AD A

Coupled Simulation of Vehicle Dynamics and Tank Slosh: Phase 2

INTERIM REPORT
TFLRF No. 368

by

F. A. Thomassy
G. R. Wendel
S. T. Green
A. C. Jank
U.S. Army TARDEC Fuels and Lubricants Research Facility (SwRI)
Southwest Research Institute
San Antonio, TX

for

U.S. Army
AMSTA-DSA-FP-PW
Warren, MI

Under Contract to

U.S. Army TARDEC
Petroleum and Water Business Area
Warren, MI

Contract No. DAAE-07-99-C-L053 (WD14)
SwRI Project No. 03.03227.14

September 2003

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Trade names cited in this report do not constitute an official endorsement or approval of the use of such commercial hardware or software.

DTIC Availability Notice

Qualified requestors may obtain copies of this report from the Defense Technical Information Center, Attn: DTIC-OCC, 8725 John J. Kingman Road, Suite 0944, Fort Belvoir, Virginia 22060-6218.

Disposition Instructions

Destroy this report when no longer needed. Do not return it to the originator.

Coupled Simulation of Vehicle Dynamics and Tank Slosh: Phase 2

INTERIM REPORT
TFLRF No. 368

by

F. A. Thomassy

G. R. Wendel
S. T. Green
A. C. Jank
U.S. Army TARDEC Fuels and Lubricants Research Facility (SwRI)
Southwest Research Institute
San Antonio, TX

for
U.S. Army
AMSTA-DSA-FP-PW

Under Contract to
U.S. Army TARDEC
Petroleum and Water Business Area
Warren, MI

Contract No. DAAE-07-99-C-L053 (WD14)
SwRI Project No. 03.03227.14

Approved for public release; distribution unlimited

September 2003

Approved by:

Edwin C. Owens, Director
U.S. Army TARDEC Fuels and Lubricants
Research Facility (SwRI)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE	2. REPORT DATE September 2003	3. REPORT TYPE AND DATES COVERED February 2003 - September 2003

4. TITILE AND SUBTITLE
Coupled Simulation of Vehicle Dynamics and Tank Slosh

5. FUNDING NUMBERS
DAAE-07-99-C-L053
WD 14

6. AUTHOR(S)
Thomassy, F.A., Wendel, G.R., Green, S.T., Jank, A.C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army TARDEC Fuels and Lubricants Research Facility (SwRI)
Southwest Research Institute
P.O. Drawer 28510
San Antonio, Texas 78228-0510

8. PERFORMING ORGANIZATION REPORT NUMBER

TFLRF No. 368

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army TACOM
U.S. Army TARDEC
Petroleum and Water Business Area
Warren, MI 48397-5000

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Computer simulation of vehicle dynamics has become a valuable tool in the design of vehicles. However, it is unable to simulate the complex dynamics of fluid "sloshing" in a tank on the vehicle. Computational Fluid Dynamics (CFD) analysis software is available that can predict fluid slosh, however, it does not facilitate computation of the vehicle's mechanical system.

This is the second phase of a multiphase program to develop and demonstrate the use of CFD analysis, coupled with vehicle dynamics analysis, to more accurately predict the dynamics of a fluid transport system. The first phase validated that CFD analysis predicts slosh dynamics of a tank subjected to motions of a vehicle encountering typical maneuvers. The results of this study (phase 2) demonstrated the successful development of subroutines that are integrated into vehicle dynamic and CFD simulation codes.

14. SUBJECT TERMS
Computer Simulation Vehicle Dynamics Tank Slosh Computational Fluid Dynamics CFD Fluid Transport System

15. NUMBER OF PAGES
62

16. PRICE CODE

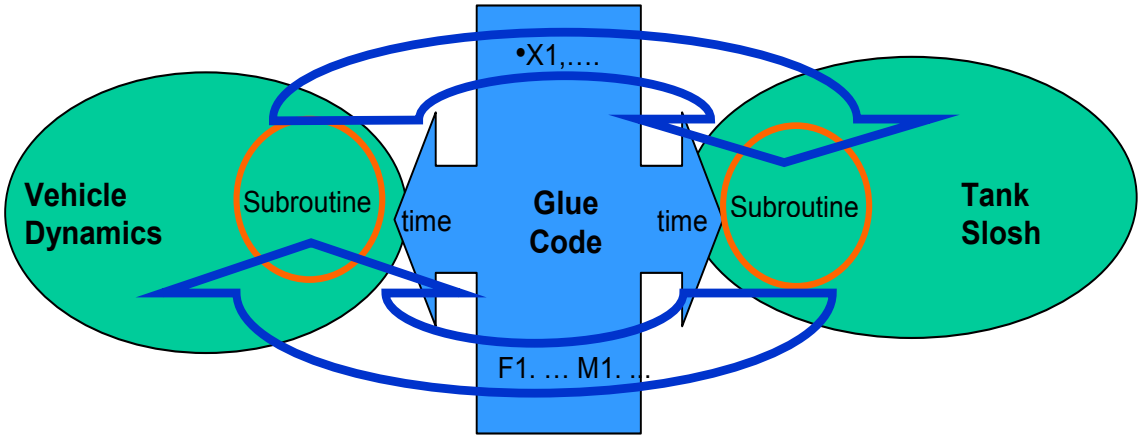
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT

EXECUTIVE SUMMARY

Computer simulation of vehicle dynamics has become a valuable tool in the design of vehicles. However, it is unable to simulate the complex dynamics of fluid "sloshing" in a tank on the vehicle. Computational Fluid Dynamics (CFD) analysis software is available that can predict fluid slosh, however, it does not facilitate computation of the vehicle's mechanical system.

This is the second phase of a multiphase program to develop and demonstrate the use of CFD analysis, coupled with vehicle dynamics analysis, to more accurately predict the dynamics of a fluid transport system. The first phase validated that CFD analysis predicts slosh dynamics of a tank subjected to motions of a vehicle encountering typical maneuvers. The results of Phase 1 demonstrated that fluid slosh loads could be accurately predicted by CFD and that the effect of fluid slosh is significant and will have an important affect on vehicle dynamics.

The objective of the second phase is to develop a methodology where vehicle dynamics and fluid slosh can be simulated simultaneously so that the highly coupled behavior of fluid and vehicle motion is captured. As such, Phase 2 encompasses a software development effort that resulted in a co-simulation procedure (CoSim) involving a master program that manages the exchange of data between and synchronization of two separate simulation codes. These codes are the vehicle dynamics and tank slosh simulations. The master program (called the Glue Code) provides motion data from the vehicle dynamics simulation to the tank slosh simulation and force data from the tank slosh simulation to the vehicle dynamics simulation (see the figure below). Synchronization of the two simulations is managed by the Glue Code by providing the data when it is needed and pausing simulations when necessary to wait for the required data.



Tank motion and slosh load results from CoSim runs have been correlated with measurements from the Phase 1 test fixture. This has demonstrated the success of communication protocols in correctly driving the CFD simulation and maintaining synchronization of the two codes. The limitation of this demonstration is that tank motion is known from test data rather than determined from simulation. The true test of this methodology will come when full dynamic interaction is correlated with test results. Initial simulations involving multiple mechanical degrees of freedom (in the vehicle model) have yielded unexpected results. This means that careful study of the coupled dynamic system will need to be performed and compared to test. It may also be necessary to communicate additional data between the two codes and alter how the fluid force is handled in the vehicle dynamics code.

The result of this study has demonstrated the successful development of subroutines that are integrated into vehicle dynamic and CFD simulation codes. A Glue Code program manages data communication between the codes and maintains synchronization. Integrated simulations have been performed and correlated with test data from the first phase of this project. The next step is to further developed CoSim and demonstrate the accurate solution in highly non-linear dynamic scenarios. Validation testing should be performed using both small- and full-scale hardware.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure **Page**

1.0 INTRODUCTION

Fluid transport systems have an inherent concern with dynamic stability since fluid motion, or "sloshing", introduces large dynamic loads to the vehicle system. Dynamic phasing of these loads can result in vehicle rollover or loss of control if the tank and vehicle system is not properly designed. As a result, a wide range of conditions including various terrain, maneuvers, and fluid fill levels need to be considered. Instability of a vehicle that is carrying or towing a tank is a significant safety concern, particularly when handling flammable fluids. Fluid spills can also be an environmental hazard.

Current vehicle dynamics modeling software, such as LMS/Motion and MSC.ADAMS software, is capable of providing good predictions of dynamic performance for typical vehicles. However, they do not include an accurate method of representing the dynamic effects of fluid in a tank. Traditional methods of representing fluid slosh dynamics use a model in which the mass of the fluid is represented mathematically as a concentrated mass swinging from a pendulum with a length that will represent the fluid's natural frequency. The pendulum model is incorporated into a vehicle dynamics model to produce a coupled simulation. The drawback of a pendulum model is that it is only good for linear fluid motion, or when the fluid surface remains relatively flat. The pendulum model is not accurate when waves develop, when the fluid rolls over, or when there is a separation of fluid. It is also very difficult to use the pendulum model to simulate a tank with baffles.

Current advancements in fluid dynamic modeling software, more commonly referred to as "Computational Fluid Dynamics" (CFD) analysis tools, make it possible to accurately represent the dynamic motion of free surface fluid sloshing, including waves and fluid separation. This type of analysis is generally referred to as a "volume of fluid" analysis, in which there is a finite fluid volume that is accounted for as it is exposed to motions from a container of a larger volume.

Vehicle motion affects fluid in the tank, which consequently imparts loads on the vehicle that affects its motion. The dynamic interaction of motion and load feedback between the vehicle and fluid must be modeled as a coupled system. There is no practical means of modeling them separately. It would be ideal to couple vehicle dynamics and fluid slosh in a simultaneous simulation process. There is no known modeling tool commercially available.

The goal of this multi-phase program is to develop the means for performing this type of analysis using commercially available software and providing the appropriate links for coupling these software packages. The first step (Phase 1) in this process successfully demonstrated and verified that a CFD simulation can accurately simulate the fluid motion and loads of a typical tank application. The second step (Phase 2) is to develop and demonstrate a methodology that couples simulation of vehicle and fluid dynamics. The results of Phase 2 are summarized in this report.

The coupled simulation (CoSim) methodology is based on the recognition that mechanical dynamics codes are inherently good predictors of vehicle motion, while CFD codes are good predictors of fluid load. As such, CoSim involves the synchronized simulation of the two codes where motion data is delivered to the CFD code and reaction loads are returned to the dynamics code. This report describes the methodology and implementation of the CoSim method. CoSim results are verified by comparison with Phase 1 test data. Finally, the potential of the CoSim method is demonstrated by comparing simulation results with varying levels of structural compliance.

The simulation method of coupling vehicle dynamics and tank slosh being developed in this program will have a significant impact on the development of fluid transport systems for both military and commercial applications. Phase 3 is planned to include validation of a full-scale model of a tank on an actual vehicle in a field test or on a dynamic simulator. Phase 4 will develop the application of coupled simulation to optimizing the design of a vehicle with tank subject to sloshing.

2.0 CoSIM CONCEPTS AND DESIGN

2.1 Purpose

While the specific task of this project is to couple mechanical dynamics with CFD, the concept developed is one of simultaneous management of data between dynamic applications whose results are codependent. As such, this methodology may be used to exchange data between any two simulations that run simultaneously.

2.2 Terminology

- *CoSim*: An acronym for the software and process of coupled simulation. This term is used as a noun to label the methodology developed here and as a verb to describe the action of coupled simulation.
- *Glue Code*: This term describes the core code and executable developed by MSC.Software. The Glue Code manages the data flow between the applications involved in CoSim.
- *MSS*: Multi-body Structural Simulation (MSS). For example, the MSC.ADAMS code is a MSS solver.
- *CFD*: Computational Fluid Dynamics (CFD).
- *Emulator Code*: A generic term used to describe code that has been developed to interact with the *Glue Code* as if it were a MSS or CFD solution solver. This code is designed to run fast and is used to streamline implementation of new applications for use in CoSim by facilitating unit testing and validation.
- *Pipe*: A pipe can be thought of as a data string that resides in memory. Pipes are read from and written to in a fashion similar to a file however, pipe access is very fast.
- *Blocking*: Blocking is a method used to pause an application during solution. In CoSim, a block occurs when data is requested by an application from the Glue Code. The application is forced to wait while the Glue Code calculates the required data or waits for that data from another application.
- *Input data*: Data flow is labeled from the perspective of the MSS code. Therefore, input data is created in the CFD code and delivered to the MSS code. This is generally force data.
- *Output data*: Data flow is labeled from the perspective of the MSS code. Therefore, output data is created in the MSS code and delivered to the CFD code. This is generally motion data.

2.3 Design of CoSim

Dr. Andrew Elliot of MSC.Consulting developed the original concepts behind CoSim. His original design and implementation are detailed in his conference papers[1] [2]. Dr. Elliot was consulted on this project and the specific designs used to create a coupling with fluid slosh simulation are detailed here.

Figure 1 illustrates the essence of the CoSim process, which is divided into three groups of software. The Glue Code acts as an executive routine that directs the flow of data between the MSS and CFD codes. CoSim starts with launch of the Glue Code, which prompts the user to start the CFD application. The CFD code uses a customized subroutine to initialize communication with the Glue Code and requests motion data for time equal to zero plus an initial increment. Blocking of this data causes the CFD code to pause while the Glue Code prompts the user to start the MSS code. The MSS code is run with initial condition loads until simulation time is greater than the time the CFD code indicated it would increment too. The MSS motion data is interpolated and delivered to the CFD code once the MSS code has advanced. At that point the MSS code is blocked and the CFD code is advanced with the Glue Code providing interpolated data for each time requested. Extrapolated load data is delivered to the MSS code when the Glue Code determines that the CFD code has advanced to an appropriate time. In this manner, the Glue Code allows the two applications to exchange data and maintain synchronization.

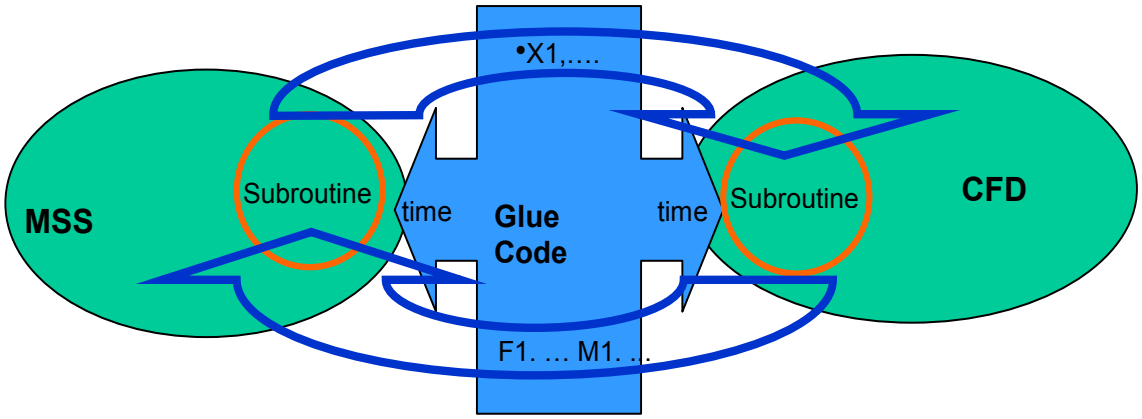


Figure 1. Co-simulation Concept

The principal feature of the CoSim process is the data pipe. A specific naming scheme is implemented to create data locations in machine memory. Two pipes are necessary for the PC and four for Unix. The PC pipes are one each for the MSS code and CFD code. For the Unix system it is necessary to have one input and one output pipe for the MSS code and another set for the CFD code. WriteFile and ReadFile functions implemented in the communication subroutines are used to access these pipes. These functions are described later.

