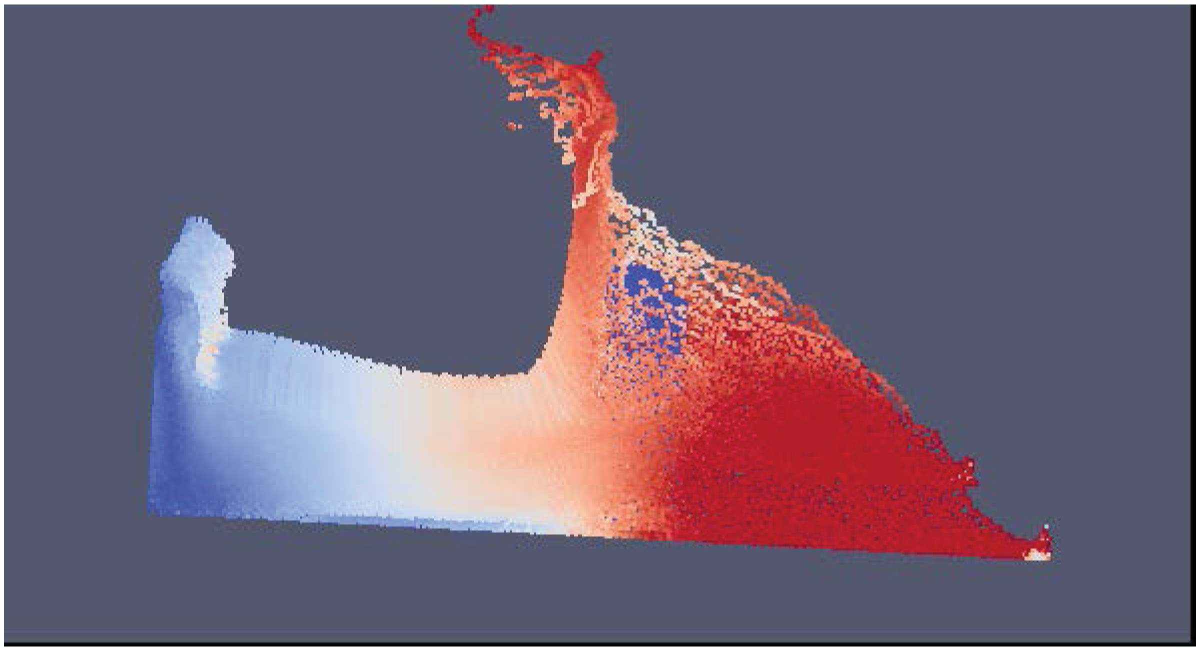


Figure 38: GPUSPH fluid particles colliding with wall

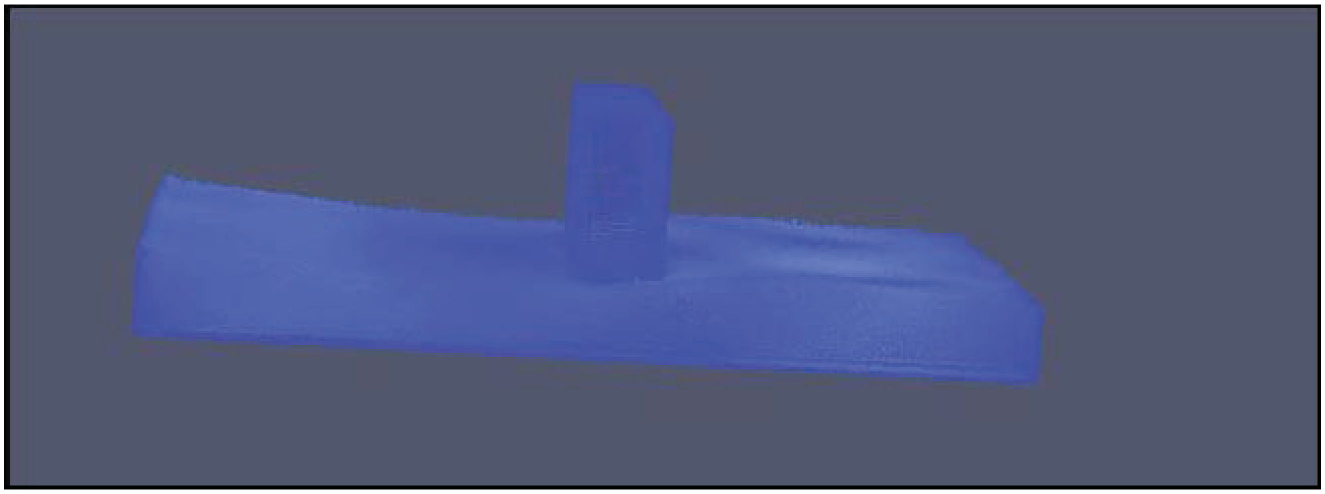


Figure 39: GPUSPH comparison model 100% complete

3.4.4.4 Parametric Study

Data was collected for each simulation with different numbers of particles. The time step and the related run time with number of particles are given in the table below. Different number of particles can be obtained by changing the value of the time step. When the time step decreases, the number of particles increases.

Table 6: GPUSPH runtime vs. number of particles results

Time step	Obstacle particles	Fluid particles	Total particles	Runtime (s)
0.05	3492	73080	76572	31
0.04	5794	144300	150094	67
0.03	10318	336600	346918	170
0.02	23756	1139850	1163606	800
0.01	95006	9059700	9154706	13000
0.009	116390	12382272	12498662	21000

From the obtained data, a plot was then created for number of particles and run time. The plot is shown in Figure 40.

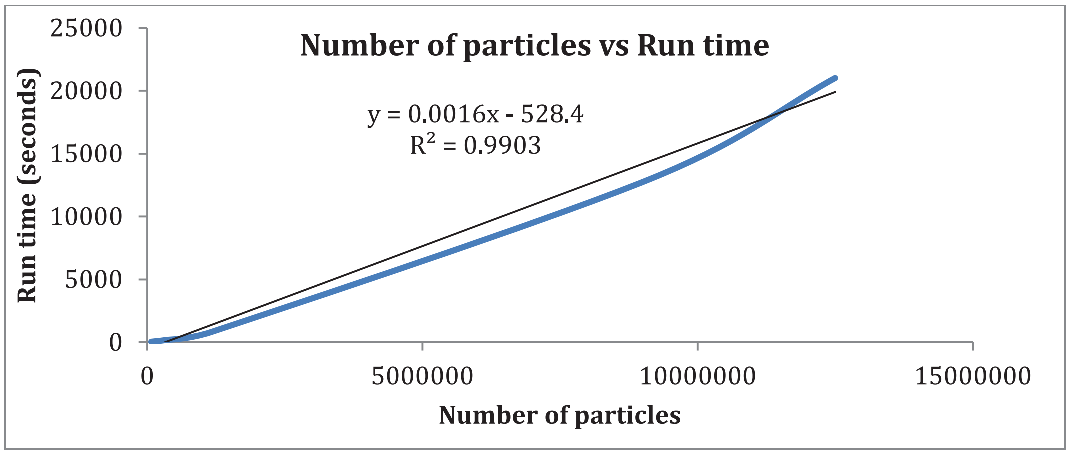


Figure 40: GPUSPH runtime vs. number of particles graph

The relationship was found to be linear. The equation was y = .001x-528.4, where y is the run time in seconds and x is number of particles.

3.4.5 Sibernetic

Sibernetic is an SPH software designed by OpenWorm. OpenWorm is an open source project dedicated to creating the first simulation of an entire organism [28]. The organism to be simulated is a C. elegans worm. Once fully developed Sibernetic will simulate the locomotion and body of the C. elegans. The code includes the ability to simulate magneto-hydrodynamics and elastic materials. It is under continued development by the OpenWorm project.

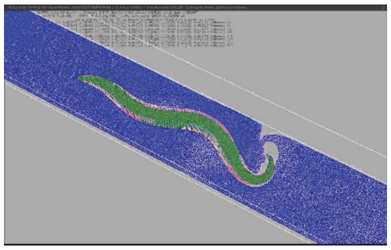


Figure 41: Sibernetic C. elegans simulation

Since the software remains under development, the usability and documentation creates a large hurdle to be overcome if the code were to be used for flood risk analysis. However, the contractile matter section of the code is of interest and may be useful in simulating elastic materials if needed.

3.4.5.1 Running the Code

To begin a simulation, the code is launched from the Linux command prompt. First, the OpenWorm file needs to be navigated to. From there, the program needs to be exported the first time with the following command.

```
export PYTHONPATH=$PYTHONPATH:'./src'
```

Now the code can be launched:

```
./Release/Sibernetic
```

It defaults to one of two pre-programmed simulations. Sibernetic provides no user interface. No user friendly method to control simulation parameters, change geometry, or insert basic CAD components currently exists. It is unclear as to how independent simulations must be executed. It is believed that changes in geometry and particle count must be done with a text file of unknown format and units. The source code is thoroughly commented but is difficult to navigate because code descriptions lack specific detail or a solid naming convention. As Sibernetic is an ongoing project, documentation is difficult to find and no general user manual exists.

