SAND2008-6285P

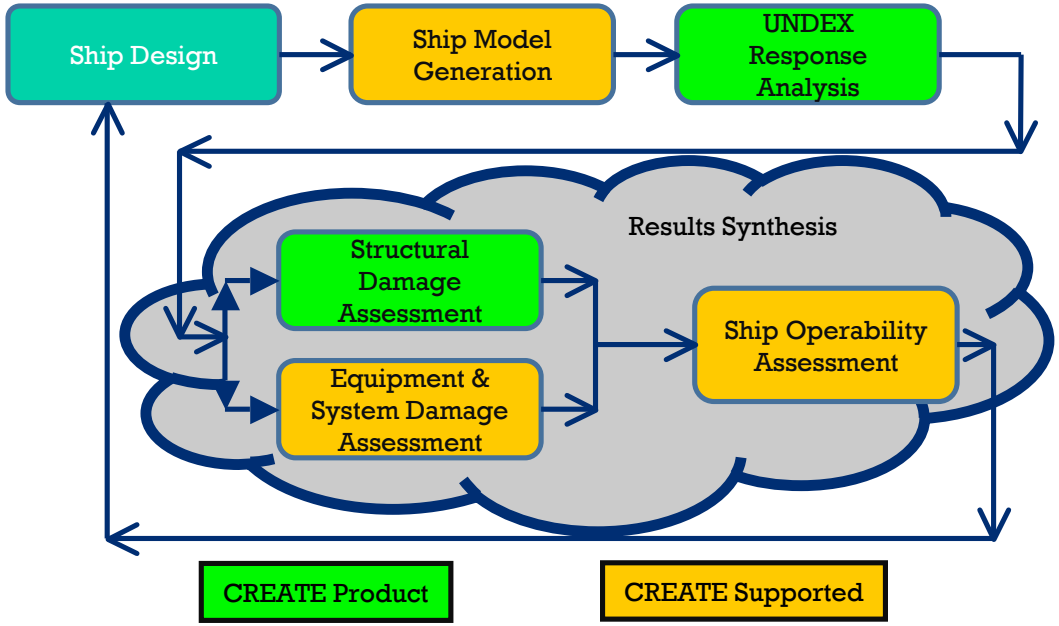
CREATE UNDEX Software FY-08 Final Review
Salinas and Gemini/Salinas Progress in 2008

September 18, 2008

Joe Jung
Computational Solid Mechanics and
Structural Dynamics Department
Sandia National Laboratories

Our Goal

To be strategic partners to the success of the NAVY/CREATE program in delivering modeling and simulation solutions for National Security



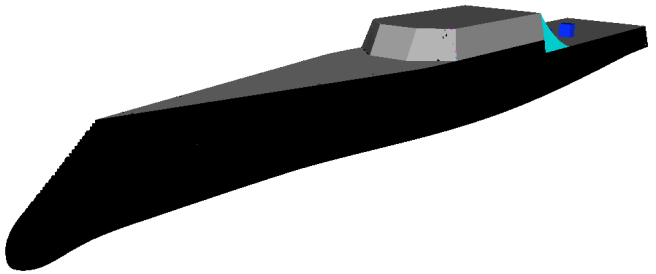
Current Status CREATE

Nastran translator

Code documentation

Sierra Mechanics (Salinas) Coupling to Gemini Via The Standard Coupler Interface (SCI)

SierraToolkit Development

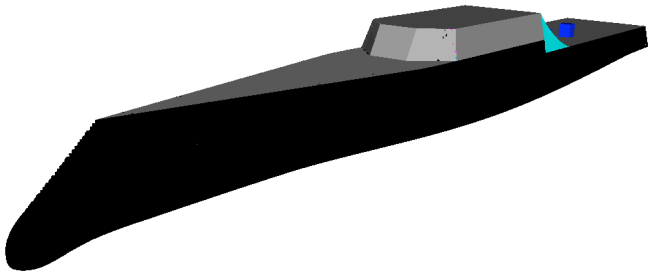


Current Status CREATE

SNL Conducted User And Salinas Developer Training At NSWC/CD

SNL Conducted Gemini/Presto & Gemini/Salinas User Training For NSWC/CD & IHD At SNL

SNL Conducted Preliminary Salinas Code Architecture Training For NSWC/CD



Current Status CREATE

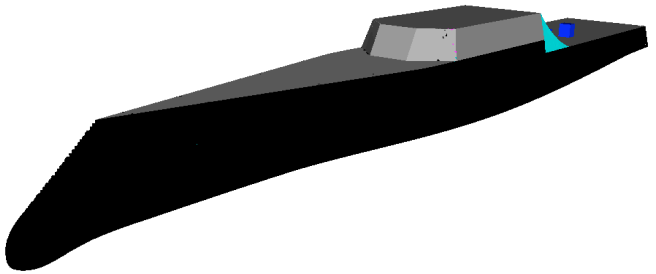
Supporting/Partnering With NSWC/CD

- Adding Navy specific element technology into Salinas
- Explicit Component Mode Synthesis (ECMS) formulation