[1] Elliott, A. S., "**A Highly Efficient, General-Purpose Approach for Co-Simulation with ADAMS®**", presented at the 15th European ADAMS Users Conference, November 2000, Rome, Italy.
[2] Elliott, A. S., "**Status Update on Advanced, General-Purpose Co-Simulation with ADAMS®**", presented at the 2002 North American ADAMS Users Conference, May 2002, Scottsdale, Arizona.

Appendix A provides three flow charts, which outline the basic design of the CoSim methodology. Several simplifications have been made with these flow charts to clearly display the principal flow of data. The flow charts of Appendix A are referenced in the sections below, which describe the design of each of the three major groups of code.

2.3.1 Glue Code Data Flow

CoSim execution begins with the Glue Code, which consists of the following code components:

inex64chkt.f: The main executive code, which contains the primary logic of CoSim.
interp.f: A fast interpolation subroutine based on the abstraction of three data points.
pipesetup.f: This subroutine creates the pipes for the MSS and CFD codes.

Dr. Elliot originally created these programs and the underlying methodology. His expertise was used to establish communication and data flow between MSC.ADAMS and Flow-3D®. It should be pointed out that integration with CFD was not originally envisioned as part of this system. One of the principle assumptions is that the non-MSS code should take displacement or velocity data as input. CFD based slosh simulations (both Flow-3D and Fluent) need acceleration data as input because non-inertial reference frames are used to solve this class of problem. This could be a significant problem in dynamic interactions involving high frequency events but may not be a problem with relatively low frequency events such as the effects of slosh on vehicle performance. This issue will not be solved in this report because test and correlation will be required to resolve this issue. An alternative may be to explore slosh simulation methods (such as explicit dynamic solutions) other than CFD.

The Glue Code flowchart can be found in Appendix A and should be referenced in the following discussion. The Glue Code calls the *pipesetup* subroutine as soon as the program is executed. Each pipe is setup to accommodate 64 double precision variables. However, the programmer can put any type of data in the pipe. The *pipesetup* subroutine then manages startup of the MSS and CFD codes by prompting the user to start each code. Each code will establish communication with the pipes by passing data that declares the number of input and output variables it expects to handle. The *pipesetup* subroutine returns to the main program when all initialization data has been successfully checked.

The remainder of the simulation process involves the MSS and CFD simulations taking turns solving iterations and exchanging data through the Glue Code. The data flow starts with the MSS code (ADAMS in the flowchart) requesting load data from the Glue Code. The Glue Code delivers initial load information without reference to either code but after that all data exchanged is determined from the coupled simulations. The MSS code is expected to request each variable individually rather than as an entire block of data. This is a specific feature of how ADAMS requests simulation data (in fact ADAMS requests the same data multiple times in a single iteration).

The MSS sends *StepFlag = true* when a successful time step has been completed. At this point the Glue Code will store the motion data from the successful time and check with the CFD code to see if it is appropriate for it to advance. The CFD code will advance any time the expected time at the end of the step is less than or equal to the time of the last successful MSS step. This means that the MSS code is always ahead of the CFD code. The Glue Code delivers interpolated motion data (*output data*) to the CFD code when it is appropriate for it to advance. The CFD code returns load data (*input data*) when its load step is complete.

Finally, the Glue Code extrapolates the stored input data for the requested MSS variable and waits for the next request from the MSS code. It must be kept in mind that interpolation and extrapolation are based upon the last three successful iterations of the relevant data. This can be a problem if the time step sizes of the MSS and CFD codes are significantly different since the data may be extrapolated well beyond and acceptable time proximity to the stored data points.

2.3.2 CFD Data Flow

The CFD code must include user subroutines that request and deliver data to the pipes established by the Glue Code. Each CFD code will have special needs for logic and sequence of events. This section describes the implementation of the CFD emulator code to illustrate the critical aspects of CoSim for the CFD code. The CFD emulator is made up of the following components:

CFDEmulator.f: The main executive code, which contains the solution sequence logic.
CFDUtils.f: Emulates the solution and generation of load data.
CCCUtils_cosim.f: Several subroutines for managing pipe communication.

The CFD emulator flowchart can be found in Appendix A and should be referenced in the following discussion. The CFD code is launched after the Glue Code prompts the user to do so. The CFD code will proceed through its initialization steps and call the subroutine *CCCconnect*, which checks communication with the pipes and confirms that the correct data is expected for in put and output. Next, the CFD code will exchange data with the Glue Code to discover the last successful time of the MSS code and deliver the solution time that the CFD code intends to take next. The MSS and CFD times are compared to determine if the CFD code should take a step (note that the same logic is used in the CFD code and Glue Code to maintain the proper sequence of events). Before a CFD step is taken the Glue Code delivers interpolated motion data. When the CFD step is completed the resultant load data is returned to the Glue Code and the process returns to the top.

2.3.3 MSS Data Flow

The MSS code must be modified to include subroutines that request and deliver data to the pipes established by the Glue Code. The MSS code (only ADAMS was implemented in this effort) will have special needs for logic and sequence of events. This section describes the implementation of the MSS emulator code to illustrate the critical aspects of CoSim for the MSS code. The MSS emulator is made up of the following components:

MSSEmulator.f: The main executive code, which contains the solution sequence logic.
MSSUtils.f: Emulates the some data functions.
consub64emulator.f: Subroutines for opening and closing pipe communication.

The MSS emulator flowchart can be found in Appendix A and should be referenced in the following discussion. The MSS code is launched after the Glue Code prompts the user to do so. The MSS code will proceed through its initialization steps and establish communication with the pipes. The MSS code is expected to solve an initial iteration and proceed to a point in simulation time that is ahead of the CFD code. The iteration process includes multiple calls to the Glue Code for extrapolated CFD data. In the solution steps taken before the first step of the CFD code the extrapolated data is based on initial conditions that are hard coded into the Glue Code (usually some steady state condition). Actual CFD data is extrapolated to the MSS code after both codes (MSS and CFD) have taken their first successful steps.

The MSS code will set *StepFlag* = *true* when it has completed a step. This flag is set only during the first request for data with a new time step. At this point the Glue Code blocks further solution of the MSS while it determines the extrapolated value of the data requested by the MSS code. If *StepFlag* = *true* then the Glue Code will check with the CFD code to determine if it should take a step and update the input data or if the MSS code should take another step.

3.0 SIMULATION MODELS

The CoSim methodology has been shown to compare well with the results of the Phase 1 work. In Phase 1 a test fixture was created to represent a small scale FMTV tank (Figure 2). Flow-3D simulations were shown to correlate well with test data for six simulated maneuvers. These maneuvers are:

- AVTP lane change at 20 mph
- AVTP lane change at 40 mph
- 9” half-round symetric bump at 10 mph
- 12” half-round symetric bump at 5 mph
- 9” trapazoidal asymetric bump at 15 mph
- 12” trapazoidal asymetric bump at 10 mph

This section describes the models used to demonstrate coupled simulation (CoSim) of the small scale TLUD tank.

Figure 2. Phase 1 Test Fixture

3.1 MSC.ADAMS Model of Phase 1 Test Fixture

The original Phase 1 Pro/Engineer (Pro/E) computer aided design (CAD) model (Figure 3) was used as a basis for modeling the kinematics of the test fixture. Mechanism/Pro (Mech/Pro), a MSC.ADAMS connection product imbedded within Pro/E, was used to define the structural joints of the mechanical system. In Figure 3, the geometry represents the Pro/E CAD model while the kinematic joints represent the MECH/Pro model that is created from the Pro/E geometry. The model was then transferred to MSC.ADAMS/View for simulation (Figure 4). An ADAMS Command File (*.acf) was created to automate the setup of all modeling parameters for simulating the six maneuvers in a CoSim with a CFD package.

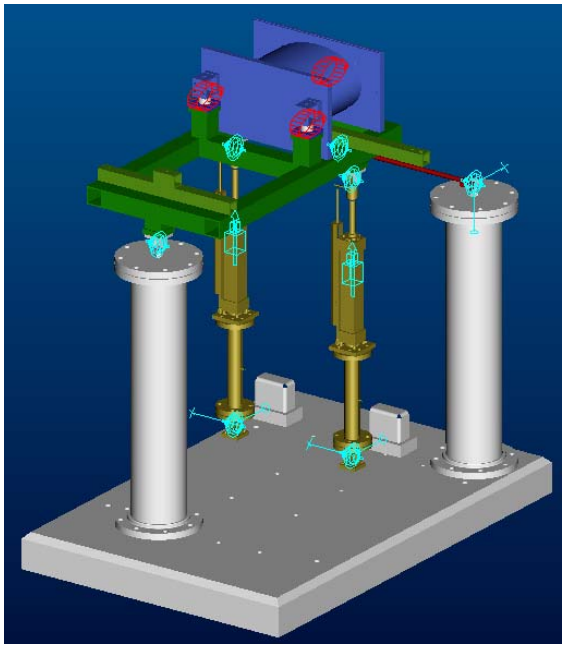


Figure 3. Pro/E and MECH/Pro Models

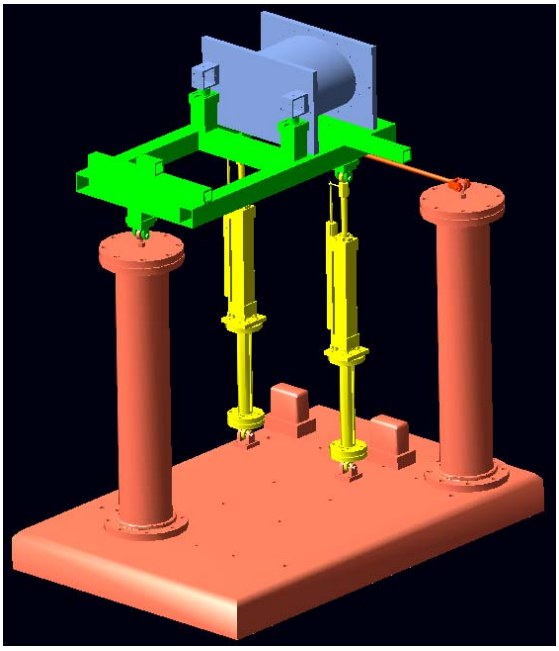


Figure 4. MSC.ADAMS Model

The solution depends on two user subroutines to handle data transfer with the GlueCode. The *consub64.f* subroutine takes care of initialization and shutdown of pipe communication. The *varsub64.f* subroutine handles the transfer of input and output data during simulation. Appendix B summarizes the installation, compilation and execution of the MSC.ADAMS model.

3.2 Flow-3D® Model of Phase 1 Slosh Tank

The model geometry and setup come directly from the original Phase 1 simulation (Figure 5). The major difference is that the simulation subroutine for motion data has been replaced with CoSim subroutines. The following Flow-3D subroutines were implemented for CoSim:

- *CCCutils_cosim.f*: Contains utility subroutines for pipe communication.
- *rusrd_cosim.f*: Handles initialization of CoSim data.
- *motion_cosim.f*: Communicates with the GlueCode to determine when to proceed with a step. Once a step is taken this subroutine collects motion data (*output data*) and proceeds.
- *qsadd_cosim.f*: Call for force data, makes corrections and communicates load data (*input data*) to the GlueCode.
- *FonlyEval_cosim.f*: Subroutine loops through the model and reports the current summation of force and torque.

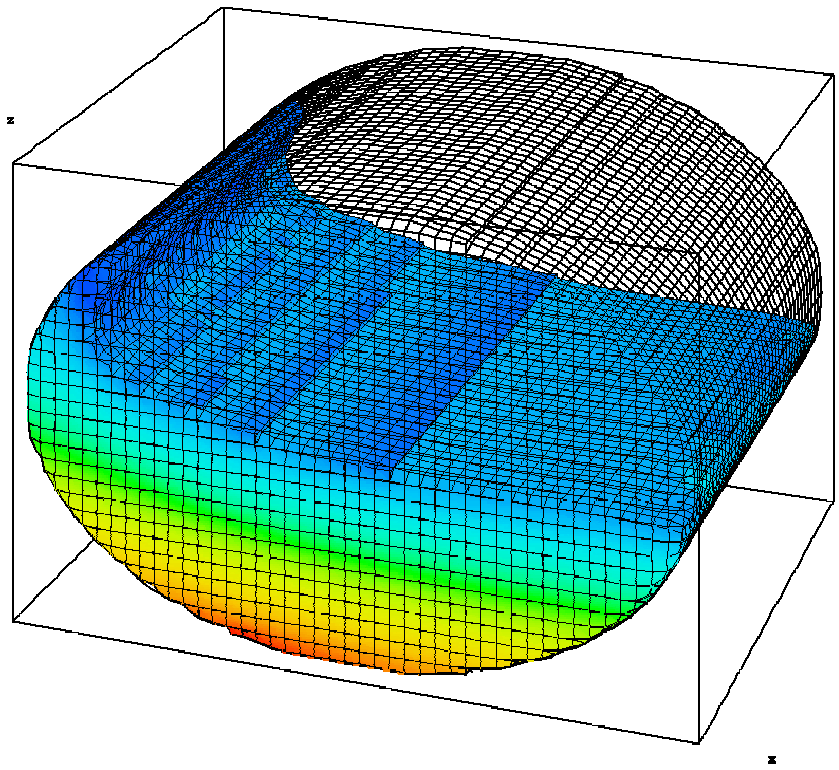


Figure 5. Phase 1 Eliptical Tank Model

Appendix B describes installation, compilation and execution of the Flow-3D model.

3.3 Fluent Model of Phase 1 Slosh Tank

The model geometry and setup are identical to that of the Flow-3D model. The basic model was created by the Jacman Group (Appendix C) to provide an access point for the input and output data. SwRI engineers completed implementation by updating and expanding the user defined functions to use CoSim communication pipes. The following Fluent subroutines were implemented for CoSim:

- *CCCutils_cosim.c*: Contains utility subroutines for pipe communication.
- *MotionForces.c*: Fluent User Defined Functions (UDFs) with subroutines for managing data transfer of CoSim data.

Integration of CoSim subroutine into Fluent was not completed in this phase of work.

4.0 RESULTS OF COUPLED SIMULATION

4.1 Prediction of Phase 1 Slosh Load

Results of the CoSim runs have been compared to the test data collected in Phase 1. The Phase 1 final report should be referenced for a complete description of the test fixture and testing procedures[1]. Results from each of the six load cases are shown to agree well with Phase 1 test data and the level of correlation is nearly identical to that of Phase 1. For this reason an exhaustive comparison of the results is not presented in this report. Appendix D shows a comparison of each load case for Phase 1 test, Phase 1 CFD results and Phase 2 test fixture simulation results. The main purpose of these comparisons is to demonstrate that CoSim between ADAMS and Flow-3D has been properly developed and is able to duplicate Phase 1 results.

Comparison with Phase 1 results has demonstrates the successful development of the CoSim concept such that a vehicle simulation (using ADAMS) can be coupled to a tank slosh simulation (using Flow-3D, see Figure 6). The simulations are correctly synchronized through the Glue Code. The limit of this demonstration (as it was with Phase 1) is that the motion of the tank is known *a prior*. As a result the vehicle simulation is kinematic. That is to say the vehicle has zero degrees of freedom where the motion is rigidly applied based on known and measured test data. The ultimate challenge is to apply CoSim using a model with flexible components (like a suspension system) and predict both the load from tank slosh and the motion of the vehicle that is influenced by those tank slosh loads.