In its current form a large amount of manipulation would need to be done to adapt the source code to the specified needs. As it is written in C++, a wide variety of users may have the ability to adapt the code. However, with no user manual or documentation adapting Sibernetic would be a difficult task.

3.4.5.2 Code Output

After launching the code, the desired simulation must be navigated to using the number pad. Once the desired simulation begins, it may be paused or resumed by pressing the space bar. It is currently unknown what format Sibernetic saves its output simulations. No simulations other than pre-programmed examples were successfully implemented.

3.4.5.3 Comparison Model Simulation

Since Sibernetic could not be edited to the comparison model, the two pre-programmed examples were analyzed. The first is an elastic cube dropping from a given height and dropping to the ground. The cube is deformed and then bounces as expected (Figure 42).

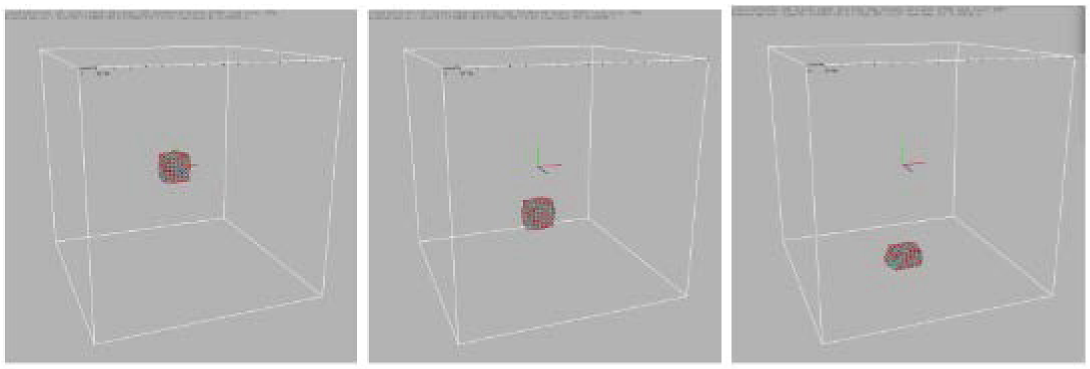


Figure 42: Sibernetic simulation of an elastic cube

The second simulation drops two cubes of fluid. The first drops onto an impermeable elastic membrane where bounces off is. The second cube is dropped onto a permeable elastic membrane (much like a trampoline). Some of the fluid leaks through, while some bounces (Figure 43).

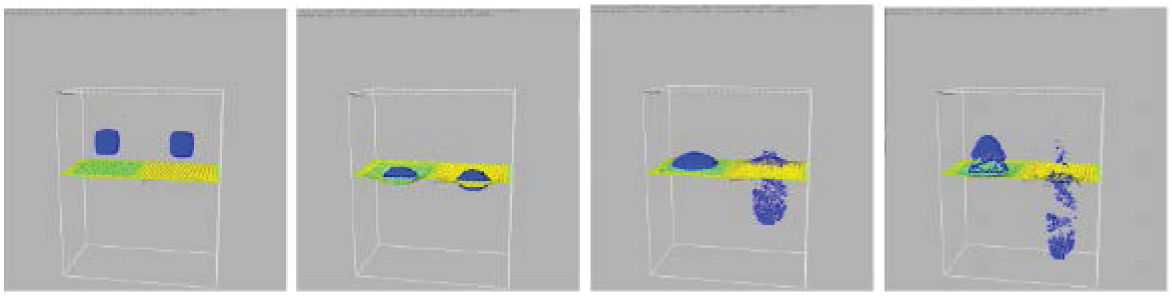


Figure 43: Sibernetic simulation of porous and non-porous elastic materials

Unfortunately, no independent simulations could be conducted to test the accuracy of Sibernetic.

3.4.5.4 Parametric Study

Sibernetic was not successful in conducting a parametric study. A parametric study was not done because independent simulations could not be conducted.

3.4.6 Fluid Sandbox

Fluid Sandbox is available to run on Windows or UNIX based operating systems and uses PhysX SDK as a physics engine. PhysX SDK is NVidia's free physics solution for game development. Some of its features include rigid body simulation, collision detection, particles, and cloth simulations.

3.4.6.1 Running the Code

Fluid Sandbox is an application and does not need to be installed. Once unpacking the file, the application is ready to be launched.

To create a simulation, an XML file needs to be generated in the scenarios folder. Here the pool boundary, the particle radius, distance factor, and objects are defined and placed in the simulation. Building objects into a Fluid Sandbox simulation is currently a challenge. All geometric objects must be made from a composition of cubes and spheres. The geometric composition would pose a challenge for simulating flows around complex objects. However, PhysX does provide better geometry options, using meshes to build complex shapes. Therefore, it might be possible to tweak the Fluid Sandbox code slightly to allow for a wider range of options.

Once the simulation has been launched the keypad may be used to toggle through a set of options. A set of essential controls is listed below.

- (L) Cycles through scenarios.
- (R) Restarts current simulation.
- (S) Changes rendering options.
- (H) Toggle GPU acceleration.
- (O) Pause simulation.
- (K) Toggle emitter.
- (W) Show or hide physical objects.
- (B) Show bounding box.
- (T) Toggle OSD.
- (Left Mouse) Rotate camera.
- (Right Mouse) Zoom.

3.4.6.2 Code Output

Fluid Sandbox does not currently output data files for analysis. To do so the source code would need to be modified. The only available output is high quality rendered videos of the designated simulation. An example can be seen in Figure 44.

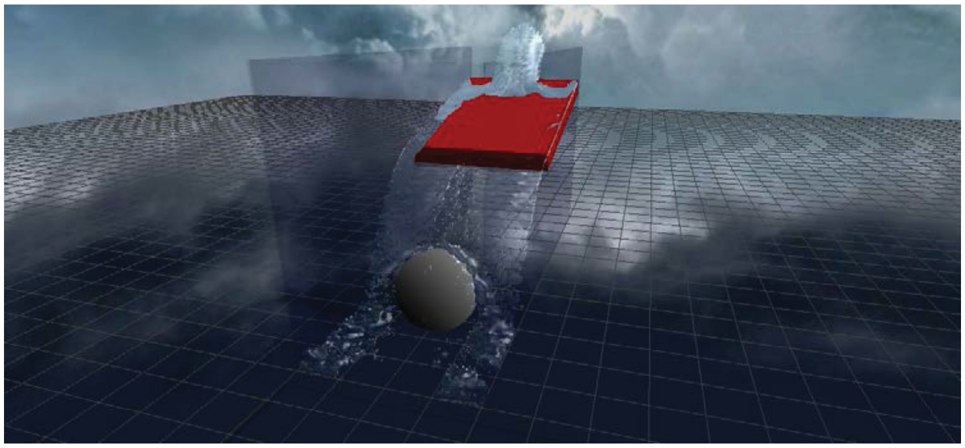


Figure 44: Fluid Sandbox high quality rendering

3.4.6.3 Comparison Model Simulation

The Model specified in Figure 7 and Figure 8 was built with Fluid Sandbox. The following figures depict the simulations results.

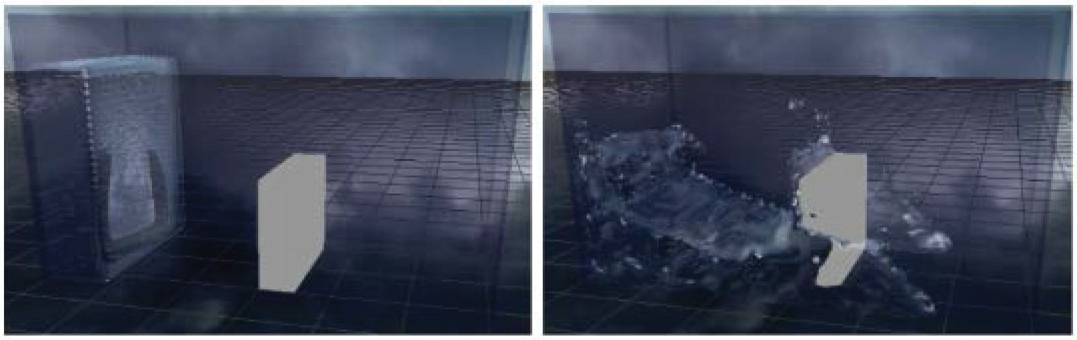


Figure 45: Fluid Sandbox comparison model initial setup

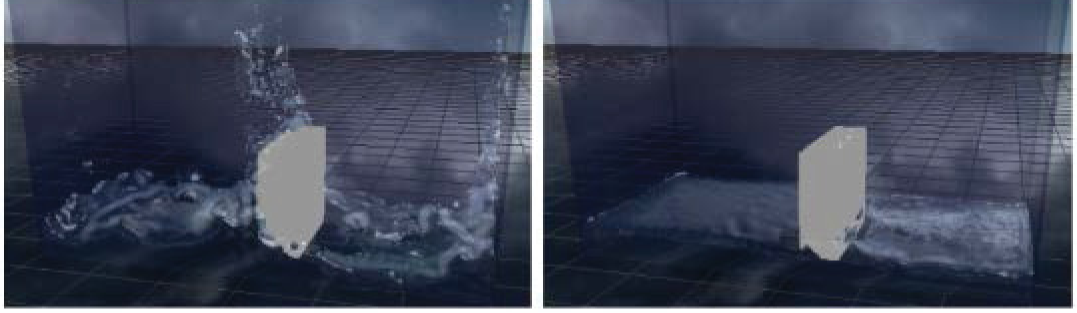


Figure 46: Fluid Sandbox comparison model swell returning to calm

Two additional render types are also available as seen in Figure 47. The first visualizes particle encapsulated in their effective distance. The second shows the center of each particle.