Explicit Time Integrator in Salinas

Verification/Validation

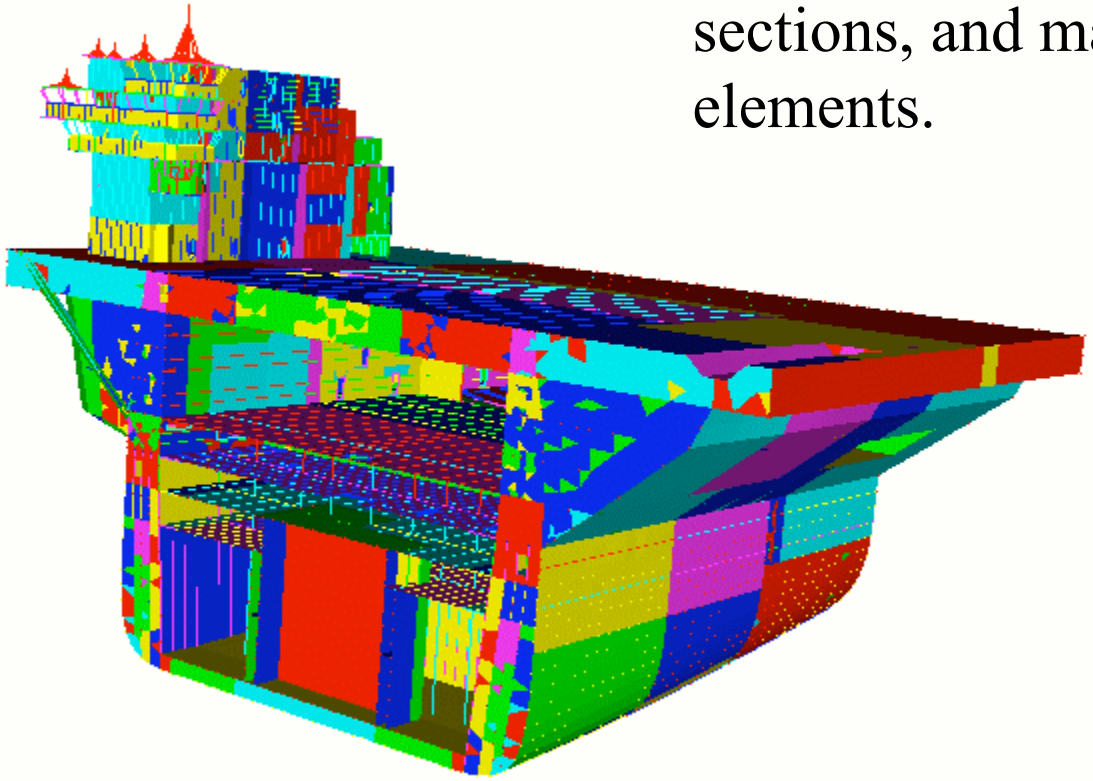
- Hydrobulge/Huang Cylinder
- IFSP/Barge model
- UHWM/Quarter scale model



Background: SIERRA Mechanics Modules

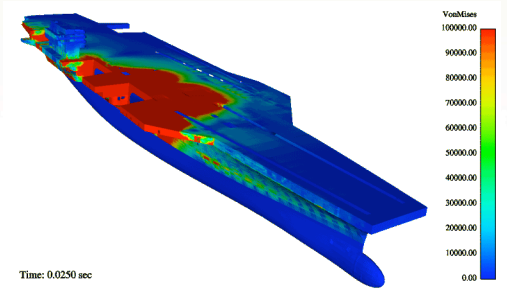
Quasistatics, Implicit dynamics and Explicit dynamics

Background: NNS - Carrier Models

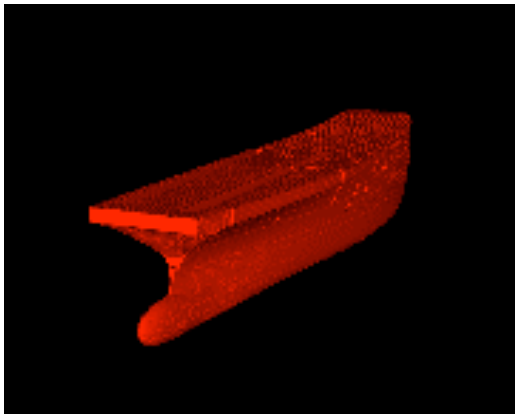


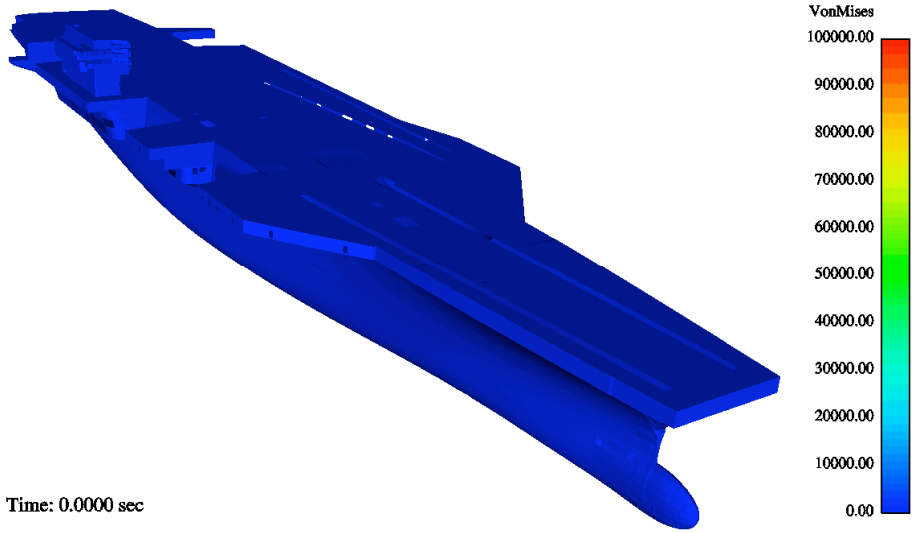
Very complex models with thousands of different material sections, and many types of elements.

Background: Shell Structures



Modal





NASGEN: Translator To Convert Nastran Models To Sierra Mechanics Data

Enhanced Support For Composites

Enhanced Support For CMS

- Includes All Superelements

Code Level Documentation

- Developer documentation added for Nasgen using Doxygen

FY09 Requirements For NASGEN Gathered For Navy Specific Models

Software Documentation

User documentation such as User's and Theory manuals in good shape.

Documentation in support of new developers was lacking.

Strategy for providing the documentation includes design template utilization and architectural documentation.

Design Documentation

Each new development requires prior documentation that includes the following (agreed to by NSWC/CD, NSWC/IH, and SNL)

- Purpose, goals and requirement
- Theory
- Changes in user interface
- Data structures, classes
- Algorithms and parallelization strategies
- Test considerations

Design template completed and in use.