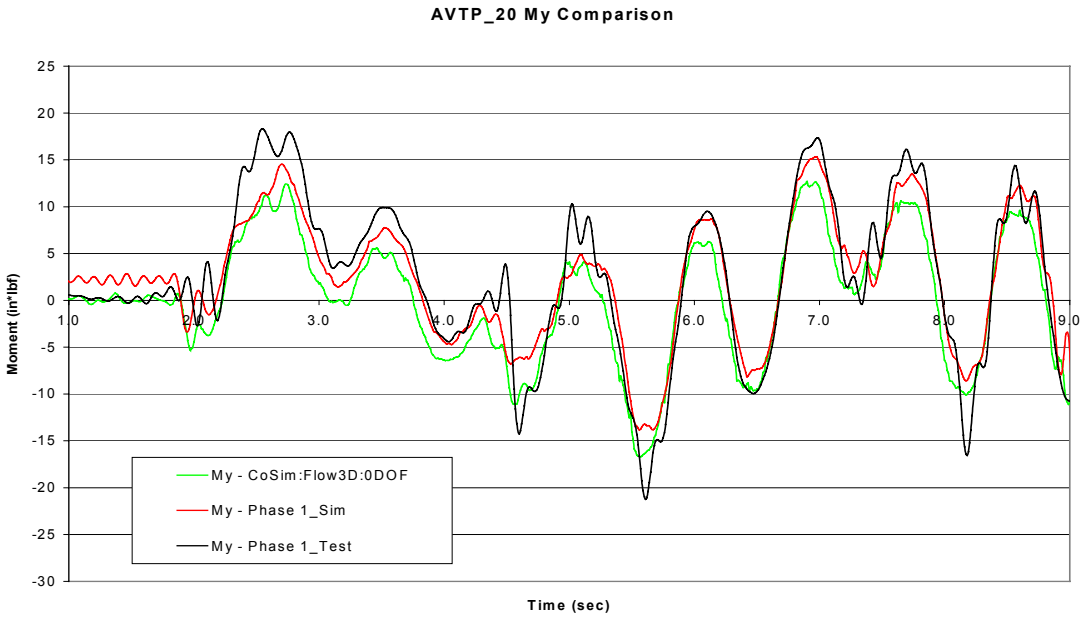


Figure 6. Spherical Tank Model

[1] Wendel, G. R, Green, S. T., Burkey, R. C., "Coupled Simulation of Vehicle Dynamics and Tank Slosh, Phase 1 – Final Report," Interim Report TFLRF No. 364, U.S. Army, TACOM, Warren, Michigan, July 2002

4.2 Prediction of Dynamic Slosh Load

Experimentation with the elliptical tank model was performed to observe the effects of adding dynamic degrees of freedom to the mechanical (MSC.ADAMS) model. This resulted in unexpected behavior that is illustrated here through a simplified model of a spherical tank (see Figure 7). This model simulated a spherical tank on the end of a centrifuge arm that translates radially while constrained by a spring.

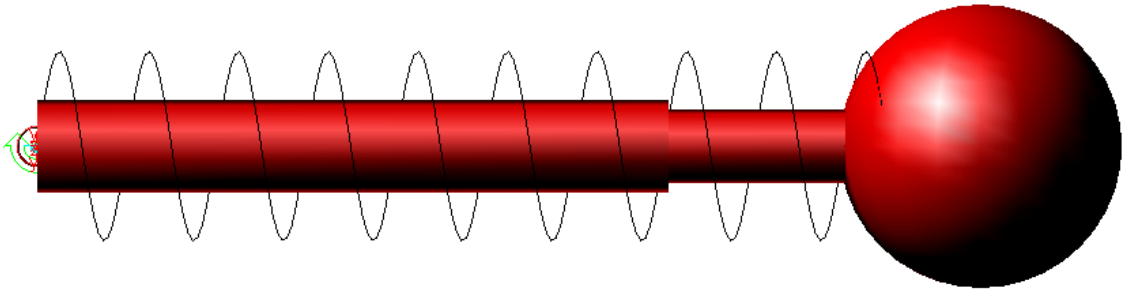


Figure 7. Spherical Tank Model

The concept of this simulation is not much different than swirling a glass of water in your hand. The system begins at rest and accelerates to a constant angular velocity within the first 45º of rotation. When the mass of the tank is small relative to the mass of the fluid the solution quickly becomes unstable because the fluid force instantly creates a proportional displacement of the spring (no damping present). This discrete (rather than smooth) motion effectively destabilizes the CFD solution. We know intuitively that the force of fluid on tank will equalize rapidly as the tank begins to move with the fluid. However, the MSC.ADAMS solution is ignorant of how the fluid force should drop off when such large perturbations are called for. This illustrates the case where the glue code extrapolator method cannot keep up with the simulation. When a significant tank mass is used in the simulation the result is initially more stable; however, the solution eventually becomes unstable as shown in Figure 8. The solution fails halfway through the second full rotation as fluid resonance couples with the period of the centrifuge. Fluid motion is relatively smooth until the large oscillations prior to solution failure. A relatively minor fluid slap event precipitated the uncontrolled motion.

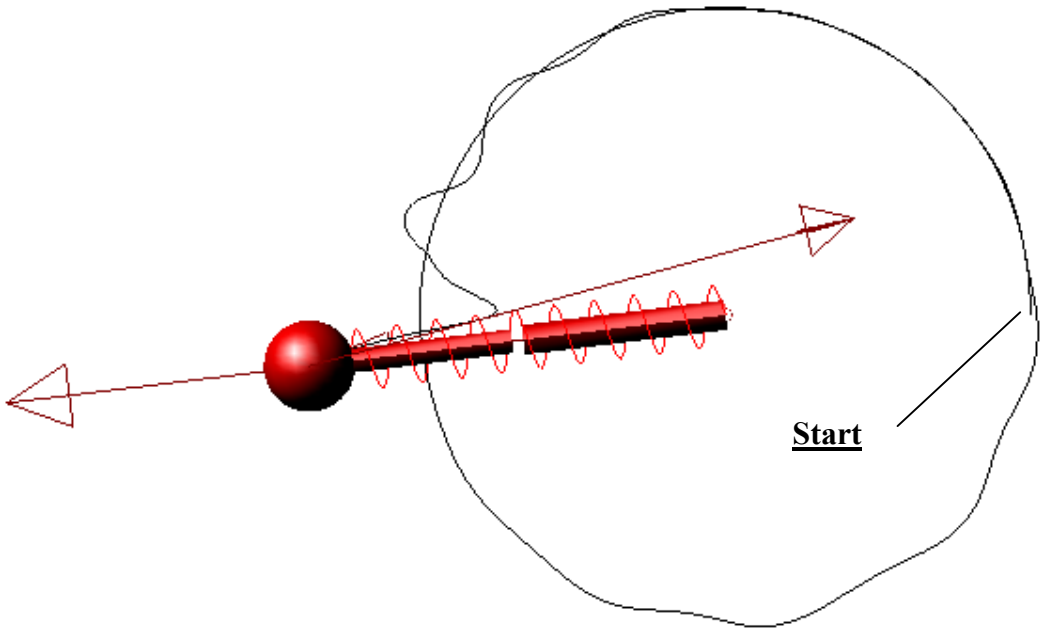


Figure 8. Path of the Spherical Tank

A theoretical and experimental investigation is planned but has not been accomplished at the time of this writing. It is therefore not possible to fully explain root cause of this phenomenon. It is possible that the extrapolation method and small time increments will be sufficient to accurately simulate the class of problems that are the focus of this research effort. However, any solution that involves relatively long duration simulations and relatively large fluid mass (proportions as yet unknown) will certainly require additional computation.

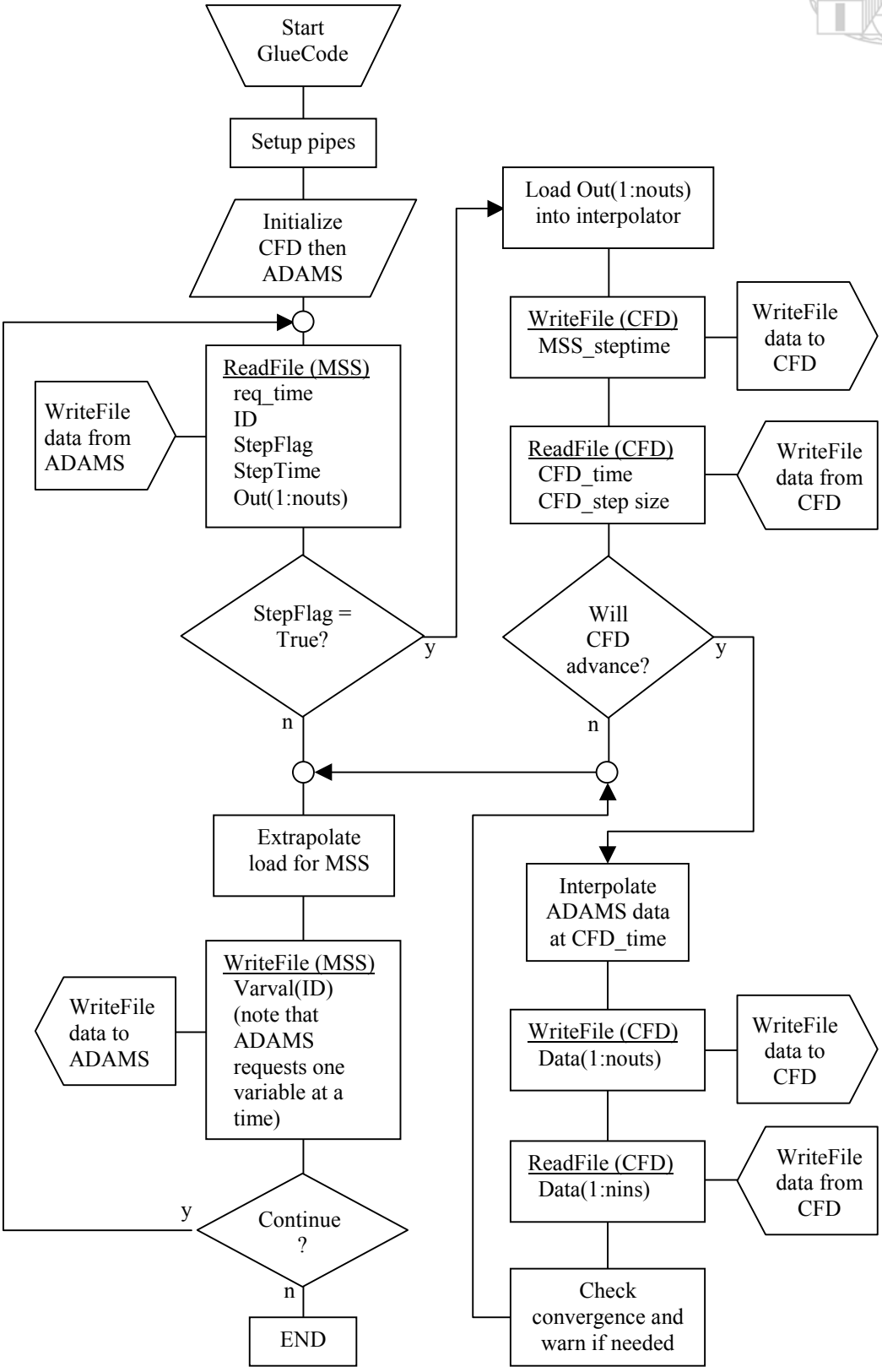
5.0 CONCLUSION

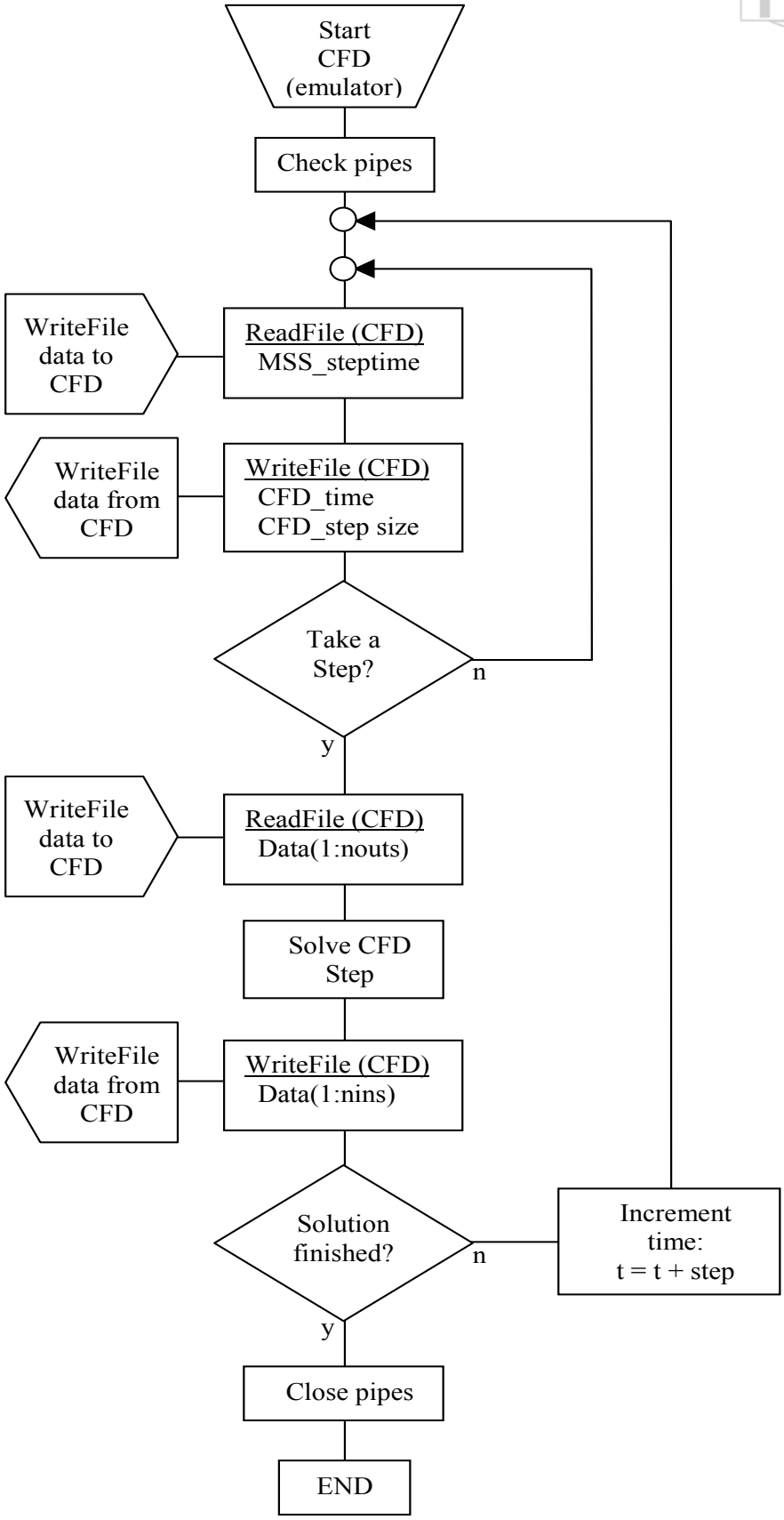
Research engineers at SwRI have implemented co-simulation between MSC.ADAMS and CFD slosh models using Flow-3D. Precise application of complex motion input from MSC.ADAMS was validated by correlation of test and simulation data. Simulations are generally robust; however, experimentation with the parameters of a simplified dynamics model indicates that further research is required.