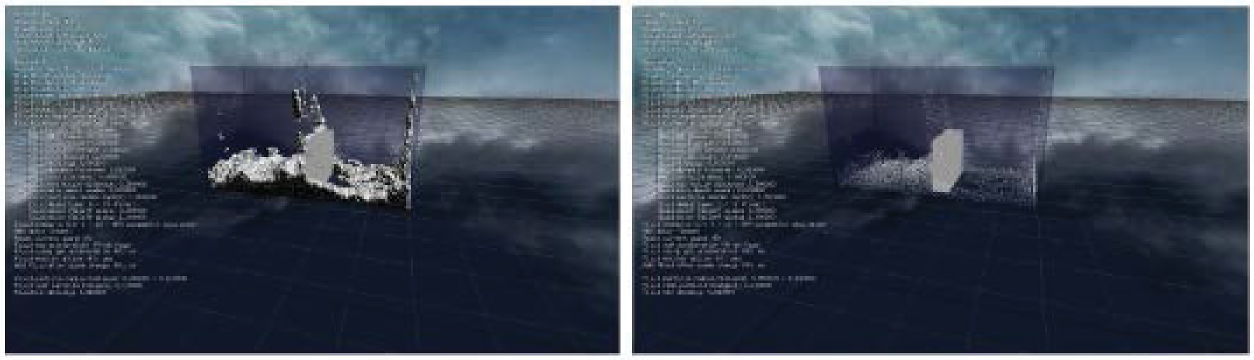


Figure 47: Fluid Sandbox other rendering options

3.4.6.4 Parametric Study

The parametric study was not able to be performed on GPU. Fluid Sandbox does have a GPU option but it is currently outdated from the latest NVidia software. Therefore, simulations for the parametric study were only performed using CPU. The following figure and table compare the example simulation run time to particle distance.

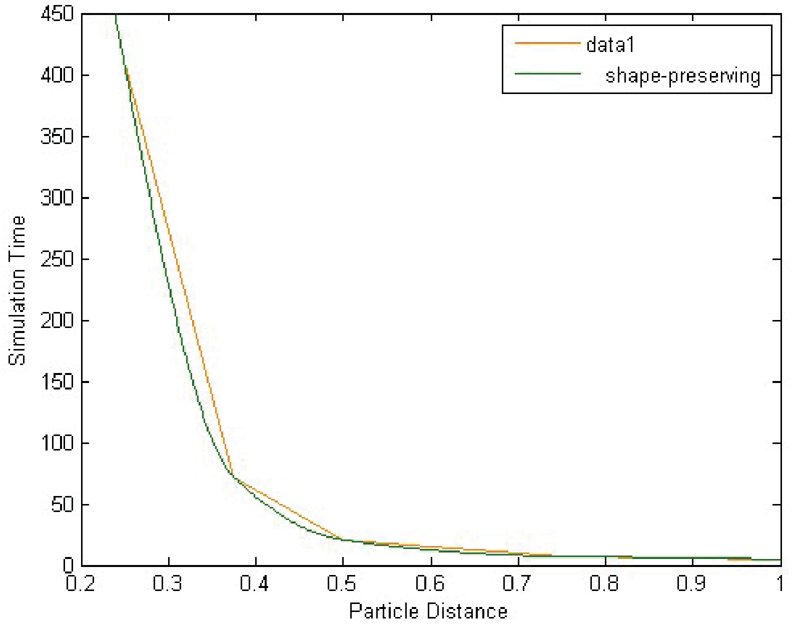


Figure 48: Fluid Sandbox runtime vs. number of particles graph

Since Fluid Sandbox is designed for gaming applications it updates the graphics continuously. Continuously updating the graphics is computationally expensive. Dropping the particle distance factor below 0.25 cause’s simulations errors.

Table 7: Fluid Sandbox parametric study results

Distance between particles	Particle radius	Fluid particles	GPU Runtime	CPU Runtime
1	0.1	7569	N/A	4.8
0.75	0.1	19773	N/A	6.3
0.75	0.1	19773	N/A	7.1
0.5	0.1	66119	N/A	20.6
0.375	0.1		N/A	71.8
0.25	0.1	382239	N/A	406.6
0.1	0.1	N/A	N/A	error

3.4.7 NDSPMHD

NDSPMHD is an SPH code geared towards performing gas dynamics and astrophysics problems [20]. It also simulates magneto-hydrodynamics and situations with dust. To create visualizations it needs its sister software, SPLASH [29].

3.4.7.1 Running the Code

To run NDSPMHD, a UNIX based operating system is needed. Once the file is downloaded and unpacked, the FORTAN compiler must be defined as well as how many dimensions the simulation will run in. In order to create a 2D simulation, the following command must be executed.

```
make SYSTEM=gfortran 2D
```

Once the above command is executed, simulations can be created. To start a simulation, a directory must be made. To make a directory and enter it (let's call it test) the following command can be deployed.

```
mkdir test cd test
```

Next, a make file needs to be created for in the directory and the location of the code specified. The following commands should be entered in succession.

```
~/Path_to_software/ndspmhd/scripts/writemake.sh 2D > Makefile export
NDSPMHD_DIR=~/Path_to_software/ndspmhd
make
```

After the above command is executed, an input file needs to be created. To continue with the naming convention, it will be called "test".

```
./2DSPMHD test.in
```

The above command will create an input file with default values. The input file will define the simulations parameters. If left as is, a cubic lattice of particles will be created with no gravity, and thus static.

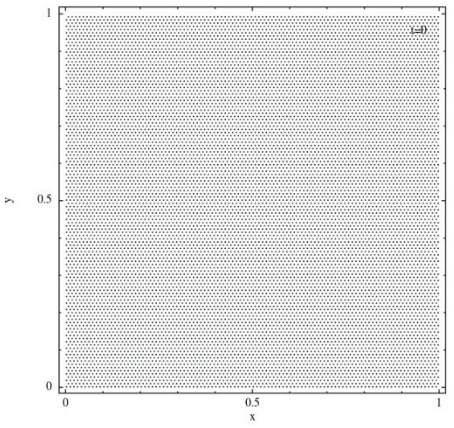


Figure 49: NDSPMHD default configuration

The final step is to create a file which defines the particle set up and initial conditions. To do so, copy a stock *.f90 file included with the code into the current directory. Continuing with the naming convention, let's call it "testing.f90". Once the file has been created execute the following command.

```
~/Path_to_software/ndspmhd/scripts/writemake.sh 2D testing.f90  > Makefile
```

Finally, the simulation may be initiated as follows. If previous simulations have been conducted, "make clean" and "make" commands should also be given prior to launching the code.

```
./2DSPMHD  test.in
```

The results will be saved to the directory that was made for the simulation.

3.4.7.2 Code Output

NDSPMHD saved the results of the simulation as *.dat files in the directory that the simulation was conducted in. To view the results SPLASH needs to be called from the same directory as the results. If SPLASH has not been compiled, it can be done by typing in the following command.

```
nsplash  test_0*.dat
```

SPLASH will ask what outputs are desired. To view the particles positions, choose y for the y axis and x for the x axis. Choose 0 for any other options and specify the output file types as desired.

The following results are given for the stock simulation with the gravitational constant set to 2. The result is a collapsing gas lattice.

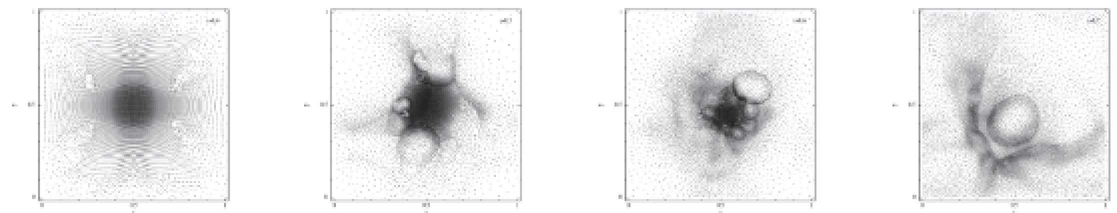


Figure 50: NDSPMHD heavy gas collapsing under its own gravity

3.4.7.3 Comparison Model Simulation

NDSPMHD was not able to run the selected comparison model. The author of the program was contacted about the possibility of running such a simulation. It was discovered that the software is not prepared to run simulations with odd boundaries or liquid water.