Software Architecture Documentation Using Doxygen (For Code Developers)

Provide documentation of the current architecture of the code

- Some manually developed structure
- Mostly automatic generation

Hyperlinked database that provides information on:

- General Layout
- Call Trees
- Inheritance Diagrams
- Many others

Nasgen Documentation

Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia Report: SAND2008-5883P

Introduction

The Nasgen utility is used for translating a Nastran input file into an Exodus file and corresponding Salinas or Presto input files.

Architectural Documentation for Nasgen

This document is intended to provide background information for the Nasgen software architecture. It is targeted to software development personnel. Such personnel should gain information on the overall architecture of the software, the interactions between different components, and the class structures. We provide information on the call trees, dependency diagrams, and internal data for the classes. The documentation is automatically generated using special tags in the source code.

This document does NOT contain information about use of the software, nor the design decisions which drive the current architecture. The software usage is found in the user's manual, and a short reference is found in section **Using Nasgen**.

Using this Document

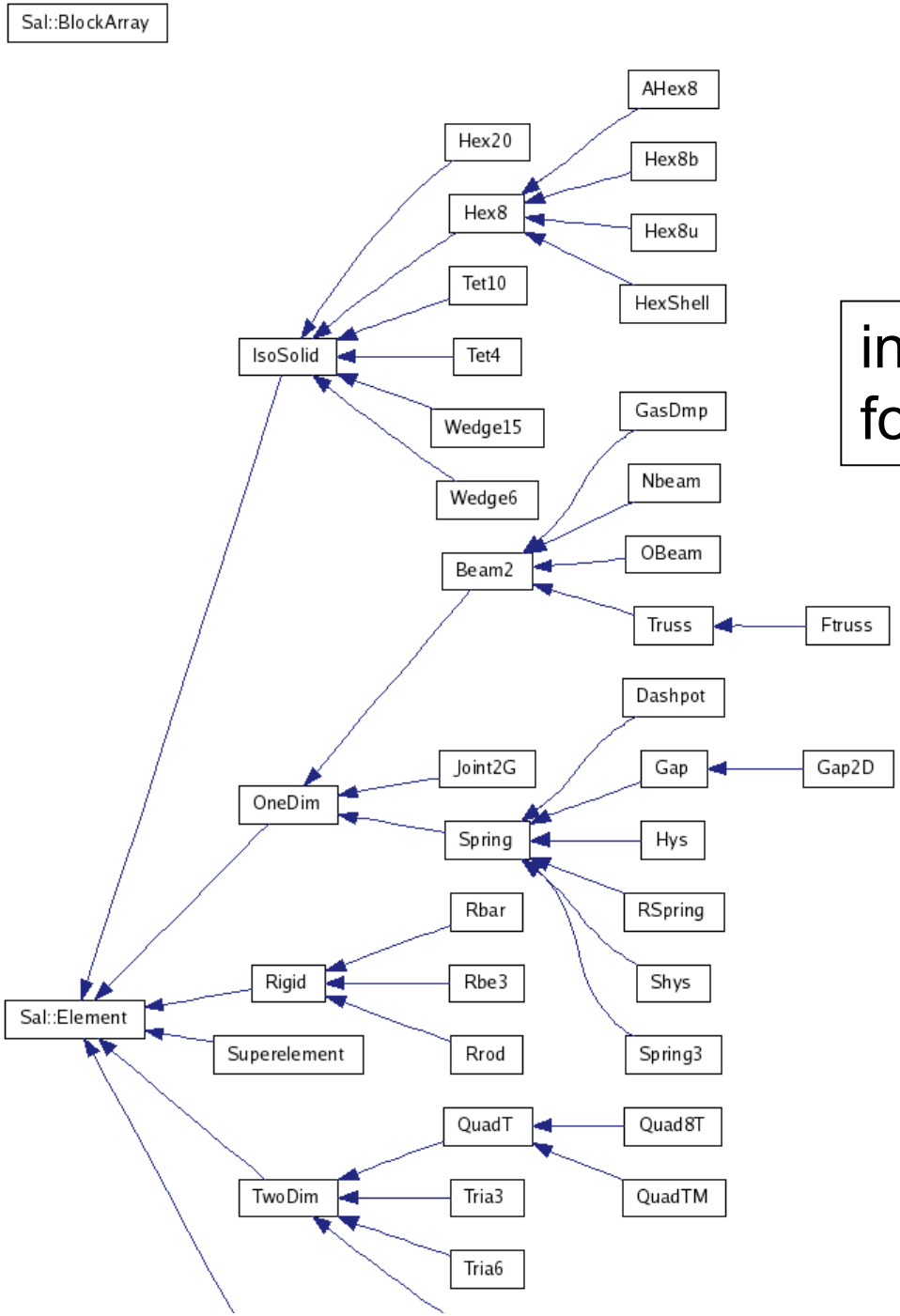
Various collections of information are available in this document. We recommend careful consideration of the table of contents. Some common starting locations for various actions are listed below.

- Call Trees. These provide a means of walking down through the software. The start of the tree can be found in the file list under `nasgen.C`.
- Dependency Diagrams. Again, these are most easily found in the file list. Each entry lists the included files for that file.
- Class Descriptions. Classes are organized in the *Data Structure* section. Each class in the code contains collaboration diagrams, private and public member functions, and detailed documentation for the functions associated with the class.
- Indices are provided for each function in the code.

What is Missing

We do not include the following:

- information about system header information.
- information from third party libraries (TPL), such as numerical libraries.
- a few, common include files are skipped because they significantly clutter the dependency graphs.



inheritance diagram for elements

Gemini/NESM Coupling Capabilities

Gemini/NESM Fully Coupled: Gemini forces passed to NESM, NESM displacements and velocities passed to Gemini.

Multiple Executables: Gemini is run as a separate executable from NESM.