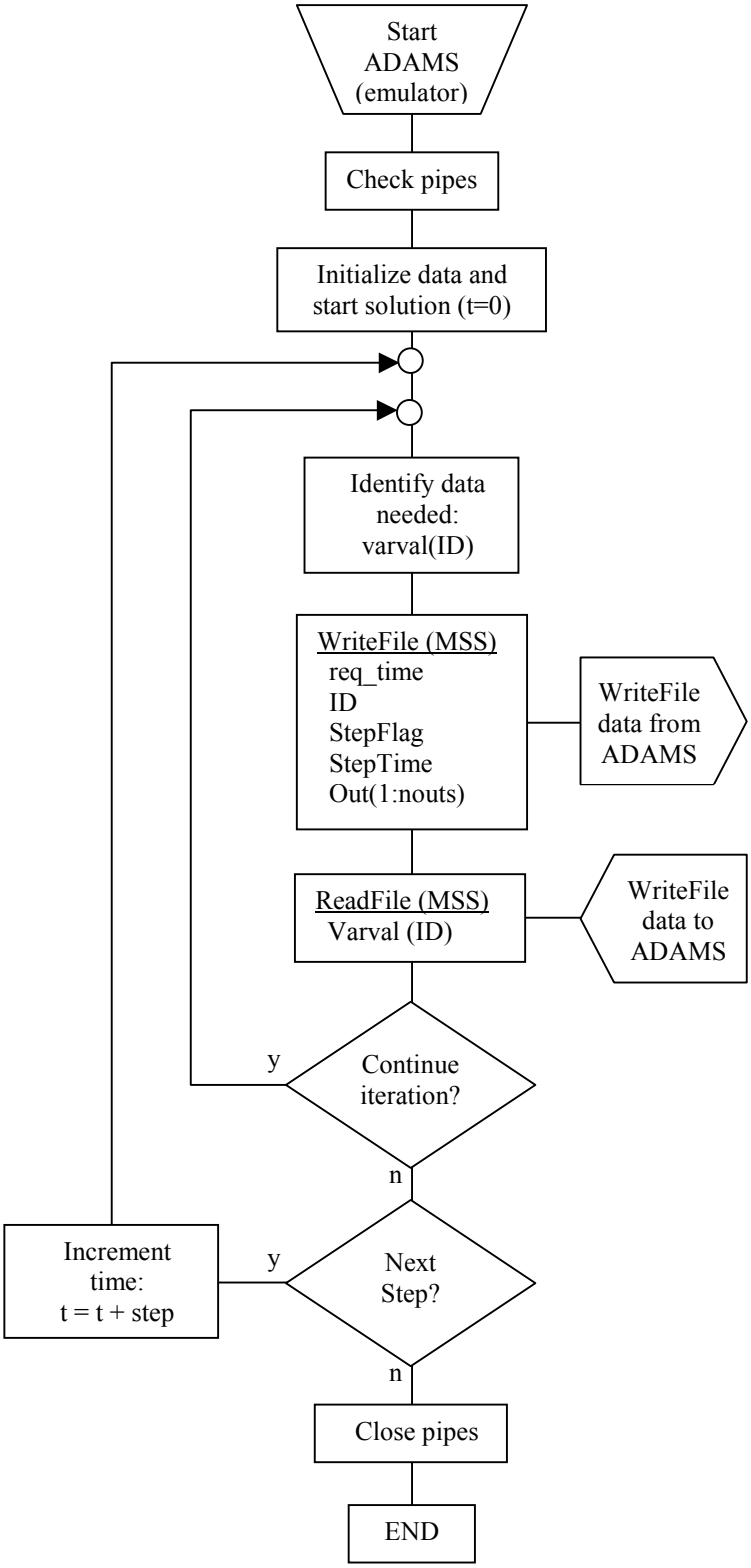
Further work is planned to investigate the consequences of co-simulation of the dynamic nature of slosh on a mechanical system. This work will include the development of a theoretical and/or empirical solution that will be validated in a small-scale test. We will continue to develop and implement full-scale vehicle models that include co-simulation with CFD and experimentally validate these models with field or laboratory test data.

APPENDIX A

Flow Charts of the Glue Code and CoSim Subroutines


Start GlueCode
Setup pipes
Initialize CFD then ADAMS
ReadFile (MSS)
req_time
ID
StepFlag
StepTime
Out(1:nouts)
WriteFile data from ADAMS
StepFlag = True?
y
n
Load Out(1:nouts) into interpolator
WriteFile (CFD)
MSS_steptime
WriteFile data to CFD
ReadFile (CFD)
CFD_time
CFD_step size
WriteFile data from CFD
Will CFD advance?
y
n
Extrapolate load for MSS
Interpolate ADAMS data at CFD_time
WriteFile (MSS)
Varval(ID)
(note that ADAMS requests one variable at a time)
WriteFile data to ADAMS
WriteFile (CFD)
Data(1:nouts)
WriteFile data to CFD
ReadFile (CFD)
Data(1:nins)
WriteFile data from CFD
Check convergence and warn if needed
y
Continue ?
n
END


Start
CFD
(emulator)
Check pipes
WriteFile
data to
CFD
ReadFile (CFD)
MSS_steptime
WriteFile
data from
CFD
WriteFile (CFD)
CFD_time
CFD_step size
Take a
Step?
n
y
WriteFile
data to
CFD
ReadFile (CFD)
Data(1:nouts)
Solve CFD
Step
WriteFile
data from
CFD
WriteFile (CFD)
Data(1:nins)
Solution
finished?
n
Increment
time:
t = t + step
y
Close pipes
END


Start
ADAMS
(emulator)
Check pipes
Initialize data and
start solution (t=0)
Identify data
needed:
varval(ID)
WriteFile (MSS)
req_time
ID
StepFlag
StepTime
Out(1:nouts)
WriteFile
data from
ADAMS
ReadFile (MSS)
Varval (ID)
WriteFile
data to
ADAMS
y
Continue
iteration?
n
Increment
time:
t = t + step
y
Next
Step?
n
Close pipes
END

APPENDIX B

Summary of CoSim on the PC

There are 6 projects in Developer Studio (Visual C++ 6.0 SP5 and Compaq Visual FORTRAN 6.6b) workspace (inex64chkt, EmulateCFD, EmulateMSS, hydr3d, prep3d, and sloshsubs).

1) Glue Code
   a) Project is 'inex64chkt'
   b) Source includes inex64chkt.f, interp2.f and pipesetup.f
   c) Project options for correct compile are:
      /compile_only /nologo /warn:nofileopt /module:"./" /object:"Release/"
   d) Build the project to create inex64chkt.exe
2) Emulator of CFD code
   a) Project is 'EmulateCFD'
   b) Source includes CCCUtils_CoSim.f, Flow3DEmulator.f and Flow3DUtils.f
   c) Project options for correct compile are:
      /compile_only /nologo /warn:nofileopt /module:"./" /object:"Release/"
   d) Build the project to create EmulateCFD.exe
3) Emulator of Multi-body Structural Simulation code
   a) Project is 'EmulateMSS'
   b) Source includes ADAMSEmulator.f, ADAMSUtils.f and consub64Emulator.f
   c) Project options for correct compile are:
      /compile_only /nologo /warn:nofileopt /module:"./" /object:"Release/"
   d) Build the project to create EmulateMSS.exe
4) Flow-3D customized subroutine code (2 projects)
   a) Project 'hydr3d' is for the solver (setup based on default Flow-3D installation)
   b) Project 'prep3d' is for the preprocessor (setup based on default Flow-3D installation)
   c) Modified source code is *_cosim.f
   d) Most code is commented with "CoSim" in the modification descriptions
   e) Rebuild each project before simulation
5) ADAMS customized subroutine code
   a) Project 'sloshsubs'
   b) Source includes varsub64.f and consub64.f
   c) The files cru.f and cru.exe are not part of the Developer Studio project but must be copied to the ADAMS installation directory in the ../common folder
   d) Project options for correct compile are:
      /architecture:pn4 /compile_only /nologo /threads /warn:nofileopt /module:"./" /object:"./"
   e) Compile each file individually (do not build project)
   f) File dllsubs.lst lists the object files
   g) Create dll through ADAMS command line (dfvars must be set):
      adams12.bat cr-u n @dllsubs.lst multi64.dll
   h) Create *.acf file to control execution of the ADAMS simulation (cosim.acf is provided). Command in this file are the following for CoSim (see ADAMS help for further details):
      cosimtest1
      cosimtest1
      control/func=user(100)
      sim/dyn, end=6.1, step=610
      control/func=user(900)
      stop
   i) The ADAMS execution command line is:
      adams12.bat ru-u multi64.dll cosim.acf
6) Execute a CoSim
   a) Launch 'inex64chkt.exe' from a command prompt (Glue Code)
   b) Start Flow-3D using custom executable or launch the CFD emulator
   c) Start ADAMS from command prompt or launch the MSS emulator

APPENDIX C

Jacman Report

Slosh Modeling and Control Tool Using Fluent User-Defined Subroutines

Prepared by: Edward Bullister/The Jacman Group
Date: 3 April 2003

TABLE OF CONTENTS

1.0 BACKGROUND

This project is in support of the efforts of SwRI to develop a co-simulation tool that couples internal tank sloshing with the dynamics of external tank support structures.

This report documents work completed in the internal tank-sloshing component of this project. The sloshing is modeled by Fluent, a computational fluid dynamics (CFD) code that models the dynamic motion of the flow in response to the motion of the tank. Specifically, Fluent solves the unsteady, three-dimensional Navier-Stokes equations for the turbulent, two-phase flow of water and air in the tank.

2.0 COMPUTATIONAL GEOMETRY AND GRID

The computational geometry is an horizontal cylinder with an elliptical cross section (see Figure 1). The initial condition of the tank is the bottom half is full of water (with the top half being full of air). The geometry was generated using Gambit, the pre-processing tool supplied by Fluent that generates geometries and meshes appropriate for fluid flow problems.

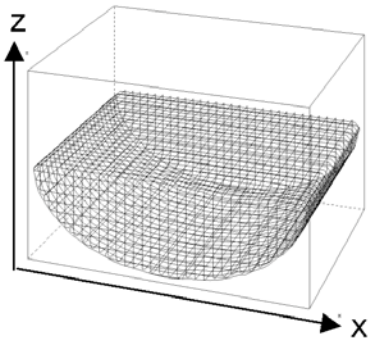
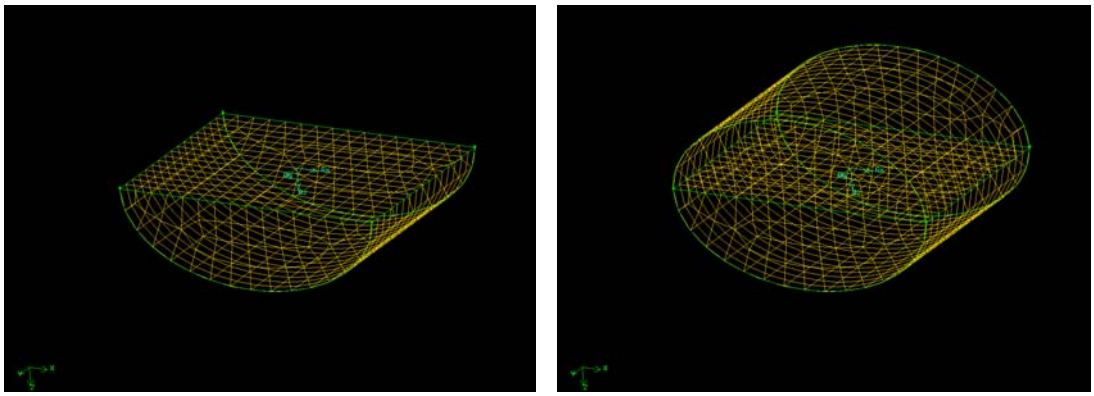


Figure 1: Above- Geometry Specification Below- Mesh Construction



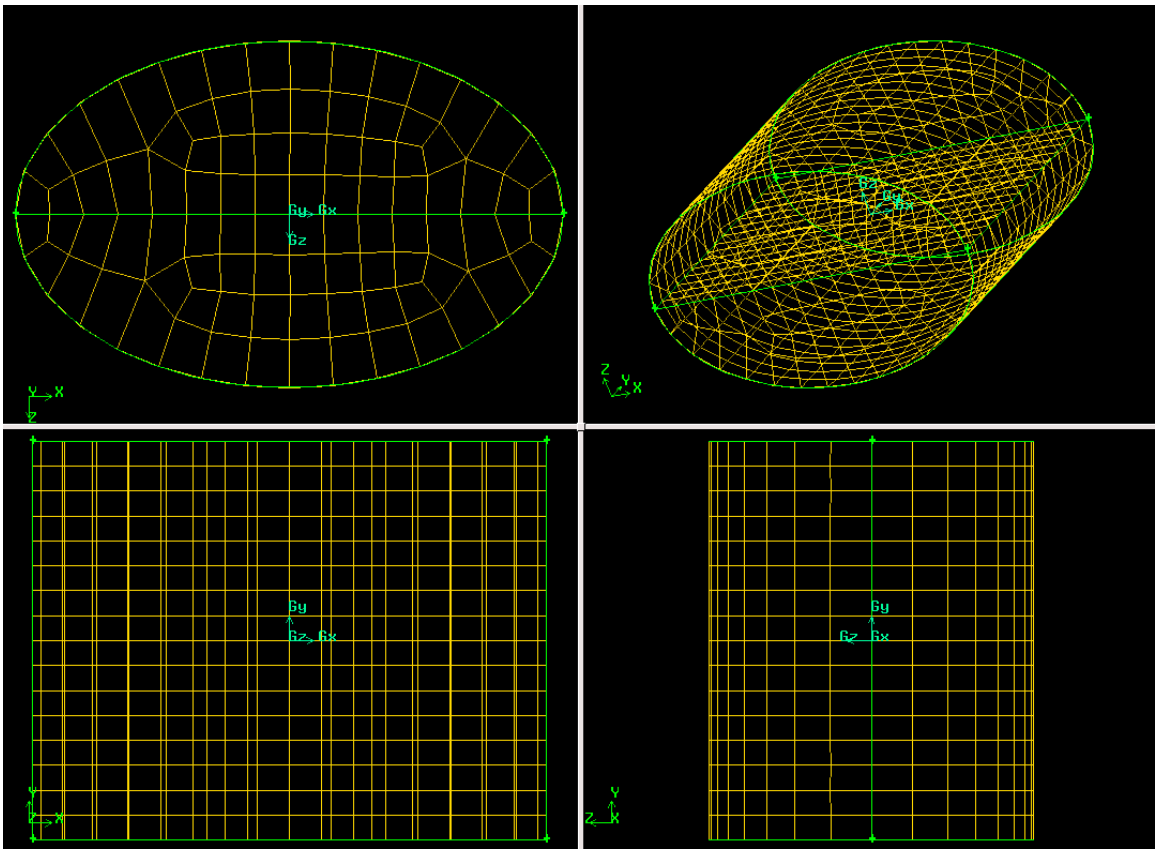


Figure 2 Computational Mesh describing Tank

The grid was generated using the Cooper method. In accordance with this scheme, a two-dimensional mesh of quadrilaterals was generated on an elliptical face on one end of the elliptical cylinder. The two-dimensional mesh was extruded in the third dimension to make layers of elements along the axis of the elliptical cylinder.

To force the paving algorithm to maintain symmetry, this was done in two steps. First, the elliptical cylinder was split in half by a horizontal plane at z=0 (see Figure 1). The Cooper method was applied to this half-cylinder. This mesh was then reflected to generate the full elliptical geometry. Finally, the redundant nodes on the reflection plane at z=0 were removed by the connection operation.

3.0 IMPLEMENTATION OF THE TANK MOTION

The direct simulation of a moving tank would require movement of all of the mesh nodes at each time step. It would also require significant additional calculations to account for convective effects of translation and rotation of the mesh. Furthermore, the classic turbomachinery capabilities of Fluent restrict the rotation vector to be constant in direction and magnitude, and a new approach with fully movable mesh would be needed. All of these issues make it undesirable to directly solve a moving body with a moving mesh.

A more efficient method was implemented. In this approach, a mesh is fixed with respect to the tank and the problem is solved in the frame of reference of the tank. When the tank undergoes acceleration, its frame of reference is no longer inertial, and the equations of motion need to account for this. A transformation of coordinates is required from the moving frame of reference to an inertial frame of reference to account for apparent forces (pseudo-forces) that arise on the bodies in the moving frame of reference.

3.1 Transformation of Coordinates

The relevant components of tank motion are:

a) The three components of linear acceleration (a_x, a_y, and a_z),
b) The three components of angular velocity (ω_x, ω_y, and ω_z), and
c) The three components of angular acceleration ($\acute{\omega}_x$, $\acute{\omega}_y$, and $\acute{\omega}_z$)

The above quantities are directly implemented in a C function (see section 3.2 below). The units of the linear accelerations are in m/sec/sec, while the angular velocities and accelerations are in radians/sec and radians/sec/sec, respectively.