3.4.7.4 *Parametric Study*

As stated in the previous section, NDSPMHD was not able to run the comparison model. As a point of comparison, a parametric study was conducted using the simulation shown in Figure 49. The purpose is only to illustrate the function of run time versus particle separation for an arbitrary simulation.

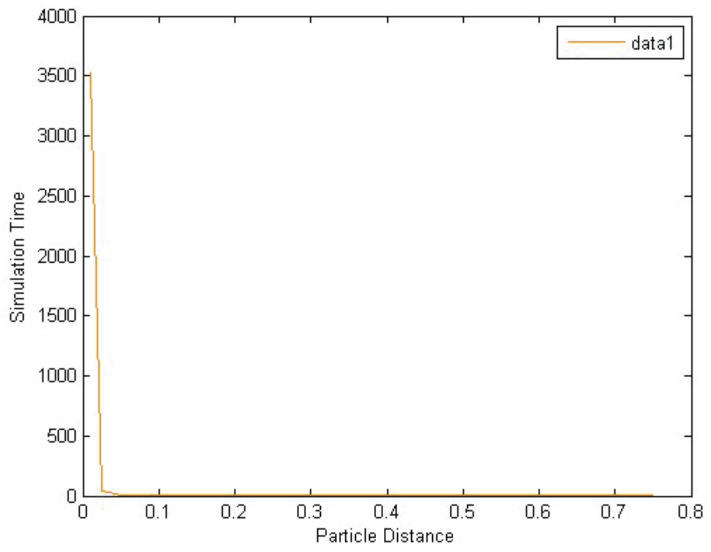


Figure 51: NDSPMHD runtime vs. number of particles graph

Table 8: NDSPMHD runtime vs. number of particles results

Distance between particles	Fluid particles	GPU Runtime	CPU Runtime
0.1	100	N/A	0.45
0.075	205	N/A	1.1
0.05	441	N/A	3.3
0.025	1681	N/A	45
0.01	10201	N/A	3522.9

NDSPMHD was not tested on GPU and its CPU runtime had a very sharp curve.

3.4.8 Other Codes Investigated

The following codes were investigated in addition to those previously mentioned. However, these codes manifested some kind of issue that made them undesirable for further investigation.

3.4.8.1 *PySPH*

PySPH is an open source SPH code. The code was developed by the Department of Aerospace Engineering, IIT Bombay [12]. The code is implemented in Python. PySPH provides documentation about installation and the code framework. It provides multiple examples that allow a first time user to become familiar with the code. However, the issue with PySPH came when trying to install. When installing PySPH, an error kept occurring. After multiple attempts at fixing the error, the support staff was e-mailed. A return e-mail was received stating that the e-mail address is no longer active. At this point, PySPH was considered undesirable.

3.4.8.2 SPH Distributed Fluid Simulator

SPH Distributed Fluid Simulator is a particle-based distributed fluid simulator application designed for the Hewlett-Packard Scalable Visualization Array [16]. It was developed by the Department of Control Engineering and Information Technology at Budapest University of Technology and Economics. The code is implemented in C++ and has the ability to run using CPUs or GPUs. The issue with SPH Distributed Fluid Simulator came when trying to install. SPH Distributed Fluid Simulator required a Linux based operating system. When trying to install, SPH Distributed Fluid Simulator needed specific RPM Package Manager (RPM) files. However, the Linux operating system that it was trying to be installed on was Ubuntu, a Debian based Linux operating system and not an RPM based system. There are ways to install RPM files on a Debian based system, but all attempts failed. Since the code was not able to be installed, it was deemed undesirable for further investigation.

3.4.8.3 COULWAVE

COULWAVE is a free surface wave model [17]. It was developed by Patrick Lynett. The code is implemented in FORTRAN. The issue with COULWAVE is with its inability for 3D rendering. On the codes main website, there is a 3D image showing water and rigid bodies. However, on review of the user manual all of the provided examples are either 1D or 2D. Determining if COULWAVE can perform 3D rendering and how to do it would create additional work. Since it could not be determined if the code can do 3D rendering, it was deemed undesirable for further investigation.

3.4.8.4 GADGET

GADGET is a freely available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory [19]. It was developed by Max-Planck-Institute for Astrophysics. The code is implemented in standard ANSI C. Installation of GADGET was not successful. Since the code is an astrophysics code and it could not be built, it was deemed undesirable for further investigation.

3.4.8.5 AQUAgpusph

AQUAgpusph is a free SPH solver developed by the CEHINAV group [15]. The code is implemented in C++. The issue with AQUAgpusph came when running the code. When attempting to run the provided examples, an error kept occurring. The error was a segmentation fault which occurs with errors in pointers. A segmentation fault means that in order to find the error, the source code must be debugged to find the pointer causing the problem. There was an attempt to contact the developer of AQUAgpusph, but there has not been any response. Since the code was not able to run the provided examples and there has not been a response from the group member, it was deemed undesirable for further investigation.

3.5 Comparison of Investigated Codes

The following table shows a comparison of the investigated codes. It compares the largest number of particles that were simulated as well as the runtime associated with that number of particles.

Table 9: Comparison of code results

Code	Used comparison model	Number of particles	Runtime
Fluids v.3	No	8,388,608	4.433 sec/frame
DualSPHysics – GPU	Yes	106,732	708 sec
DualSPHysics - CPU	Yes	714,662	104,019 sec
Fluidix	Yes	1,000,000	740.5 sec
GPUSPH	Yes	12,498,662	21,000 sec
Fluid Sandbox	Yes	382,239	406.6 sec
NDSPMHD	No	10,201	3,523 sec

As it can be seen in Table 9, GPUSPH was able to simulate the largest number of particles. Most of the other codes can simulate particles in the range of hundreds of thousands. Some of the codes gave an error when the number of particles got too large.

3.5.1 Codes for Further Investigation

After researching and running the above SPH codes, two of the above codes were determined to have the highest probability to be modified for the incorporation of component flooding failure analysis. The codes chosen for further investigation are:

- DualSPHysics
- GPUSPH

3.5.2 DualSPHysics

DualSPHysics was easy to run, easy to create a model, and provides detailed output information. It incorporates SPH theory and has the option to run using different smoothing kernels. Some of the biggest advantages to DualSPHysics are that the user can import 3D models, it is completely open source, and that it is still being developed and used. One concern with DualSPHysics is the error that occurred when simulating a larger amount of particles on GPU.

3.5.3 GPUSPH

GPUSPH is an open source code which can be easily downloaded to use. The key advantages of GPUSPH are the availably of the various source codes and the use of multiple GPU cards. The use of multiple GPU cards makes the computing fast. The computing capability of the Xi computer which uses two GeForce Titan X cards, is 5.2. Additionally, GPUSPH can handle more than 32 million particles, which is a fairly high number. The output of the computation is saved with the date and time which makes it very easy to locate the result of the simulation. Also, the simulation results can be easily imported to ParaView, which provides a great simulation platform. Overall, GPUSPH seems to be a good choice for further investigation.

4. Example External Events Analyses

4.1 Flooding

Currently, the flooding analysis has focused on using the SPH-based physics tools representing water behavior to mimic different kinds of floods. In this section, we provide (through graphics), examples of the types of flooding calculations that can be performed with capable SPH-based tools. The examples in this section were all calculated with the NEUTRINO software. The first example, in Figure 52, we show how particle based approaches can be used to represent a dam break. In the upper left part of the figure, the initial water has left the dam (not shown, but represented in the left part of the graphic). As the scenario progresses, the water continues to leave the impacted dam and inundates the plant site. In Figure 53, we show how we can track specific attributes such as wave height during a simulation for a tsunami. The physical properties of the virtual water can be tracked during the scenario simulation.