File Based Communication: Uses Gemini file based communication routines. Can be extended to MPI or sockets.

Gemini/Sierra Mechanics Coupling Cont.

Subcycling of NESM: When Gemini timestep exceeds NESM timestep.

Solid or Shell: Sierra can use solid or shell elements.

Interface Mesh if Needed: Possible to define a surface mesh for Gemini and interpolate the forces to a NESM mesh. Useful during replay.

Gemini/Sierra Mechanics Coupling Cont.

Parallel Gemini/Parallel NESM: Gemini/NESM each run on multiple processors.

Teamwork: NSWC/IH and SNL worked very closely to couple Gemini to NESM via the SCI. Resolved several hurdles together.

- Parallel coupling issues
- Portability issues

Rational for Toolkit Development and FY'08 Objectives (Parallel Mesh Data Management)

- Rational
  - Prepare for changing hardware, e.g., Multi-core machines
  - Simplify current implementation
  - Improve performance
- Objectives
  - Initial conceptual model for mesh module in Sierra Toolkit
    - Document: high-level description of capabilities & requirements
  - API for mesh module
    - Header files and documentation
  - Prototype implementation(s) for mesh module
  - Performance testing of mesh module prototype(s)
    - Definition of performance test, collection/comparison of timing data
  - Algorithm-support infrastructure: mesh traversal, application of element-routines to heterogeneous mesh, etc.
    - API documentation, prototype implementation

Initial conceptual model for mesh module in Sierra Toolkit

- "Domain model" document describes the high-level concepts, capabilities and requirements for the mesh module.
  - 'living' document which will continue to change in response to the needs of application developers (i.e. the users of the mesh module).

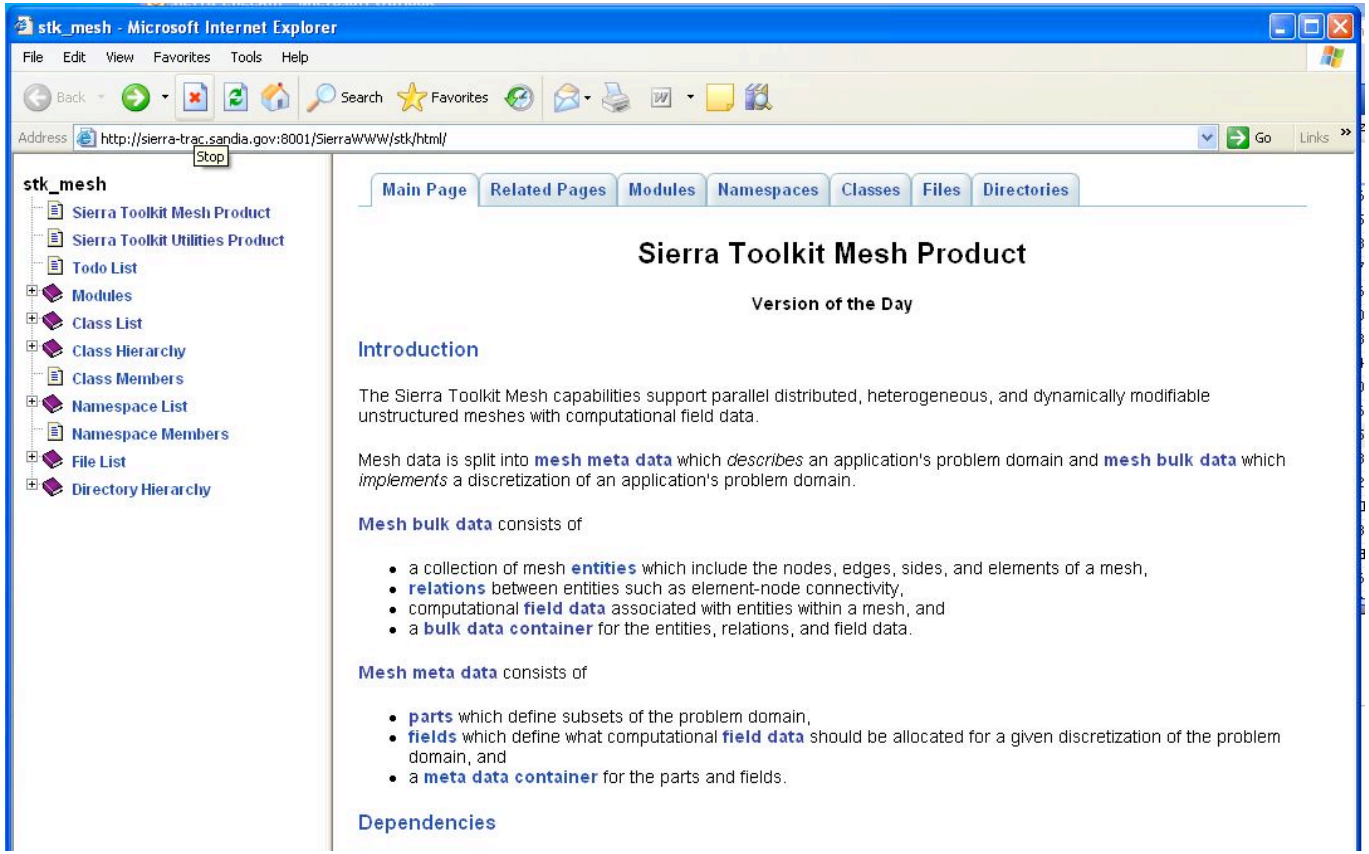
- Important distinction: here the term "mesh module" refers to the in-memory storage of mesh data structures for use by the simulation code. *Not* the on-disk mesh database produced by mesh-generators.

- Significant requirements:
  - Heterogeneous mesh and field data
    - Mixture of element topologies, ability to define fields on subsets of the mesh, and on subsets of the nodes on each element.
  - Dynamic mesh modifications
    - To support element death, parallel load balancing, H-adaptivity
  - Don't allow one feature to impose overhead on users of other features.