Figure 3 shows the equations used for the transformation of coordinates[1]. The translation position of the tank is represented by **h**, and the rotation of the tank is represented by **ω**. Starred quantities are referenced to the moving frame of reference, so that, for example, **r** is a position with respect to the stationary (inertial) coordinates, and **r*** is the position referenced to the moving tank.

If the tank were an inertial frame of reference, equation 7.40 would have only the first term on the right hand side; the acceleration would be the same from either perspective. However, with the non-inertial frame of reference, additional acceleration terms arise in the right side of equation 7.40, corresponding to centrifugal, Coriolis, angular, and linear acceleration, respectively.

[1] Mechanics, by Keith R. Symon, Addison-Wesley Series in Physics, Reading, Ma., 1971, p 279.

$$\mathbf{r} = \mathbf{r}^* + \mathbf{h}, \tag{7.38}$$

$$\frac{d\mathbf{r}}{dt} = \frac{d^*\mathbf{r}^*}{dt} + \boldsymbol{\omega} \times \mathbf{r}^* + \frac{d\mathbf{h}}{dt}, \tag{7.39}$$

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{d^{*2}\mathbf{r}^*}{dt^2} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}^*) + 2\boldsymbol{\omega} \times \frac{d^*\mathbf{r}^*}{dt} + \frac{d\boldsymbol{\omega}}{dt} \times \mathbf{r}^* + \frac{d^2\mathbf{h}}{dt^2}. \tag{7.40}$$

Figure 3: Equations for Transformation from Non-Inertial to Inertial Coordinate System

3.2 Implementation in FLUENT

These accelerations are input in Fluent in the form of body (pseudo-) forces applied to the fluid. For convenience, the relevant C source code and comments are included below as follows:

```
/*

Above Inputs are x, y, and z components of tank acceleration, angular velocity,
and angular acceleration.  Follow right hand rule for angular rotations.

a_lin       Linear acceleration  of tank (meters/sec/sec)
omega_t     Angular velocity     of tank (radians/sec)
omegad_t    Angular acceleration of tank (radians/sec/sec)

Calculated components of acceleration of tank reference frame at each point:

a_cen = omega_t X (omega_t X r)   Centrifugal Acceleration
a_cor = 2 * omega_t X v           Coriolis    Acceleration
a_ang = omegad_t X r              Acceleration due to angular acc of tank
a_lin = [direct input]            Acceleration due to linear  acc of tank


Sum acceleration components and multiply hy -rho (in kg/m**3) to get
pseudo-force (in newtons/m**3) on fluid mix:

pseudo_f = -rho_mix * (a_cent + a_cor + a_ang + a_lin)
/*


    cross(temp_vec,omega_t,r);
    cross(a_cent,omega_t,temp_vec);  /* Centrifugal Component */

    cross(temp_vec,omega_t,v);
    scalar(a_cor,2,temp_vec);        /* Coriolis Component */

    cross(a_ang,omegad_t,r);         /* Angular Acceleration Component */

    f_pseudo[d] = -rho_mix * (a_cent[d] + a_cor[d] + a_ang[d] + a_lin[d]) ;
    return(f_pseudo[d]);
```

The input data defining the tank motion is below. In the C function, this data appears just above the equations and the input values so defined are used as input for the above equations.

```
   /*   INPUT OF ACCELERATION AND ROTATION OF TANK */

    a_lin[0] = 0.0;       /*  Linear acceleration of tank (meters/sec/sec) */
        if(CURRENT_TIME < 0.5)
        a_lin[1] = 3.0;
    else
        a_lin[1] = 0.0;
/*  a_lin[1] = 3.0 * sin(6.28*CURRENT_TIME);  */
    a_lin[2] = 9.8;

    omega_t[0] = 0.0;     /*  Angular velocity     of tank (radians/sec)   */
    omega_t[1] = 0.0;
    omega_t[2] = 0.0;

    omegad_t[0] = 0.0;   /*  Angular acceleration of tank (radians/sec/sec */
    omegad_t[1] = 0.0;
    omegad_t[2] = 0.0;

    /*   END OF INPUT OF ACCELERATION AND ROTATION OF TANK */
```

3.3 Time Stepping Issues

The simulation time is available for printout through the variable CURRENT_TIME. The CURRENT_TIME is set at the beginning of each time step. Within a given time step, approximately 10 – 20 iterations take place. Within each iteration, the pseudo-forces are applied through the getForce function and the solution variables are incrementally updated. Nonetheless, the value of CURRENT_TIME remains fixed throughout this whole iteration process. After the last iteration, at the end of the time step and at the same CURRENT_TIME, the execute_at_end function is called to output the net reaction force and torque. Only afterward, at the beginning of the next time step, is the CURRENT_TIME updated.

Based on the sloshing frequency analysis and experience with the test model (see section 4 below), the time step was set as a constant. This fixed time step can be read and printed out through the constant value of the CURRENT_TIMESTEP. A fixed time step seemed to be the simplest and best way to set the time step.

Another other option is to turn on the Adaptive Time Stepping in Fluent. This adjusts the time step to accommodate convergence criteria specific to the problem to be solved. If the integration with the dynamics package requires some more elaborate time step adjustment or SwRI is interested in some other way of time stepping, any such other time stepping control can be accommodated.

4.0 EXAMPLE CASE FOR SLOSHING TANK MODEL

Several sloshing problems were run to check the operation of the Fluent code with the user defined functions. First, an analytical estimate was calculated to estimage the frequency of tank sloshing. Second, Fluent was applied to solve the full unsteady Navier-stokes equations for the sloshing fluid flow in the tank.

4.1 Analytical Estimate of Sloshing Frequency

ESTIMATES OF NATURAL FREQUENCY OF SLOSHING IN A RECTANGULAR TANK

x y z

$a := 0.416 \cdot m \quad b := 0.324 \cdot m \quad h := 0.132 \cdot m$ Tank Dimensions

$i := 0 \quad j := 1$ Mode

$$fsq := \frac{g}{4 \cdot \pi} \cdot \sqrt{\left(\frac{i}{a}\right)^2 + \left(\frac{j}{b}\right)^2} \cdot \tanh\left[\pi \cdot h \cdot \sqrt{\left(\frac{i}{a}\right)^2 + \left(\frac{j}{b}\right)^2}\right]$$

$f := \sqrt{fsq}$

$f = 1.436\,Hz$

Ref : http://www.mscsoftware.com/support/library/conf/auc97/p02397.pdf

Figure 4: MathCAD calculation of expected slosh frequencies

Figure 4 shows the results of an estimate of the frequency of the sloshing within the tank. The first mode in the y-direction (j=1) was chosen to the natural frequency of the first mode of oscillation in the y-direction. Because the x and y dimensions of the tank are approximately the same, the frequency of the first mode in x should be similar. Higher modes yield higher harmonics that will be of higher frequency.

The estimate of the frequencies present in the sloshing can be useful in defining a time stepping strategy in Fluent. The estimate can also give a check to corroborate the Fluent simulation predictions.

4.2 Fluent Setup of Simulation of Tank Sloshing

Based on the above analysis of a 1.4 Hz primary slosh frequency, the time step was set to a fixed value of 0.01 seconds. The subsequent Fluent simulation confirmed that the primary slosh frequency was in fact in this range.

A rule of thumb is that, at the very least, 10 time steps should be used to accurately capture a transient event. The 0.01-second time step should resolve well for the 1.436 Hz natural frequency of the first mode and several of its harmonics. Unless the mounting/control system introduces unexpectedly higher frequency components into the system, a fixed time step set to resolve the desired frequency components is probably the most robust. If a fixed time step is for some reason not desirable, another time stepping method can be accommodated.

4.3 Results

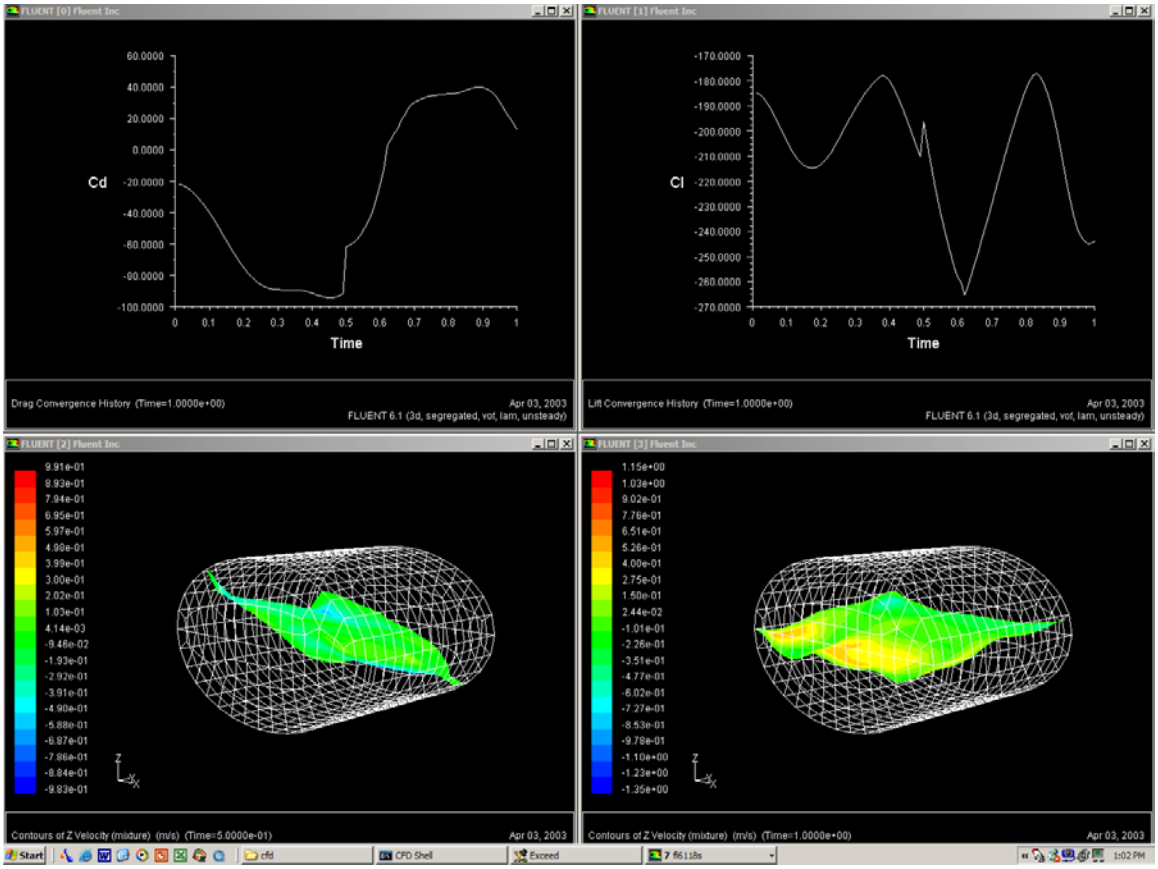


Figure 5: Screen Shot of Fluent Simulation with 0.3 G's of Acceleration in the y-direction applied to the tank for the first 0.5 seconds.

Top Left: Coefficient of Y- Force applied by the fluid to the tank. Note that the frequencies of oscillation are consistent with the predicted values for a rectangular tank.

Top Right: Coefficient of Z- Force applied by the fluid to the tank. Note that the consistently negative value for coefficient of lift. This is due to the downward gravitational pull.

Bottom Left: z-velocity plotted on water surface at T=0.5 sec. Note the buildup of water at the back (-y) end of the tank in response to a forward (+y) acceleration.

Bottom Right: z-velocity plotted on water surface at T=1.0 sec. Note that by T=1.0 sec, the water in the back (-y) of the tank has sloshed back and the level is slightly higher in the forward (+y) portion of the tank.

5.0 OUTPUT OF THE RESULTANT NET FORCES AND TORQUES

The six components of force and torque applied to the tank by the fluid are calculated by integrating the pressure over all tank walls. The results are printed at the end of each time step. The details are in the C function execute_at_end. The CURRENT_TIME and CURRENT_TIMESTEP are also output in execute_at_end.

6.0 LOGISTICS IN RUNNING THE TANK MODEL IN FLUENT

The code was developed on a Windows platform using Fluent 6.1. The example simulations can be run from a command shell window via the tank.flu journal file:

```
fluent 3d –i tank.flu
```

The geometry was generated using the Fluent geometry and mesh generator Gambit. The steps used in the geometry and mesh generation are recorded through the journal file tank.jou.

The current tank.flu file sets up real-time plotting of the time history of the Coefficient of lift, labeled Cl and Coefficient of drag, labeled Cd. This is shown in Figure 5. Theses plotted values are proportional to the forces in z and y, respectively. They give a qualitative indication of the motion of the flow, and give the frequencies of oscillation. For example, the negative “lift” coefficient corresponds to the gravitational force in the negative z direction.

7.0 DESCRIPTION AND DOCUMENTATION OF FILES

7.1 Tank.c

This is a C language source code file that contains all of the user-defined functions (UDFs) that are executed by Fluent. These functions include:

1) getForce: The function that uses the user-defined tank motion to set up the pseudo-forces acting on the fluid.
2) execute_at_end: The function executed at the end of each time step that integrates the force and torque components and prints the net force and torque on the tank.
3) momentum_x, momentum_y, and momentum_z: intermediate functions that provide hooks between getForce and the Fluent solver
4) cross and scalar: vector manipulation functions

7.2 tank.flu

Tank.flu is a journal file that contains all of the information for setting up the fluent run. The command

```
fluent 3d –i tank.flu
```

typed into a console window will cause the example problem to run.

7.3 tank.jou

tank.jou is a journal file that contains all of the information for setting up the Gambit run to define the mesh. From within Gambit, the file->run journal menu will bring up a dialog box that will input tank.jou to generate the mesh. The “intervals” parameter can be used to adjust the mesh spacing. It has been run at 10 and 16 rows of cells.

7.4 tank.cas, tank.dat

These are files produced by fluent that store the problem (case) definition and output data.

7.5 tank.msh, tank10.msh, tank16.msh

Tank10.msh and tank16.msh are two meshes output by Gambit that have 10 and 16 cells across, respectively. To change resolutions, the desired file is selected and copied over tank.msh, which is the filename referenced by Fluent. After it is copied, Fluent can be run.

The accuracy was improved by fixing an incorrect handling of boundary cells in Fluent. Fluent was dropping 50% of the force in the cells adjacent to the wall boundary. For low- and moderate- resolution runs, this had a significant impact (15-25%) on the accuracy. A workaround was implemented that corrects the error. With this correction, even coarsest mesh (tank10.msh, having only 4 vertical layers of cells) gives accurate force calculations.