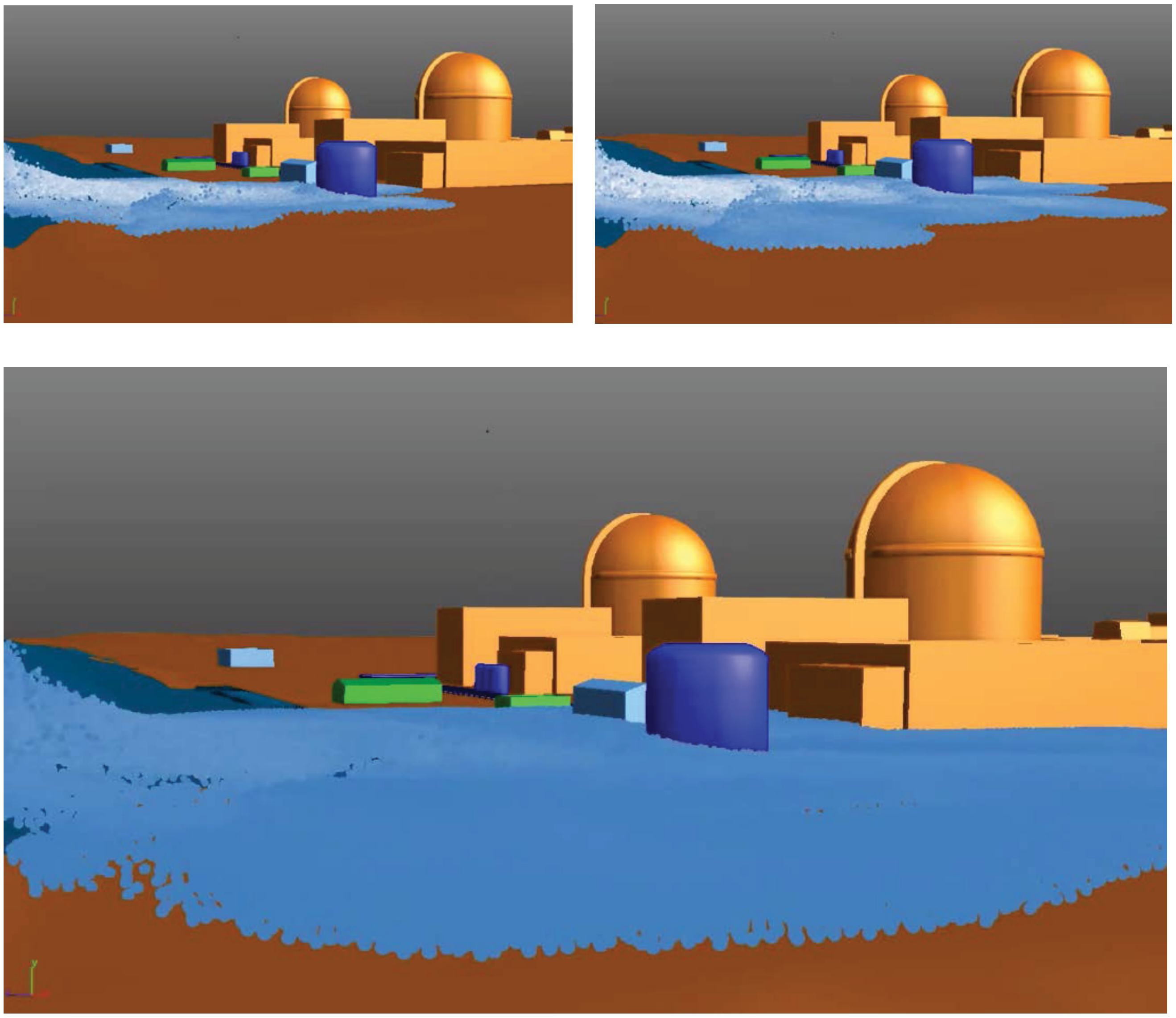
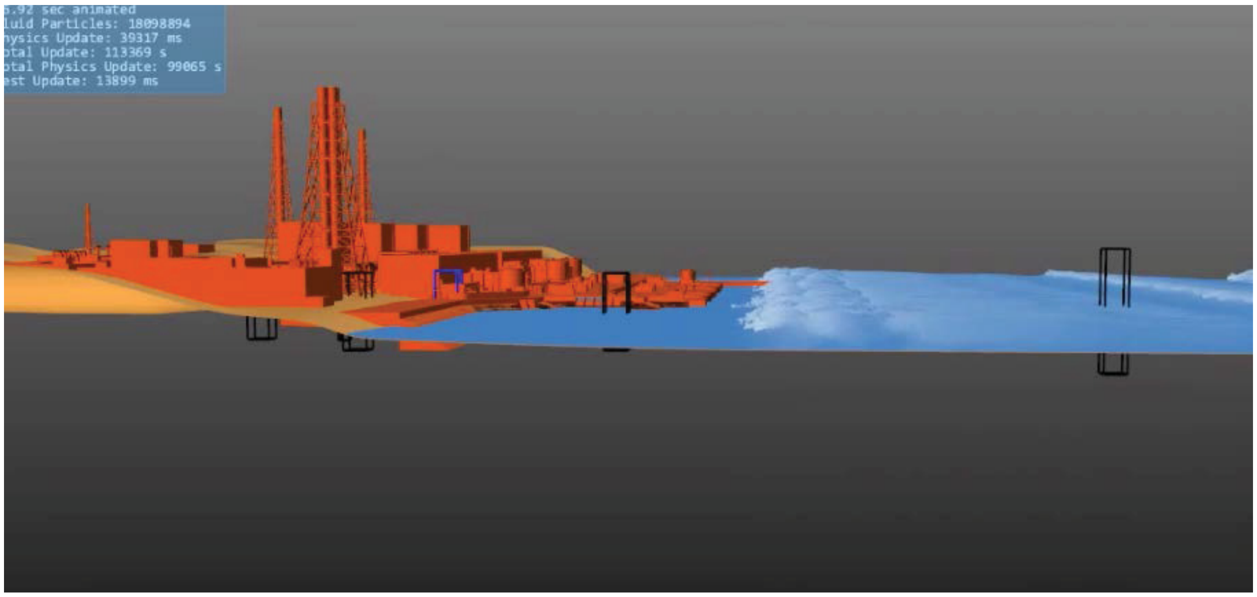
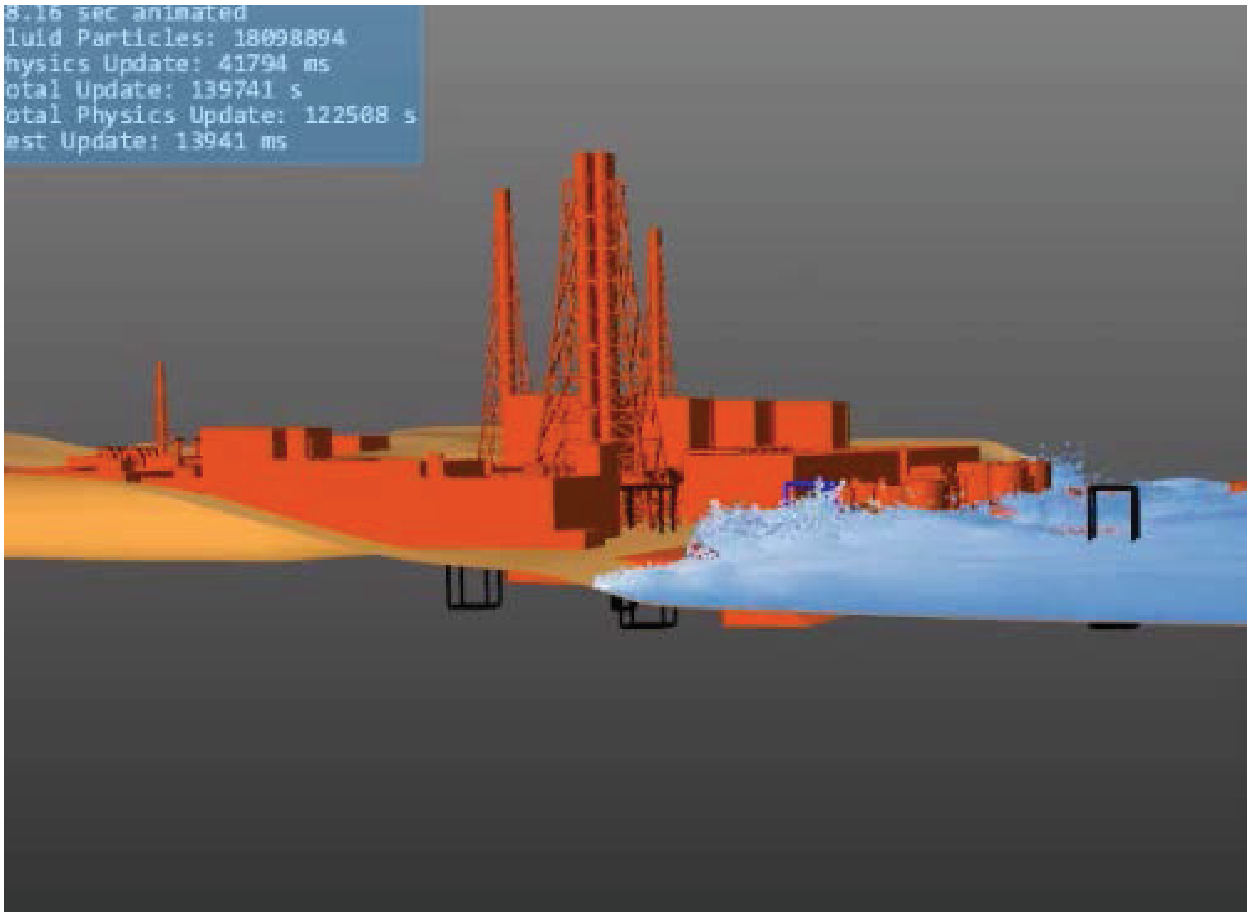


Figure 52. Example flooding analysis showing progression of a dam break flood.



```
5.92 sec animated
luid Particles: 18098894
hysics Update: 39317 ms
otal Update: 113369 s
otal Physics Update: 99065 s
est Update: 13899 ms
```



```
8.16 sec animated
luid Particles: 18098894
hysics Update: 41794 ms
otal Update: 139741 s
otal Physics Update: 122508 s
est Update: 13941 ms
```

Figure 53. Example of tracking wave height during a tsunamic flooding scenario.