API (and documentation) for mesh module

- Documentation for the mesh module includes the conceptual overview document, as well as doxygen-generated documentation which can be delivered as an HTML package, or as PDF or Latex.

- Screen-shot of HTML



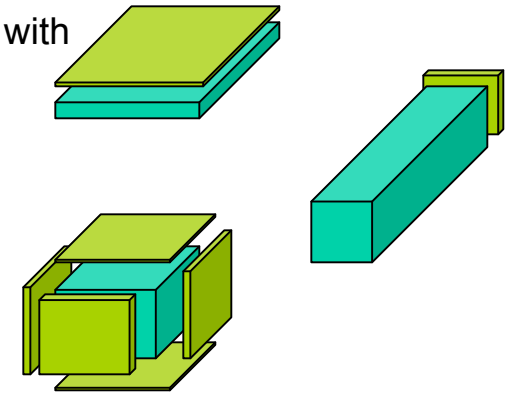
Prototype implementation(s) for mesh module

- We developed and experimented with three mesh implementations:
  - Parallel Heterogeneous Dynamic Mesh (phdMesh)
    Had been initially developed under LDRD funding, had many of our needed capabilities already in place.
  - Really Simple Mesh (rsMesh)
    Developed from scratch, attempted to replicate capabilities with a simpler API and implementation than phdmesh.
  - Array-Based Mesh (abMesh)
    Developed by a third party (another Sandia group), uses simple array data structures and was hoped to provide superior performance.

Performance testing of mesh modules prototypes(s)

- Performance refers to the cost of operations such as traversing the mesh (visiting the data for all nodes or all elements, etc) and delivering data to element-routines or other computational algorithms.

- Developed two performance tests:
  1. Simple element-loop and node-loop calculations on a large mesh in 3 configurations:
     - Mesh 1: flat square plate of 1 million hex-8 elements with 1 million quad-shells on one surface.
     - Mesh 2: long beam of 50x50x500 hex-8's with 2500 quad-shells on one end
     - Mesh 3: cube of 100x100x100 hex-8's with 10,000 quad-shells on each of the 6 sides



  2. More realistic calculation (element internal force) on arbitrary mesh configuration (mesh read from file).

- Initial tests showed all mesh prototypes were competitive for the loop and traversal calculations, but with some differences in mesh-creation times, etc. (The internal-force performance testing is ongoing.)

- We decided to proceed with a modified version of phdMesh, merging in aspects of the rsMesh API, and calling the result "stk_mesh" (Sierra ToolKit Mesh)

Consolidated development efforts to a single mesh implementation

- We decided to proceed with a modified version of phdMesh, merging in aspects of the rsMesh API, and call the result “stk_mesh” (Sierra ToolKit Mesh)
- Performance testing showed that phdMesh was competitive for the traversal and loop calculations, may need some optimization of mesh-creation phase.
- abMesh was attractive due to its simplicity, but lacked support for important capabilities (adding support for heterogeneity and dynamic mesh modifications would have eliminated the simplicity).
- rsMesh had some attractive API features, which are being merged into stk_mesh.

Training For Users And Developers

Sandia applications represent more than 10 years of effort. Bringing Navy personnel up to speed has been a principle focus.

- User focused training at SNL facilities in March.
- User and Developer training in May at NSWC/CD.
- Follow-on training at SNL in August.

Salinas – Element Technology Insertion

Methodology is in place to quickly add elements to the library.

Working closely with NSWC/CD to support addition of Navy specific element technology:

- Beam (NBeam)
- Composite Quad Shell
- Spring/Mount

Comparison Of NBeam In Salinas Versus Nastran (Frequencies)

Mode #	Nastran Result	Salinas Result
1	40.62	40.61
2	80.89	80.81
3	249.6	249.1
4	484.8	481.8
5	682.3	679.9
6	?	783.7
7	1263.6	1263.6
8	1277.9	1264.1
9	1295.9	1289.0

Explicit Integrator in NESM

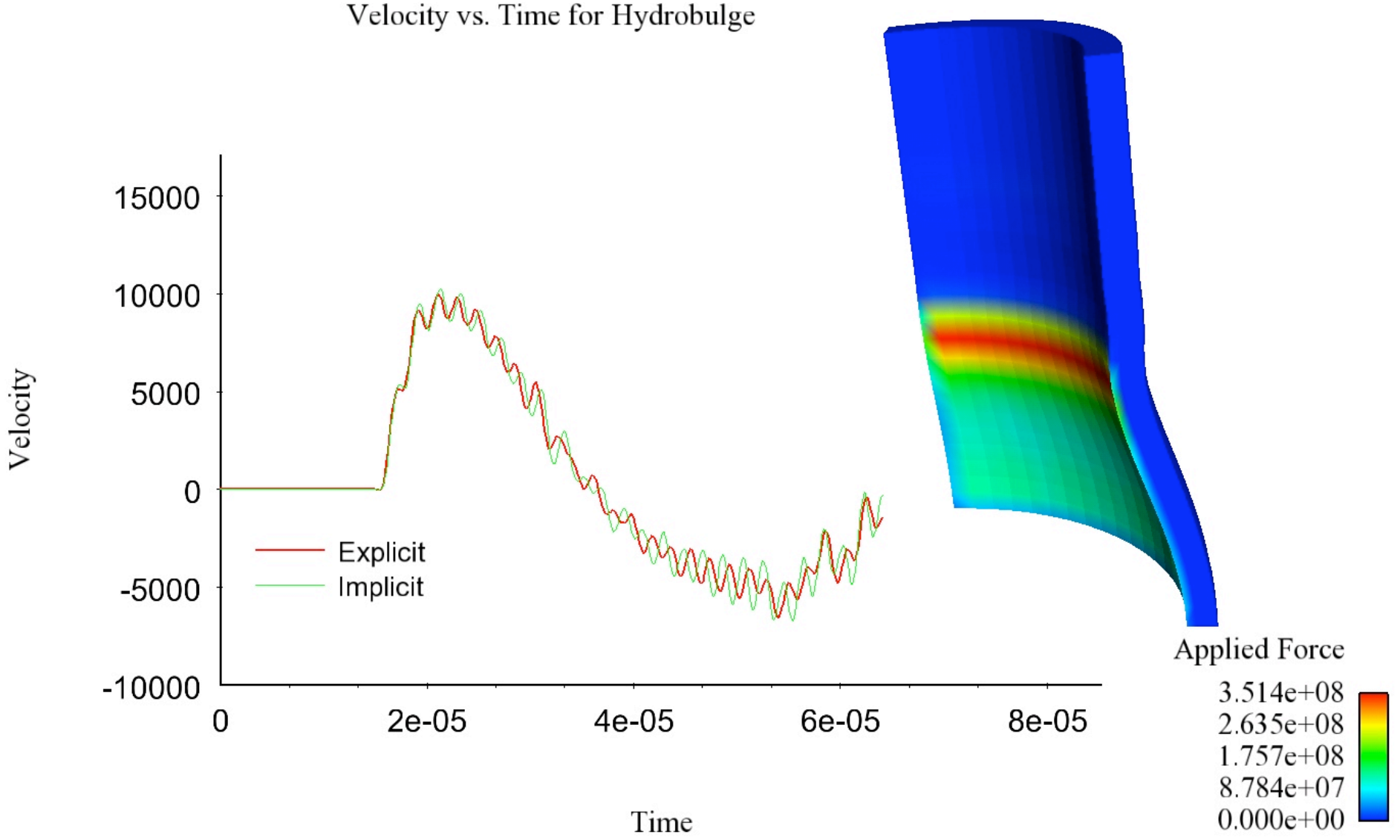
"Nearly" diagonal mass solution strategy