8.0 APPENDIX: WORK INSTRUCTION FORM

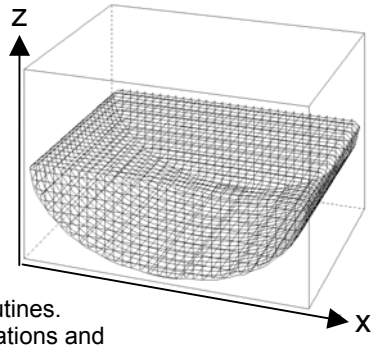
DIVISION 03
WORK INSTRUCTION FORM

WI No.: 4912 **Hardware Deliverable ☐ Yes ☒ No**

DATE ISSUED: 24 March, 2003	**REQUESTED COMPLETION DATE: 7 APR 2003**	**REQUIRED COMPLETION DATE: 7 APR 2003**	**INITIATOR: FERN THOMASSY**

Charge No.: n/a	**Government Project Yes ☒ No ☐**
	Sponsor: (Optional)
Person Recommended: (Optional) **Assigned to: Jacman Group**	**Distribution: (Optional)**
Brief Task(s) Description **(sample or test item): Fluent Model and Subroutines**	Allocated Resources (Optional): **$$7020 Hours** **(notify Initiator prior to exceeding)**

Work Instructions



Statement of work:

1) Build model of an elliptical tank that is half full of water using Fluent.
   a) Use SI units (kg, m, s)
   b) Fluid properties: density=1000 kg/m^3, dynamic viscosity=1e-3
   c) Ellipse equation: $22.9845*x^2+57.2785*z^2=1$
   d) Tank length is 0.324-m.
   e) Orientation: See figure, however, (0,0,0) is at geometric center.
2) Implement motion control (non-inertial reference frame assumed) through subroutines. Control inputs should be functions in the subroutine. Assume that linear accelerations and rotational velocities will be defined as control inputs to the subroutine. Simulation time and current step size must be available in the motion subroutine.
3) Implement 'live' output of total reaction load (forces and torques) about the reference frame. Output should be implemented as write statements in a user subroutine. Time of the 'successful' step should also be output.

Goal: The final product will be a working model where motion input and force output is communicated through the subroutines during solution. SwRI will use these subroutines as a basis for communication in co-simulation with an external code that will model structural dynamics of the tank (tank is rigid but attach to flexible structure). Fluent is given motion control from the dynamics code and returns reaction load.

Assumptions: It is assumed that the current time and the maximum next step size are communicated in the motion control subroutine. Please document clearly the meaning of all time variables and where in the step/increment cycle each input and output variable belongs.

Page 1 of ____1_____

Authorized By (Initiator): ___Fern___ **Completed By: _____**

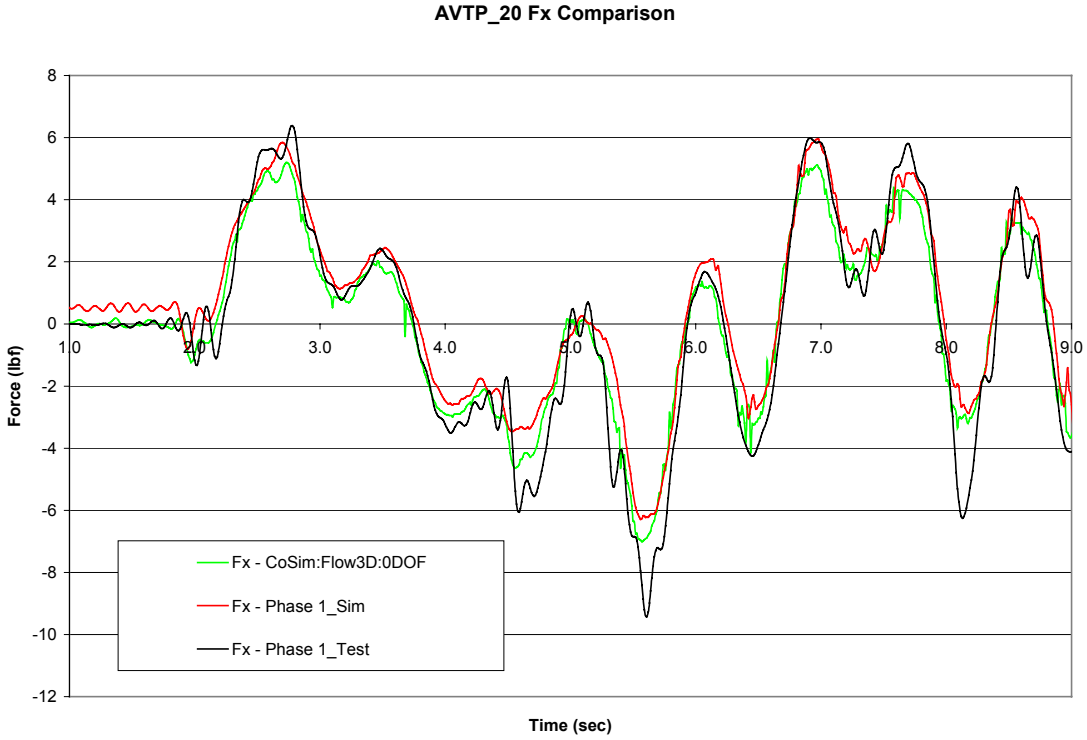
Completion Approved By (Initiator):_____ **Completion Date: _____**

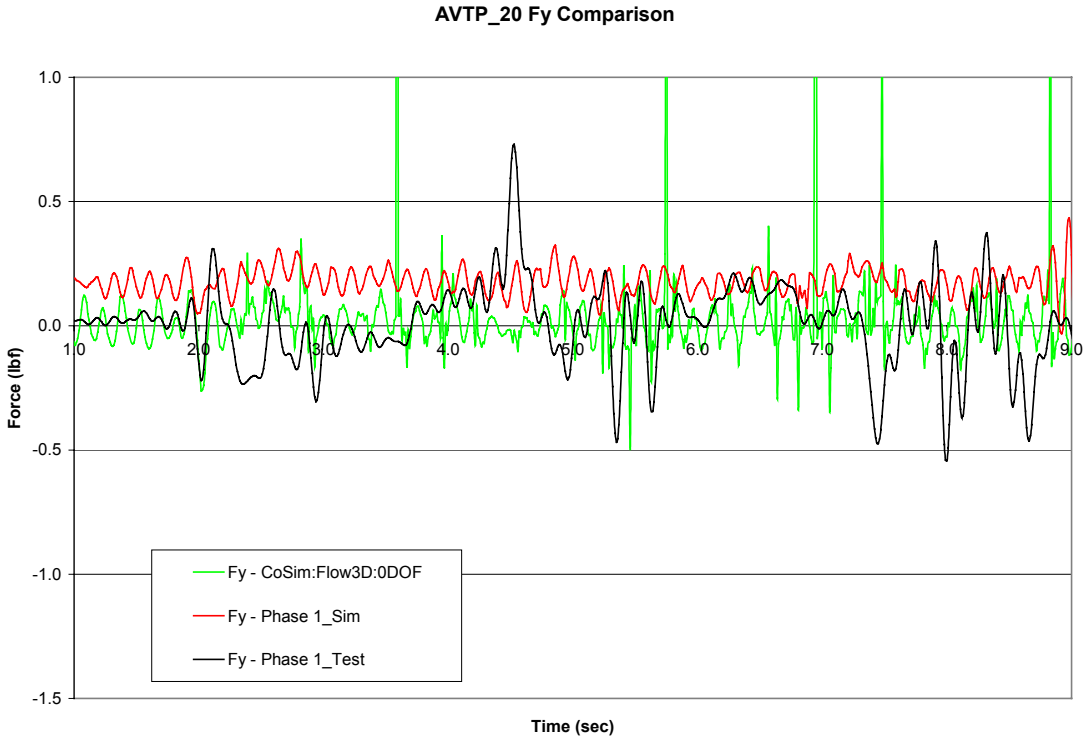
The "Yellow Copy" of this Work Instruction must be signed, dated, and returned to the Initiator.

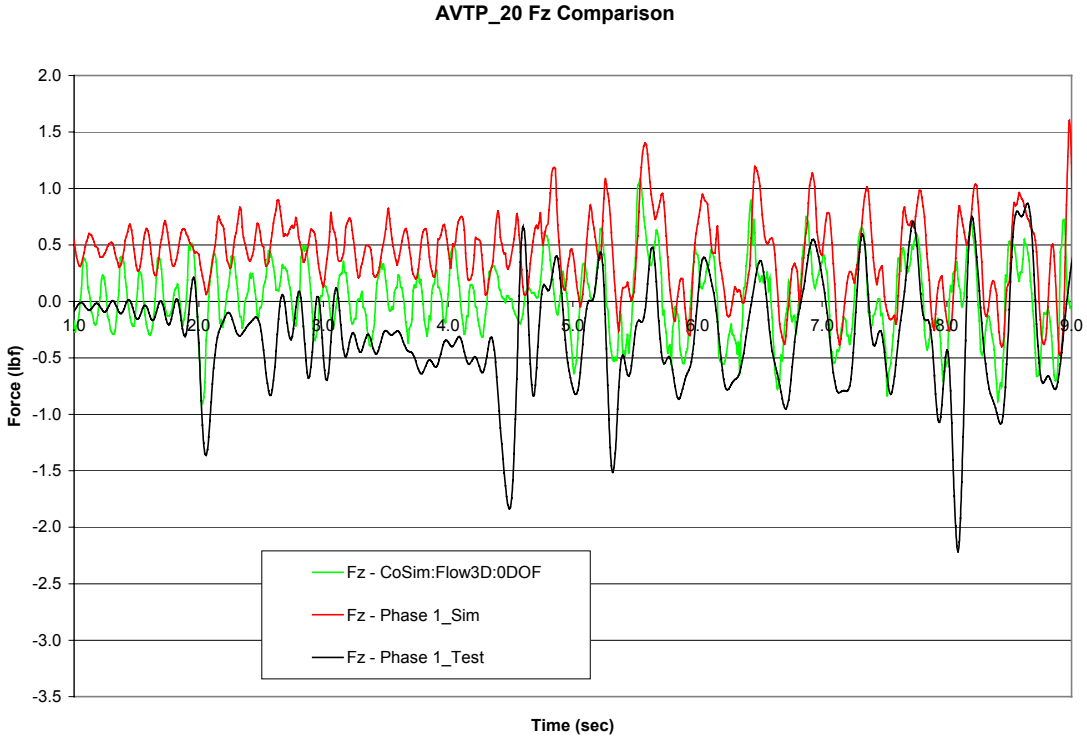
Form 5.5-1, Revision 7 (6/28/01)

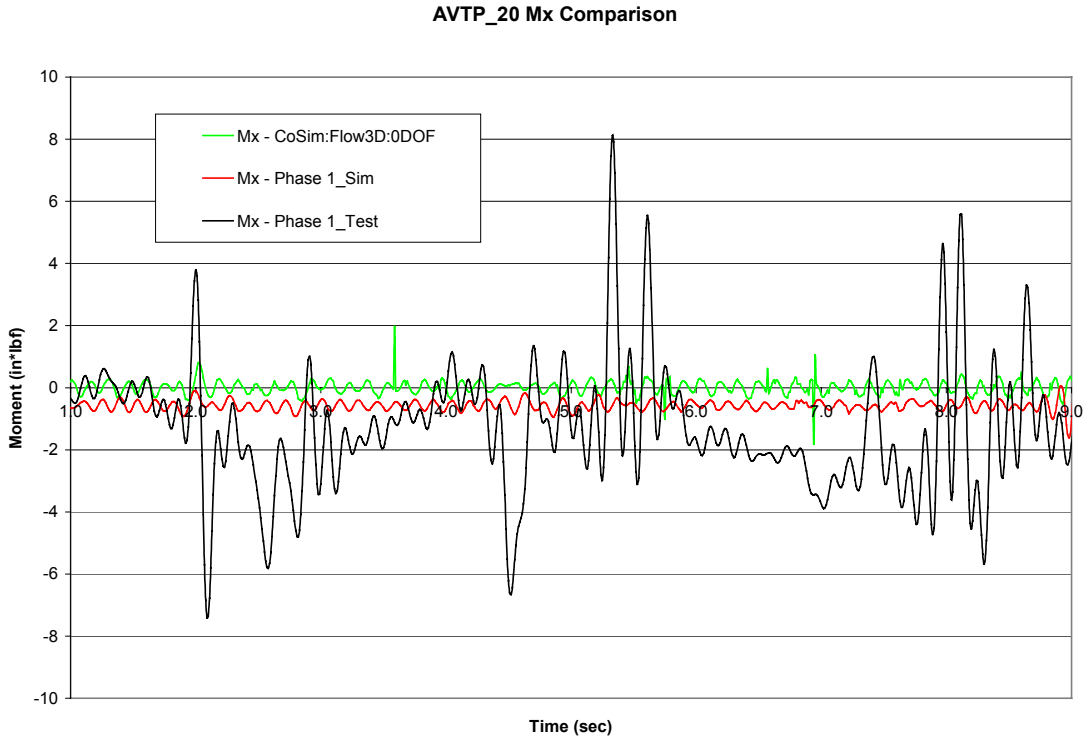
APPENDIX D

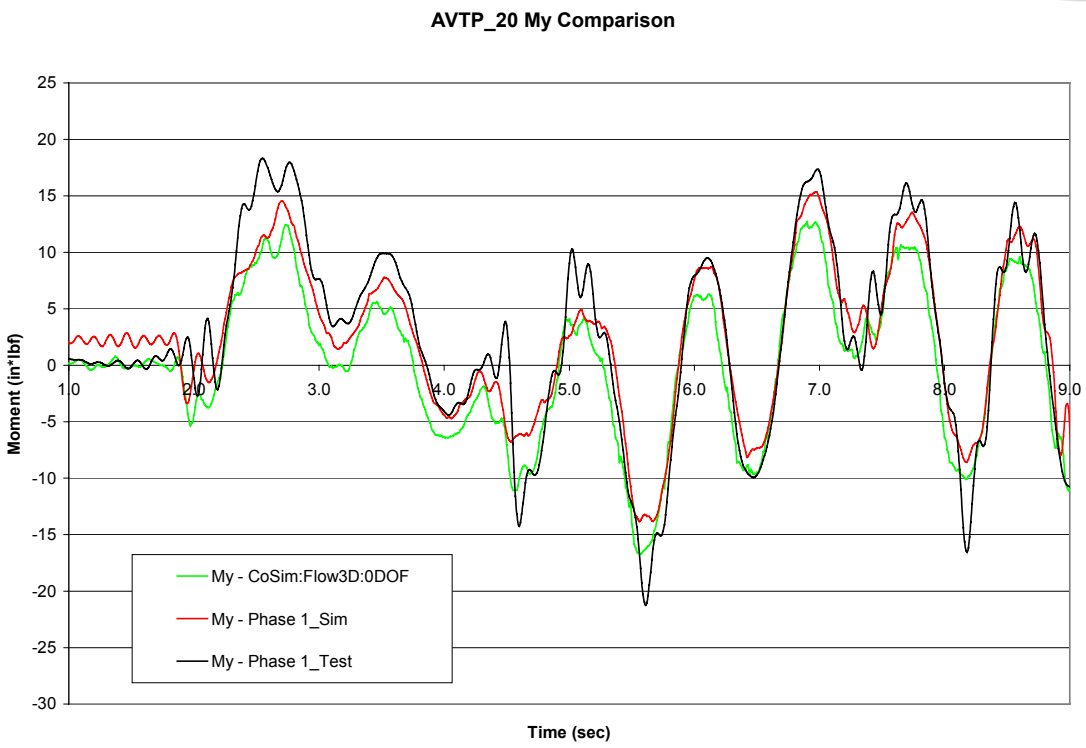
Results versus Test

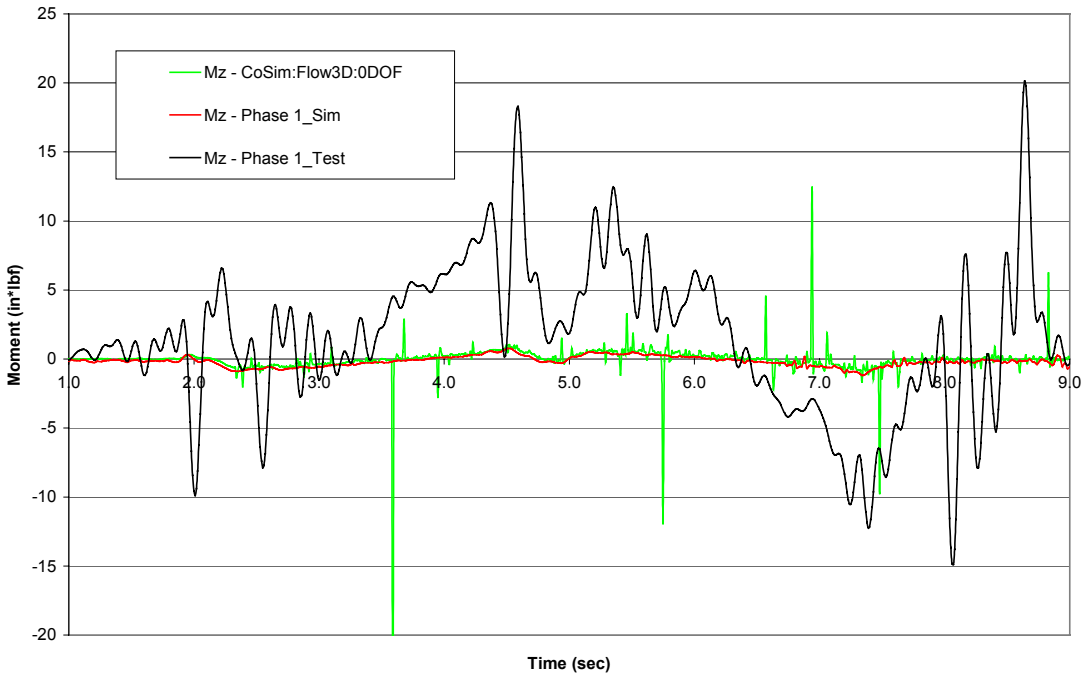

AVTP_20 Fx Comparison
Force (lbf)
Time (sec)
8
6
4
2
0
-2
-4
-6
-8
-10
-12
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Fx - CoSim:Flow3D:0DOF
Fx - Phase 1_Sim
Fx - Phase 1_Test