In Figure 54, we illustrate one of the strengths of the physics-based simulation wherein we can keep track of observable quantities such as force (here measured in Newton) for various objects found within the 3D model. In this figure, the force is being tracked on a specific building in the 3D model. This physical information can be used to provide advanced simulation approaches where objects deform or fail during the simulation due to loads being imposed on the objects. For example, it may be that a door or pipe fails – this failure then might lead to internal flooding (see Figure 55) where water then impacts other components. In the internal flooding scenario, the water will continue to exit the failed pipe until its water supply is depleted or a valve is closed during the simulation. Also, as can be seen in the figure, we can track where the water goes inside the room – components that are being inundated are changed to "red" in order to visually identify possible component failures.

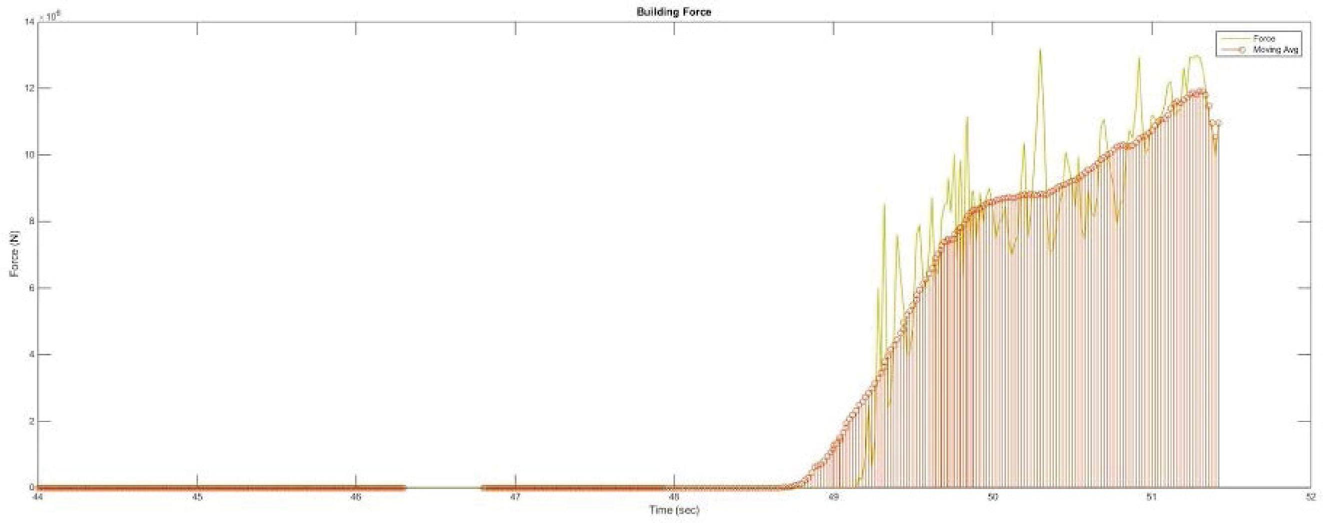


Figure 54. Force measure on specific objects calculated by the SPH-based tool.

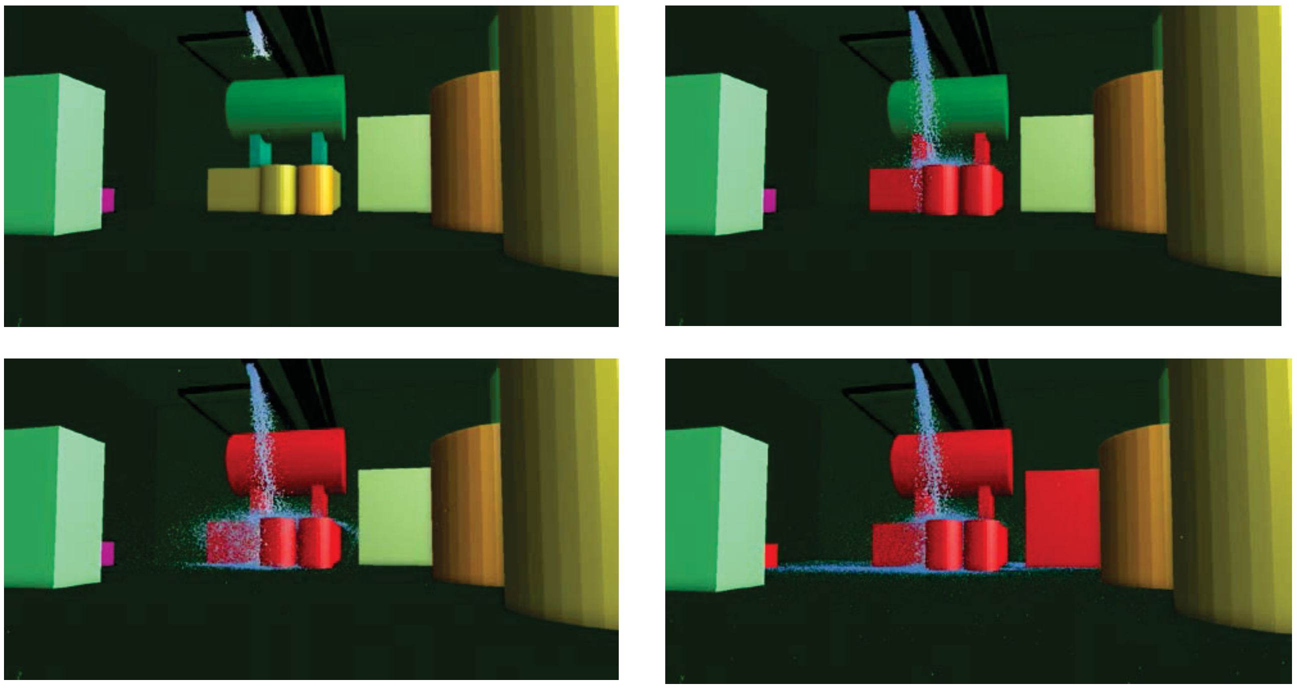


Figure 55. SPH-based simulation where a pipe break leads to internal flooding (components that are impacted by water a color coded in red) that shows the progression of water as a function of time.

4.2 Seismic

Early SPRA results focus is on an investigation of non-linear soil-structure interaction (NLSSI) into the SPRA calculation process when calculating in-structure response at the area of interest. Two specific nonlinear effects included are localized soil nonlinearity and gapping and sliding. Other NLSSI effects are not included in the calculation described in this report. The commercial software program, LS-DYNA, is used for the NLSSI analyses. The results presented document initial model runs in the linear and nonlinear analysis process.

For this analysis, we use a generic Nuclear Power Plant (NPP) structure and a generic system and perform linear and nonlinear probabilistic response analysis using:

- Linear SSI (CLASSI, frequency domain)
- Nonlinear SSI (LS-DYNA, time domain)

Nonlinear SSI results were generated by developing responses at three ground motion scale factors (note the linear SSI used just one and then assumes that ISRS scales linearly with increasing ground motion). The same fragilities calculated for the linear analysis are used. Results are then generated at each ground motion level. In both the linear and nonlinear SSI the probability of system failure is computed by convolving system conditional failure probability with the seismic hazard.

The study considers a generic NPP reinforced concrete reactor building and representative plant safety system. Simplifications in the seismic hazard, structure model, soil properties, and plant system will be introduced to limit the analytical effort in this initial study. Additional details on this study are given in INL/EXT-14-33222. Complexity can be added in subsequent phases of this project.

The selected representative NPP structure is a pressurized water reactor building example. It consists of a pre-stressed concrete containment structure and reinforced concrete internal structure. The structure and its stick model representation are shown in Figure 55.