- Addresses complex structures such as superelements, concentrated masses, etc.
- Couples well to existing parallel software infrastructure
- Requires no refactor for changes in time steps

Integrator design completed

Integrator is complete (software implemented) for linear and mildly nonlinear elements

NESM/Gemini Coupled Calculations (NESM-Explicit)

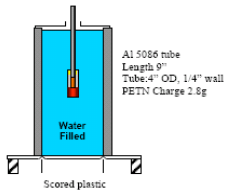
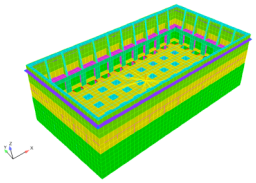
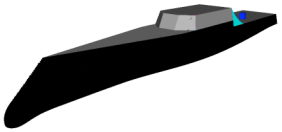


Leveraging NESM

Navy funding is not the only source for development of SM applications. Other significant developments have occurred over the past year.

- Structural Acoustics
- Pervasive Failure
- Energy Dissipation Models

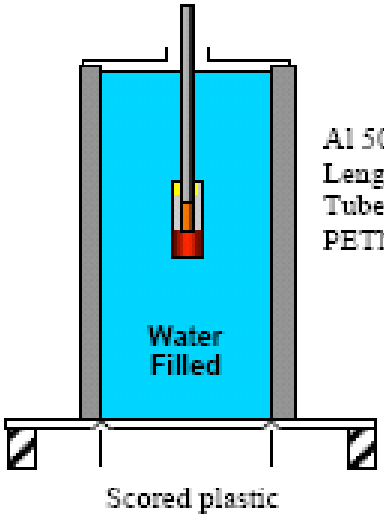
Coupling Details

	NESM	Sierra Coupler → SCI	Gemini
	Coding ✓ Debugging ✓ Verification ✓	Coding ✓ Debugging ✓ Verification ✓	Coding ✓ Debugging ✓ Verification ✓
	Coding ✓ Debugging ✓ Verification ✓	Coding ✓ Debugging × Verification ×	Coding ✓ Debugging ✓ Verification ✓
	Coding ✓ Debugging ✓ Verification ✓	Coding ✓ Debugging × Verification ×	Coding ✓ Debugging ✓ Verification ✓

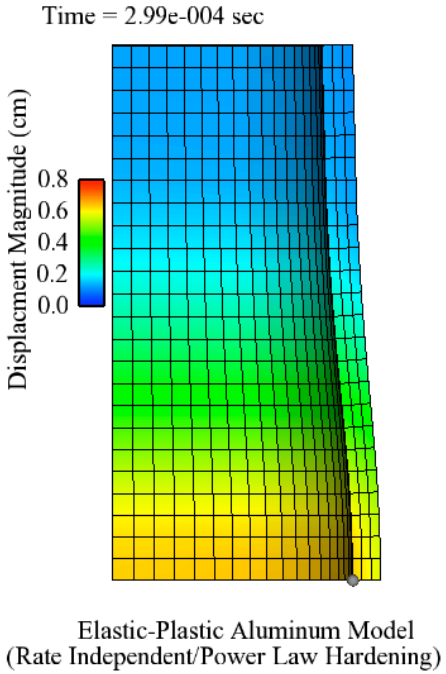
*Performance Assessment And Optimization Is Still Needed For All Cases

Hydrobulge Validation Problem (With Linear And Non-Linear Models)

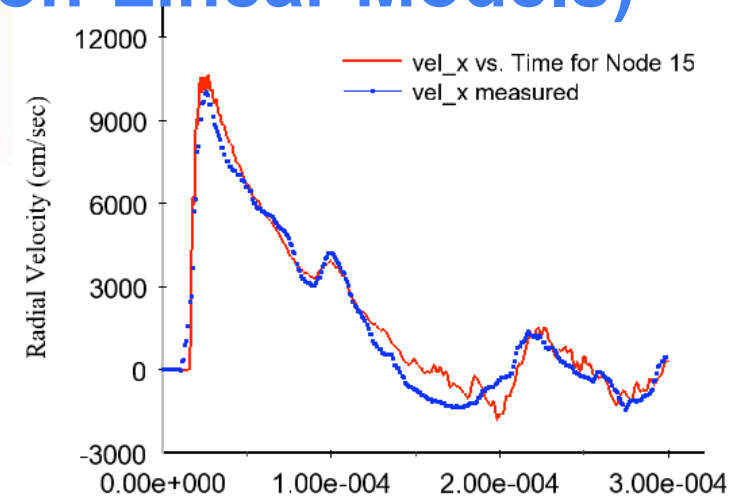
Water filled cylinder with a charge of PETN detonated.