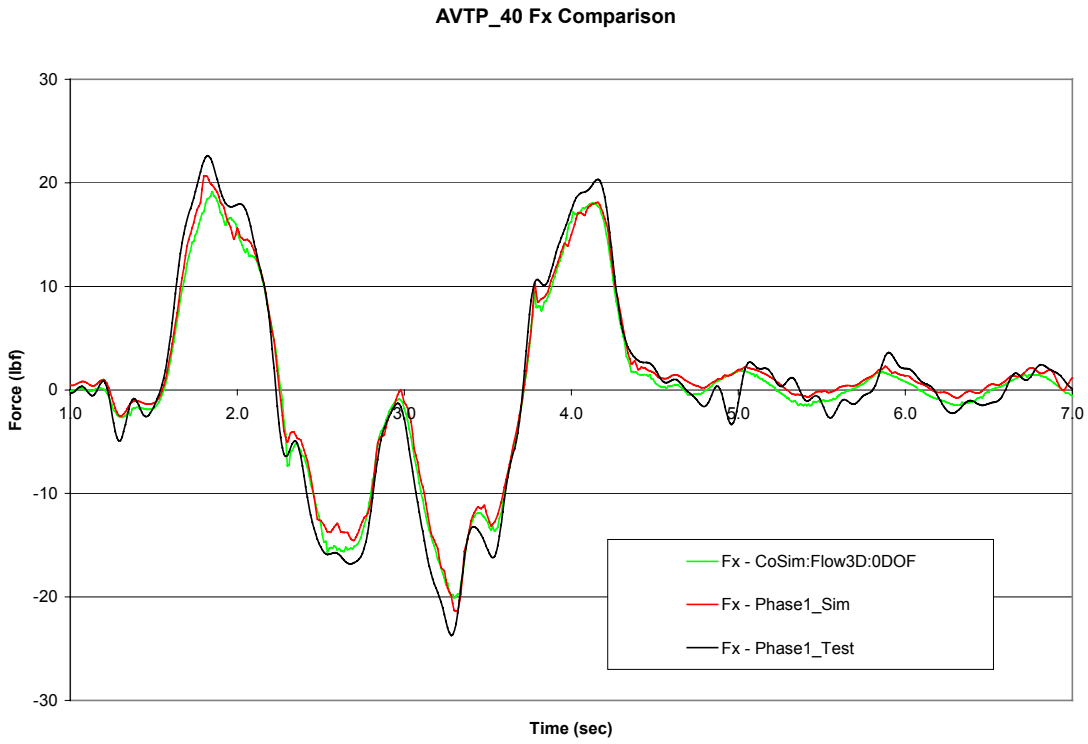

AVTP_20 Fy Comparison
Force (lbf)
Time (sec)
1.0
0.5
0.0
-0.5
-1.0
-1.5
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Fy - CoSim:Flow3D:0DOF
Fy - Phase 1_Sim
Fy - Phase 1_Test

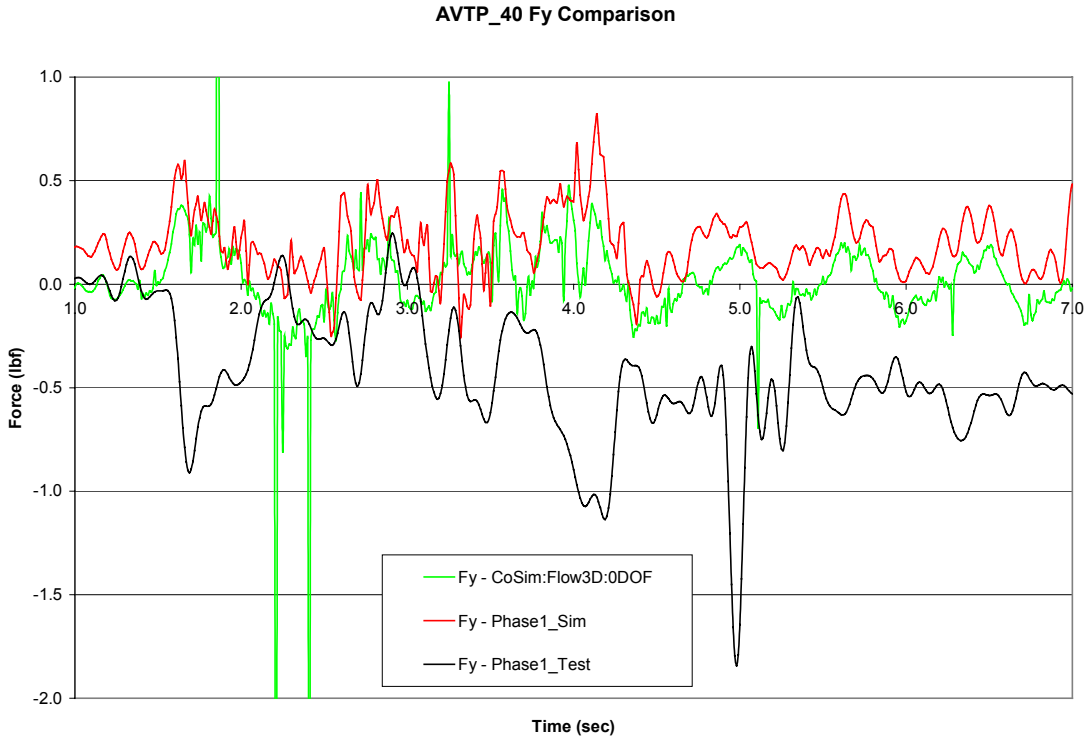

AVTP_20 Fz Comparison
Force (lbf)
Time (sec)
Fz - CoSim:Flow3D:0DOF
Fz - Phase 1_Sim
Fz - Phase 1_Test

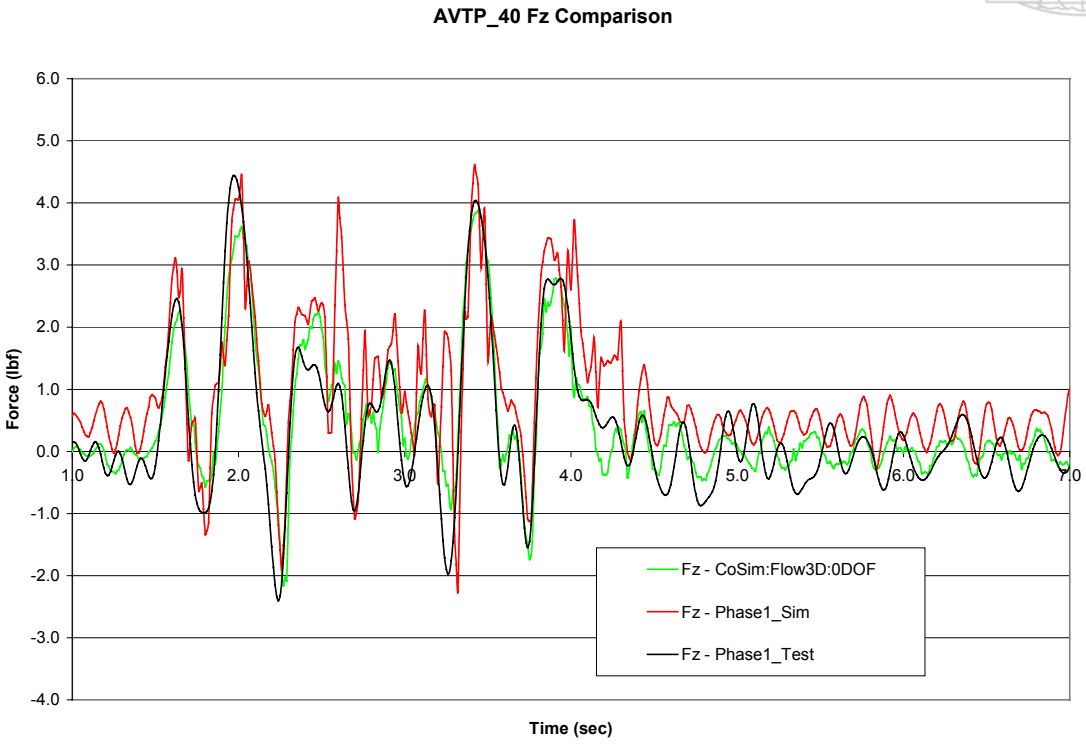

AVTP_20 Mx Comparison
Moment (in*lbf)
Time (sec)
Mx - CoSim:Flow3D:0DOF
Mx - Phase 1_Sim
Mx - Phase 1_Test

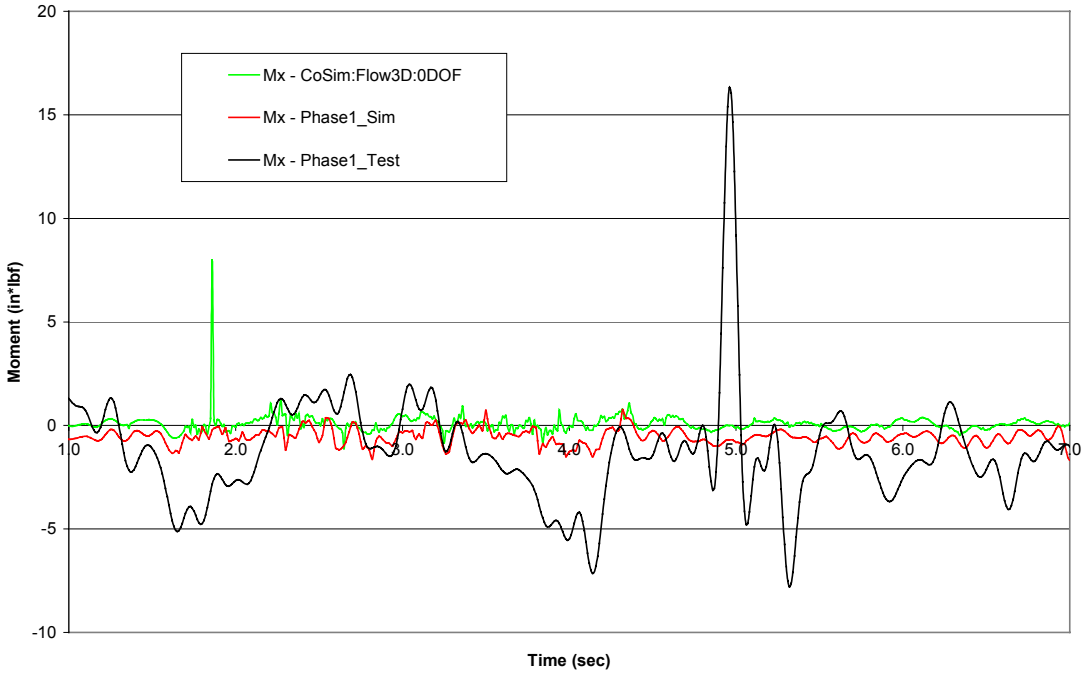

AVTP_20 My Comparison
Moment (in*lbf)
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase 1_Sim
My - Phase 1_Test

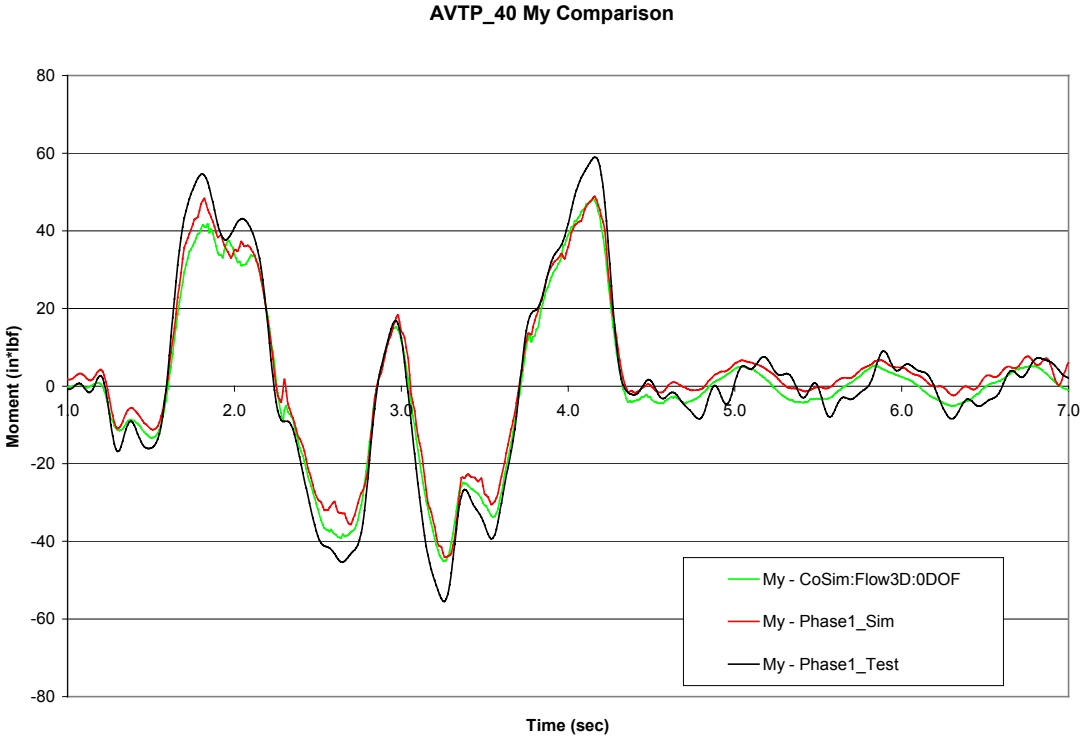

AVTP_20 Mz Comparison
Moment (in*lbf)
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase 1_Sim
Mz - Phase 1_Test