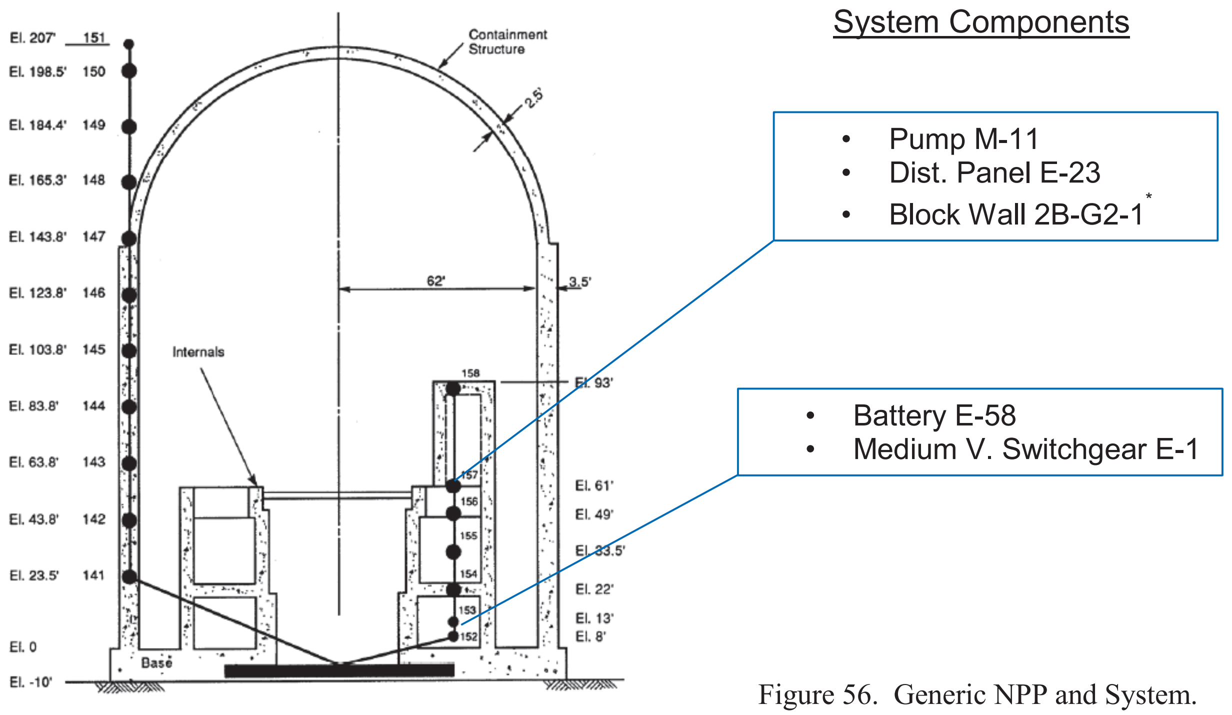


Figure 56. Generic NPP and System.

Probabilistic response analysis was performed using the Latin Hypercube Sampling (LHS) approach for thirty simulations and models run in the linear computer program CLASSI. Preliminary fragility results based on the linear (traditional) SPRA analysis are provided in Table 10.

Table 10. Linear SSI Fragilities.

Component	Floor	A_m	β_c	HCLPF
Pump 670-M-11	EL 61’	2.70g	0.45	0.95g
Battery 670-E-59	EL 22’	1.14g	0.28	0.59g
Dist. Panel 670-E-23	EL 61’	1.60g	0.59	0.40g
Block Wall 2B-G2-1	EL 61’	0.60g	0.28	0.31g
Switchgear 670-E-1	EL 22’	1.90g	0.47	0.64g

The nonlinear analysis results (for the layout shown in Figure 56) show that at low levels of ground motion the linear and nonlinear models produce similar results (Figure 57).

Preliminary NLSSI analysis demonstrates that a functional NLSSI model has been developed that includes 1) local soil nonlinearities at the foundation and 2) geometric nonlinearities. This NLSSI model will be run multiple times at increasing levels of ground motion to generate in-structure response spectra that will be input into the advanced SPRA calculations. Comparison of traditional SPRA and advanced SPRA results will be provided in future work.

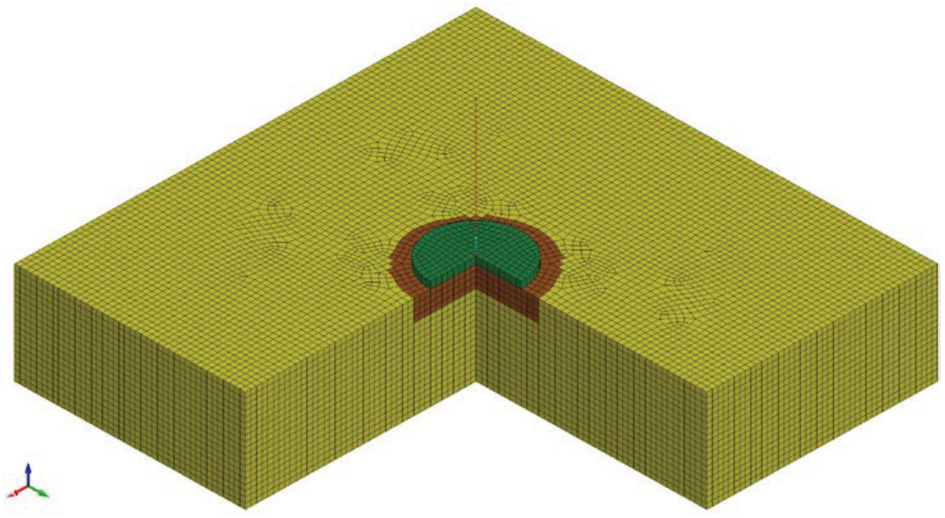
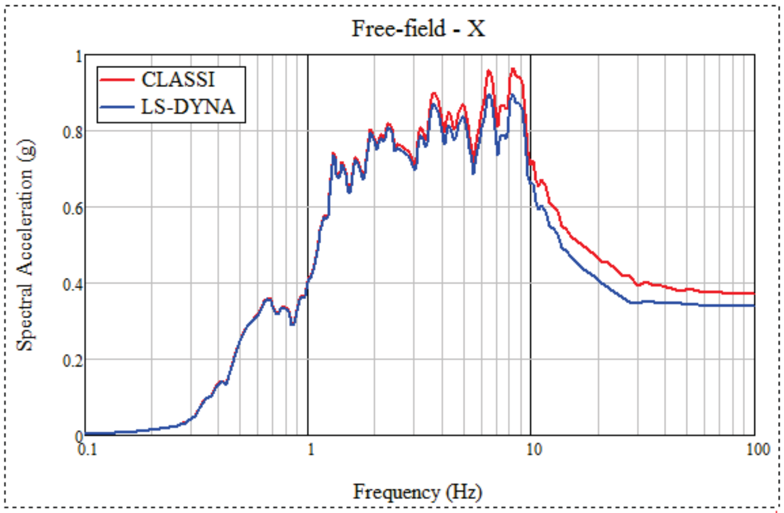
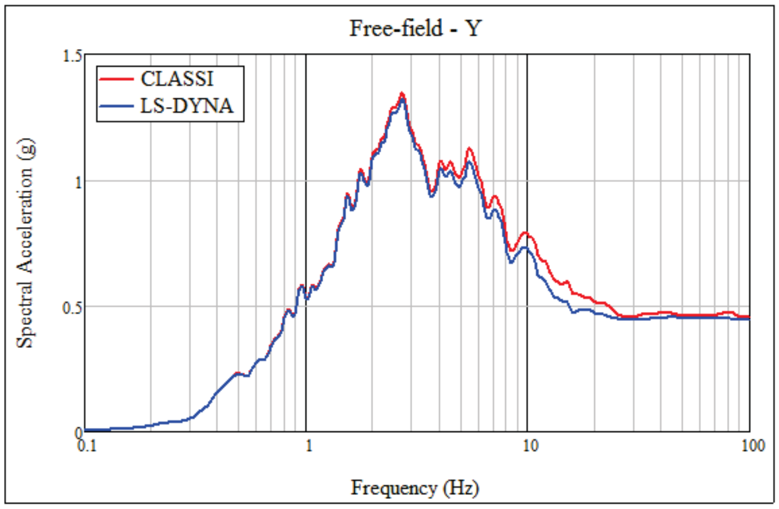


Figure 57. Section View of the Nonlinear Model Illustrating the Nonlinear Soil (Brown), Linear Soil (Yellow) and Basemat (Green).

Work will continue in FY16 on further exploring the calculation aspects of IA2 for both external hazards of flooding and seismic initiating events.