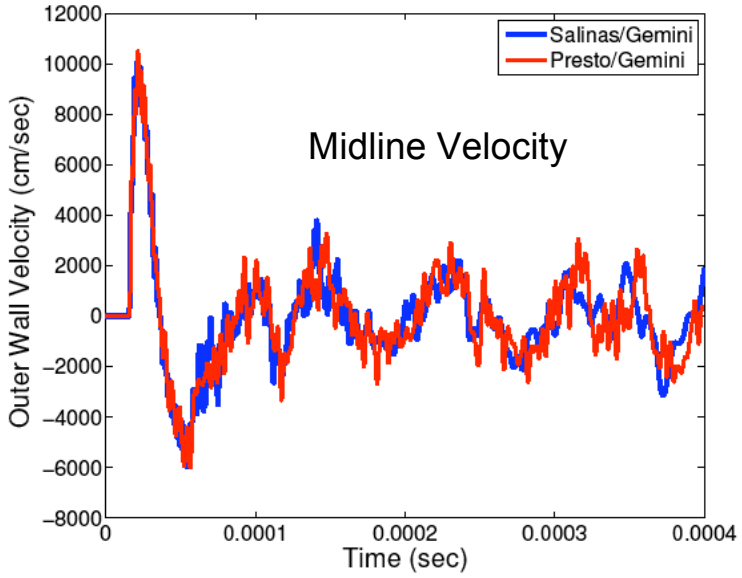


[2] G. Champers, et al. Pressure measurements on a deforming surface in response to an underwater explosion. *Shock Compression of Condensed Matter*, Ed. Schmidt, Dandekar, Forbes, American Institute of Physics, 1998.

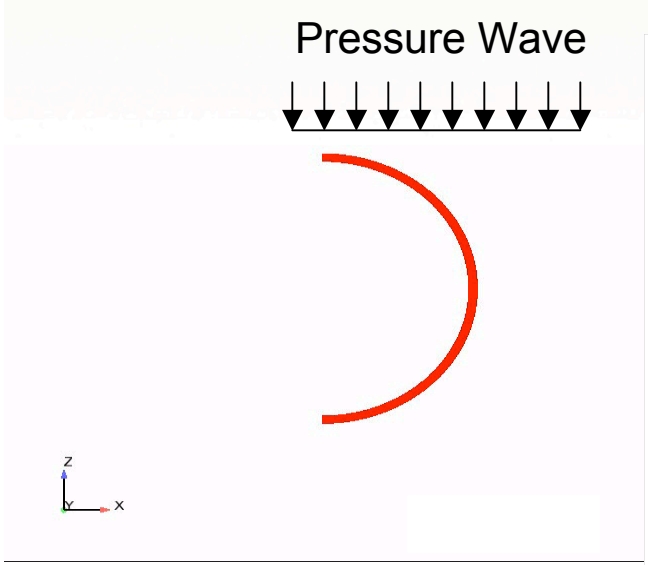


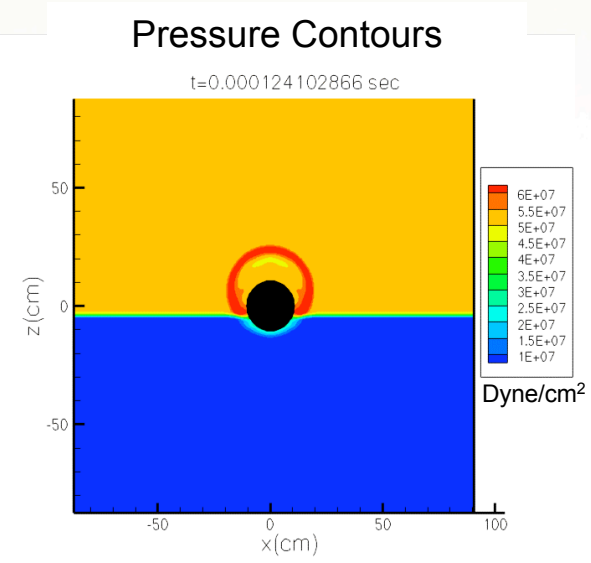
Elastic-Plastic Aluminum Model
(Rate Independent/Power Law Hardening)

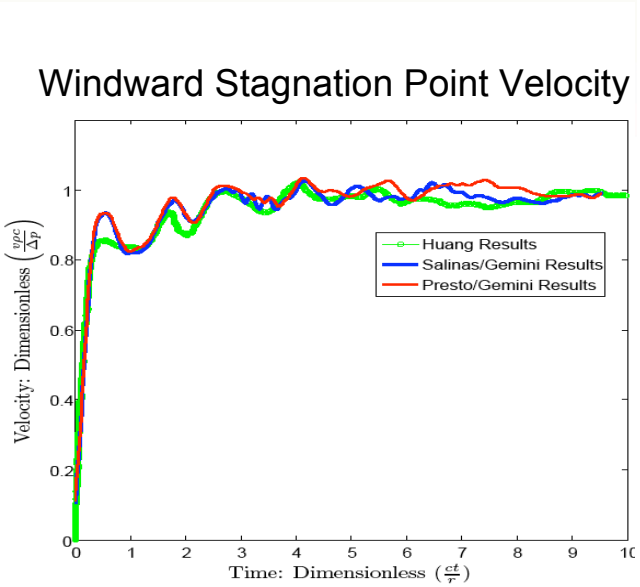




Huang Cylinder (Benchmark)







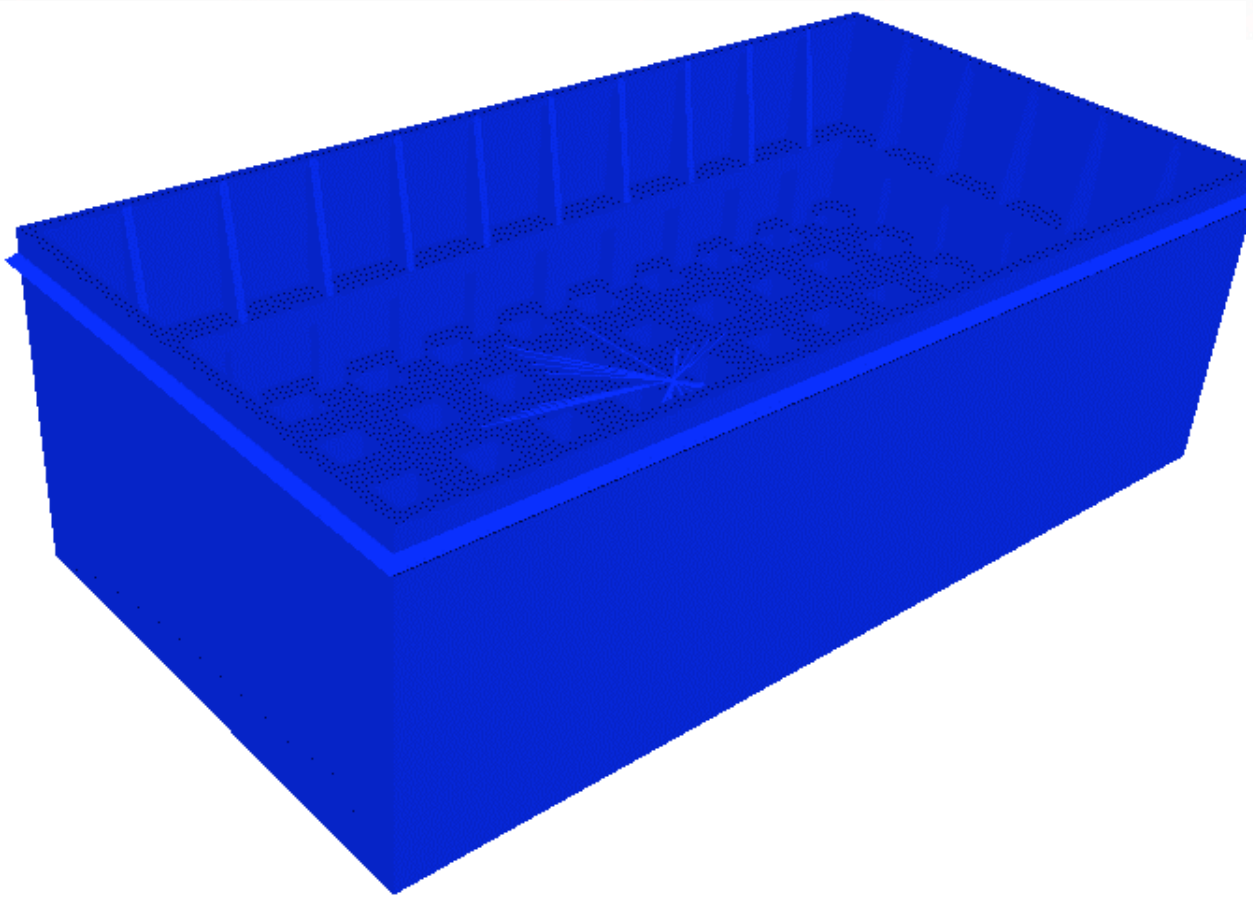
Plane shock wave (50:1 pressure ratio) interacts with a linear elastic circular cylinder.

Numerical derived results with an acoustic wave [1].

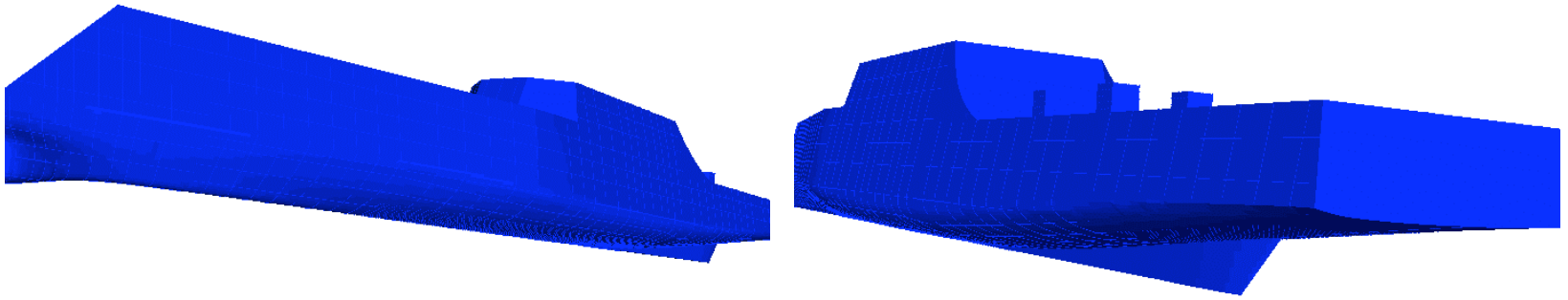
Used for an understanding of coupling set-up.

[1] H. Huang. An exact analysis of the transient interaction of acoustic place waves. *J. Appl. Mech.*, 37(4):1091–1099, 1970.

Barge Model With Arbitrary Pressure Loading



Quarter Scale Model With Arbitrary Pressure Loading



CREATE – Future Work

Explicit Component Mode Synthesis (ECMS) capability

Wet Modes

V&V: Barge and Quarter scale

Additional NASGEN translator work

Explicit integrator in Salinas

- Performance
- Constraints

Documentation/Training