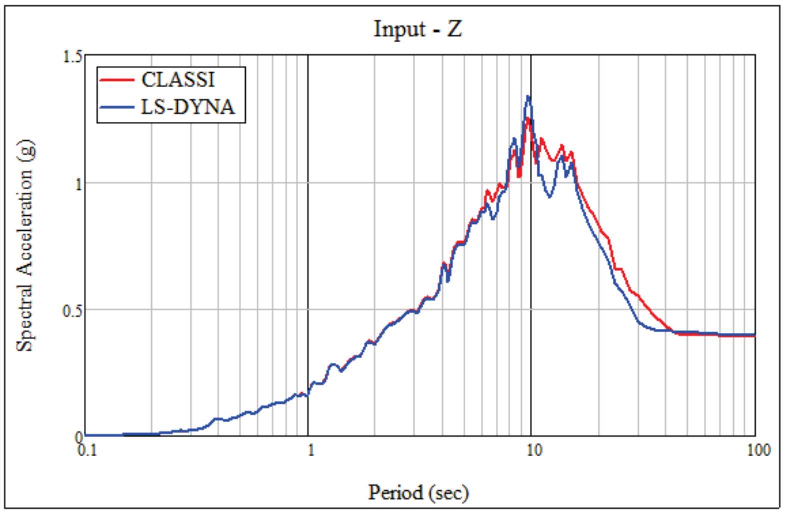


Figure 58. SPRA results comparing the acceleration estimated using linear (CLASSI) and non-linear (LS-DYNA) SSI calculations (results shown for the X, Y, and Z directions).

5. REFERENCES

C. Smith, C. Rabiti, R. Martineau, “Risk-Informed Safety Margins Characterization (RISMC) Pathway Technical Program Plan,” INL-EXT-11-22977, Revision 2, September 2014

J. Coleman, C. Rabiti, C. Smith, R. Szilard, R. Youngblood and H. Zhang, "Advanced Safety Analysis Program Strategy and Use-Case Analysis," Idaho National Laboratory, INL-EXT-13-30210-DRAFT, February 2014.

C. Smith, R. Szilard and K. McCarthy, "Advanced Safety Analysis Program Objectives and Value Proposition," Idaho National Laboratory, INL-EXT-13-30460-DRAFT, February 2014.

R. Szilard and C. Smith, “Advanced Reactor Safety Program - Stakeholder Interaction and Feedback," Idaho National Laboratory, INL-EXT-14-32928, August 2014.

R. H. Szilard, C. L. Smith, R. Youngblood, “RISMC Advanced Safety Analysis Project Plan – FY 2015 – FY 2019,” INL-EXT-14-33186, September 2014.

S. Hess, "Risk Informed Safety Margin Characterization for Effective Long Term Nuclear Power Plant Safety Management," in *3rd International Conference on NPP Life Management (PLIM) for Long Term Operations (LTO)*, Salt Lake City, 2012.

Draft NEI White Paper, NEI-NRC Risk-Informed Steering Committee (RISC), January 2015.

C. Smith, et al., “Risk Informed Safety Margin Characterization (RISMC) Advanced Test Reactor Demonstration Case Study,” Idaho National Laboratory, INL/EXT-12-27015, August 2012.

D. Gaston, G. Hansen and C. Newman, “MOOSE: A Parallel Computational Framework for Coupled Systems for Nonlinear Equations,” in *International Conference on Mathematics, Computational Methods, and Reactor Physics*, Saratoga Springs, NY, 2009.

D. Gaston and e. al, "Physics-based Multiscale Coupling for Full Core Nuclear Reactor Simulation," *Annals of Nuclear Energy,* (submitted for publication).

R. Youngblood and C. Smith, “Technical Approach and Results from the Fuels Pathway on an Alternative Selection Case Study," Idaho National Laboratory, INL-EXT-13-30195, 2013.

S. Prescott, C. Smith, T. Koonce, "Case Study for Enhanced Accident Tolerant Design Changes," Idaho National Laboratory, INL-EXT-14-32355, June 2014.

C. Smith, D. Mandelli, S. Prescott, A. Alfonsi, C. Rabiti, J. Cogliati, R. Kinoshita, "Analysis of Pressurized Water Reactor Station Blackout Caused by External Flooding Using the RISMC Toolkit," Idaho National Laboratory, INL-EXT-14-32906, August 2014.

U.S. Nuclear Regulatory Commission, "Recommended Actions to be Taken Without Delay - The Near-Term Task Force," 2011.

H. Gougar, “Enhanced Severe Accident Analysis for Prevention Technical Program Plan," Idaho National Laboratory, INL-EXT-14-33228, September 2014.

C. Smith, D. Schwieder, C. Phelan, A. Bui and P. Bayless, "Risk Informed Safety Margin Characterization (RISMC) Advanced Test Reactor Demonstration Case Study," Idaho National Laboratory, 2012.

EPRI, "Seismic Evaluation Guidance: Augmented Approach for the Resolution of Fukushima Near-Term Task Force Recommendation 2.1 - Seismic," 2013.

C. Stoots and et al, "Verification and Validation Strategy for LWRS Tools," 2012.

EPRI, "2014 Research Portfolio - Risk and Safety Management," 2013. [Online]. Available: http://portfolio.epri.com/ProgramTab.aspx?sId=NUC&rId=275&pId=7807&suId=7811.

Pagani, L.P., Apostolakis, G.E., Hejzlar, P., 2005. The impact of uncertainties on the performance of passive systems. Nucl. Technol. 149 (2), 129–140.

US NRC, "Closure Plan for Reevaluation of Flooding Hazards for Operating Nuclear Power Plants," COMSECY-15-0019, June 2015.

A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "Raven as a tool for dynamic probabilistic risk assessment: Software overview," in Proceeding of M&C2013 International Topical Meeting on Mathematics and Computation, CD-ROM, American Nuclear Society, LaGrange Park, IL (2013).

D. Mandelli, C. Smith, Z. Ma, T. Riley, J. Nielsen, A. Alfonsi, C. Rabiti, J. Cogliati, " Risk-Informed Safety Margin Characterization Methods Development Work," INL/EXT-14-33191, INL, September 2014.

A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, R. Kinoshita, A. Naviglio, "Dynamic Event Tree Analysis Through RAVEN", International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA 2013), September 22-26, Columbia, SC, USA, (2013).

A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, and R. Kinoshita, "RAVEN: Development of the adaptive dynamic event tree approach," Tech. Rep. INL/MIS-14-33246, Idaho National Laboratory (INL), (2014).

R. Youngblood and C. Smith, "Technical Approach and Results from the Fuels Pathway on an Alternative Selection Case Study," INL/EXT-13-30195, INL, September 2013.

Bathe, K.-J. (1996). "Finite Element Procedures." Prentice Hall, Upper Saddle River, New Jersey.

Bolisetti, C., and Whittaker, A. S. (2015). "Site Response, Soil-Structure Interaction and Structure-Soil-Structure Interaction for Performance Assessment of Buildings and Nuclear Structures." MCEER-15-0002, Multidisciplinary Center for Earthquake Engineering Research (MCEER), University at Buffalo, The State University of New York, Buffalo, NY.

Bolisetti, C., Whittaker, A. S., and Coleman, J. L. (2015). "Frequency- and Time-Domain Methods in Soil-Structure Interaction Analysis." *Nuclear Engineering and Design*, Under Review.

Coleman, J. (2014), "Demonstration of NonLinear Seismic Soil Structure Interaction and Applicability to New System Fragility Seismic Curves," INL/EXT-14-33222, Idaho National Laboratory, Idaho Falls, Idaho.

Computers and Structures Inc. (2011). Computer Program: SAP2000 - Structural Analysis Program, Version 11.0.0. Computers and Structures, Inc., Berkeley, California.

Livermore Software Technology Corporation (LSTC). (2009). "LS-DYNA Theory Manual." Livermore, California.

Livermore Software Technology Corporation (LSTC). (2013). "LS-DYNA Keyword User's Manual - Version R 7.0." Livermore, California.

Lysmer, J., Ostadan, F., and Chin, C. (1999). Computer Program: SASSI2000 - A System for Analysis of Soil-Structure Interaction. University of California, Berkeley, California.

Ostadan, F. (2006). "SASSI2000: A System for Analysis of Soil Structure Interaction - User's Manual." University of California, Berkeley, California.

Spears, R., and Coleman, J. (2014). "Nonlinear Time Domain Soil-Structure Interaction Methodology Development." INL/EXT-14-33126, Idaho National Laboratory, Idaho Falls, Idaho.

Bolisetti, C., and Coleman, J. (2015). "Advanced Seismic Soil Structure Modeling.: INL/EXT-15-35687, Idaho National Laboratory, Idaho Falls, Idaho.

Willford, M., Sturt, R., Huang, Y., Almufti, I., and Duan, X. (2010). "Recent Advances in Nonlinear Soil-Structure Interaction Analysis using LS-DYNA." *Proceedings of the NEA-SSI Workshop*, October 6-8, Ottawa, Canada.

S. Prescott, R. Sampath, C. Smith, T. Koonce, “Advanced Small Modular Reactors – Prototype Development Capabilities of 3D Spatial Interactions and Failures During Scenario Simulation,” INL/EXT-14-33211, September 2014.

Akinci (2012). "Versatile Rigid-Fluid Coupling for Incompressible SPH," http://www.nadir.tk/research. *ACM TOG, SIGGRAPH proceedings*.

WinDAM, http://www.usbr.gov/ssle/damsafety/risk/BestPractices/27-SeismicEmbankmentPP20121121.pdf

US NRC, "A Proposed Risk Management Regulatory Framework", NUREG-2150, April 2012.