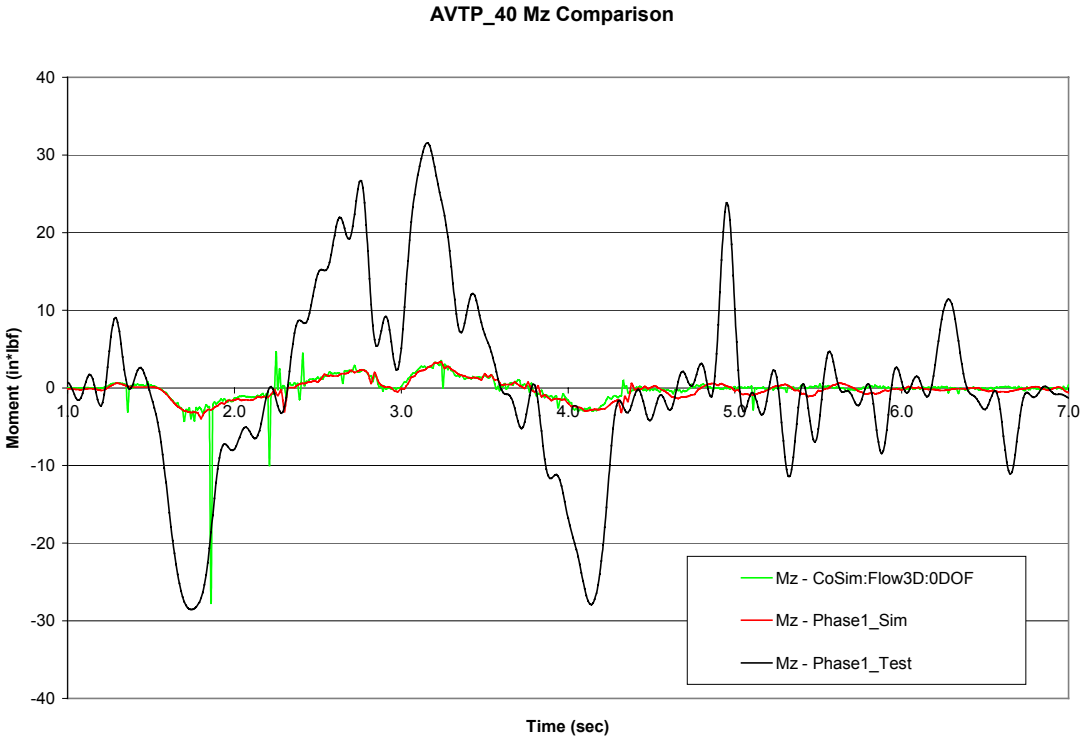

AVTP_40 Fx Comparison
30
20
10
0
-10
-20
-30
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Force (lbf)
Time (sec)
Fx - CoSim:Flow3D:0DOF
Fx - Phase1_Sim
Fx - Phase1_Test

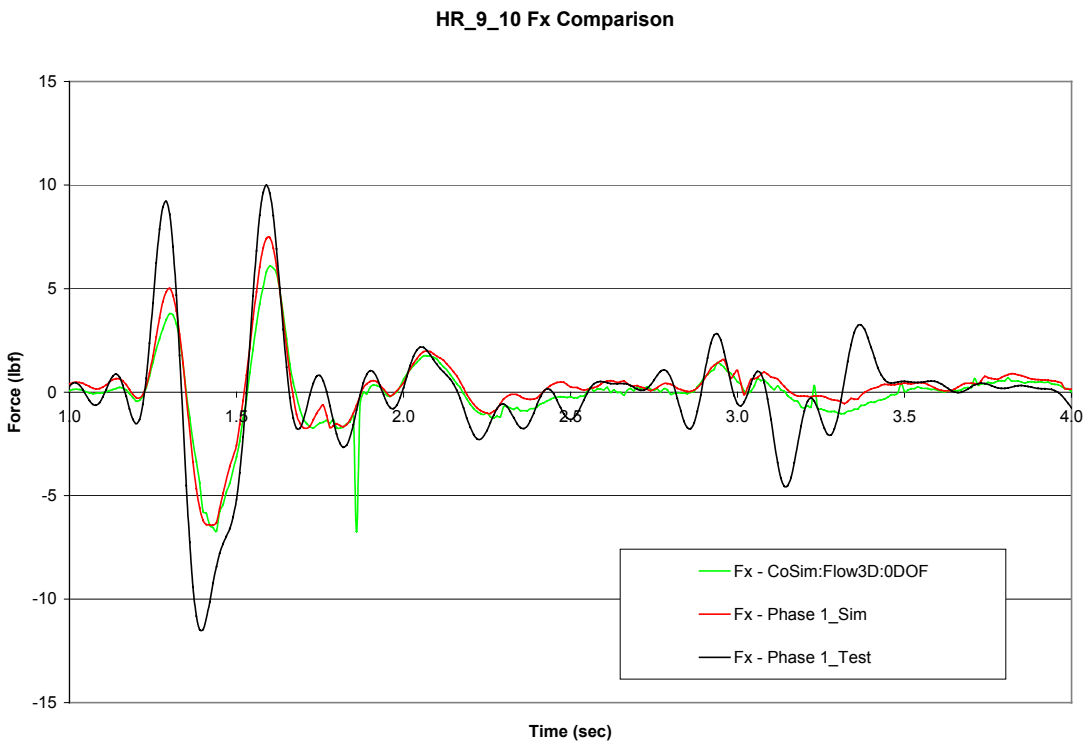

AVTP_40 Fy Comparison
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Force (lbf)
Time (sec)
Fy - CoSim:Flow3D:0DOF
Fy - Phase1_Sim
Fy - Phase1_Test

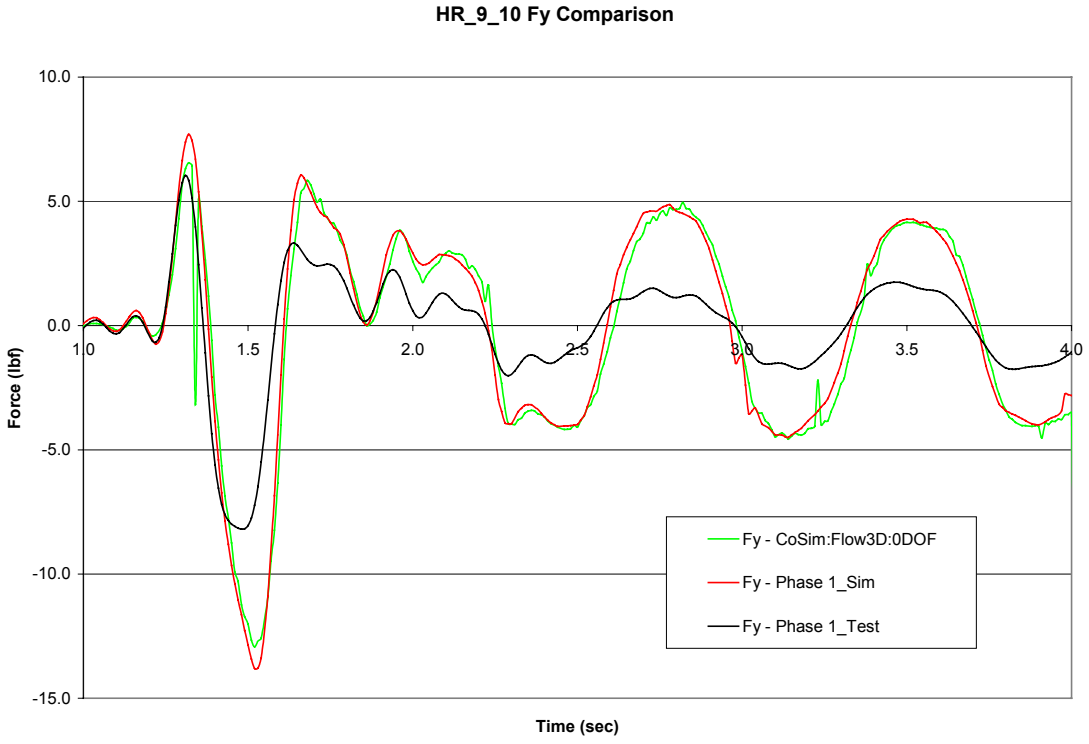

AVTP_40 Fz Comparison
6.0
5.0
4.0
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-4.0
Force (lbf)
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Time (sec)
Fz - CoSim:Flow3D:0DOF
Fz - Phase1_Sim
Fz - Phase1_Test

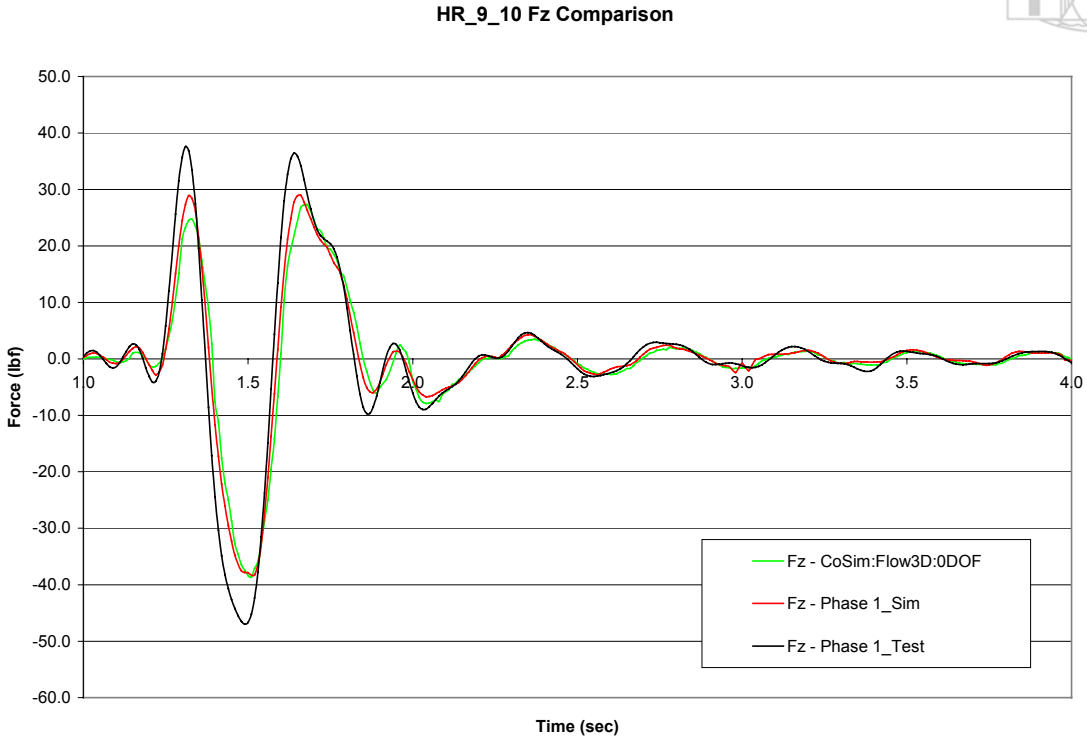

AVTP_40 Mx Comparison
20
15
10
5
0
-5
-10
Moment (in*lbf)
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Time (sec)
Mx - CoSim:Flow3D:0DOF
Mx - Phase1_Sim
Mx - Phase1_Test

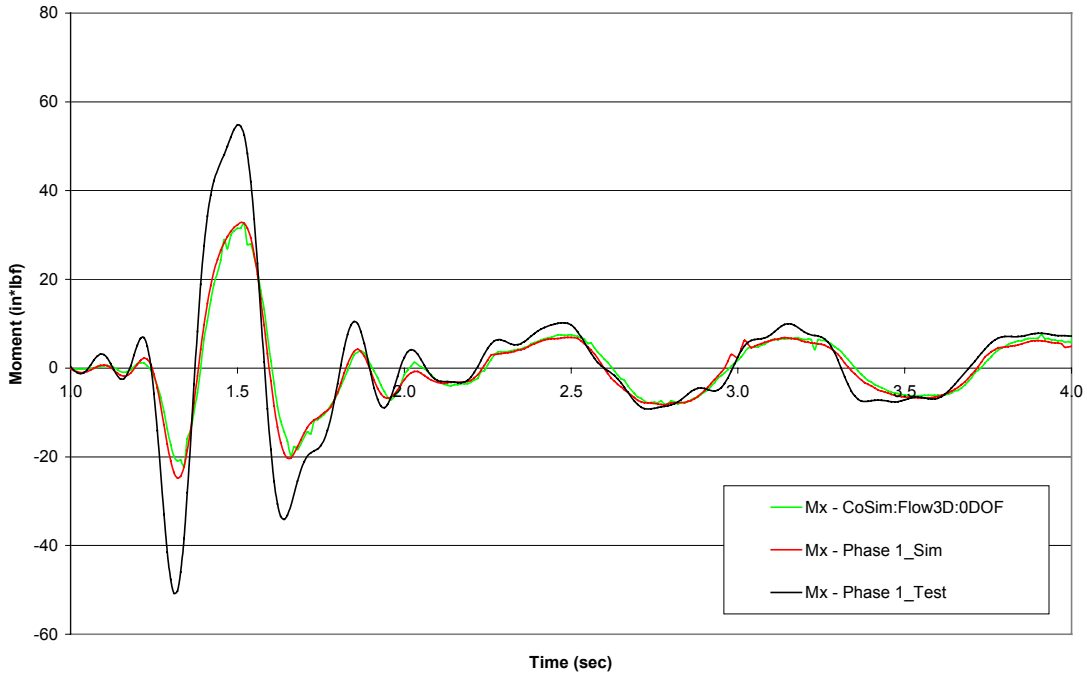

AVTP_40 My Comparison
80
60
40
20
0
-20
-40
-60
-80
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Moment (in*lbf)
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase1_Sim
My - Phase1_Test

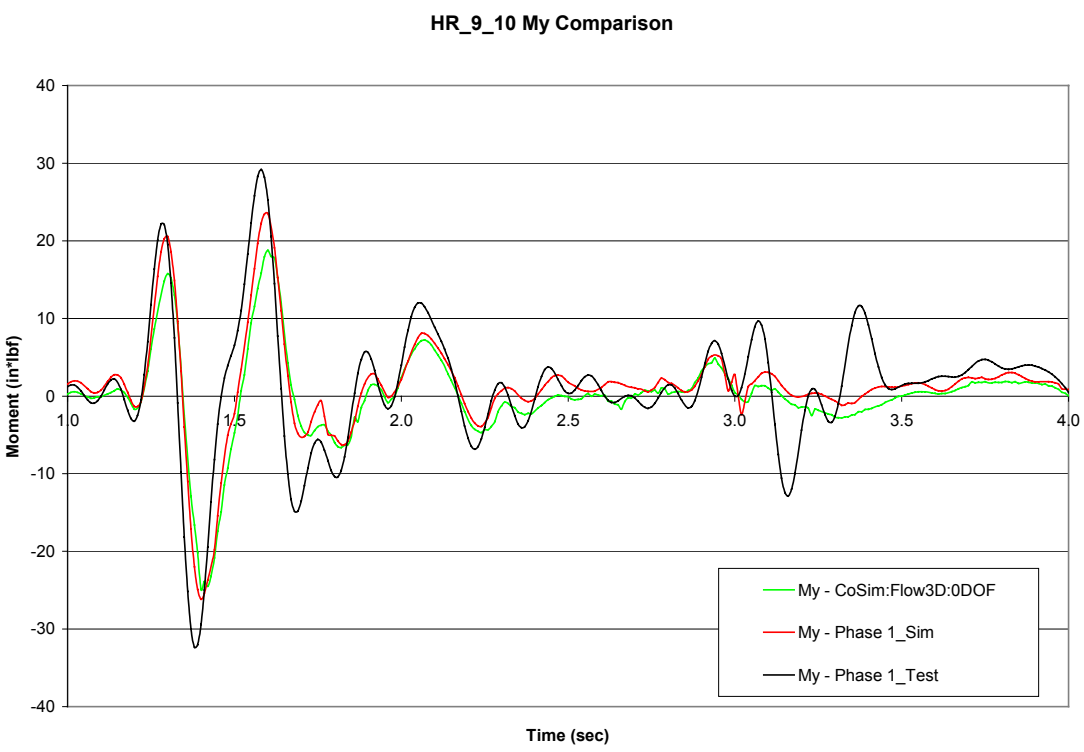

AVTP_40 Mz Comparison
40
30
20
10
0
-10
-20
-30
-40
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Moment (in*lbf)
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase1_Sim
Mz - Phase1_Test

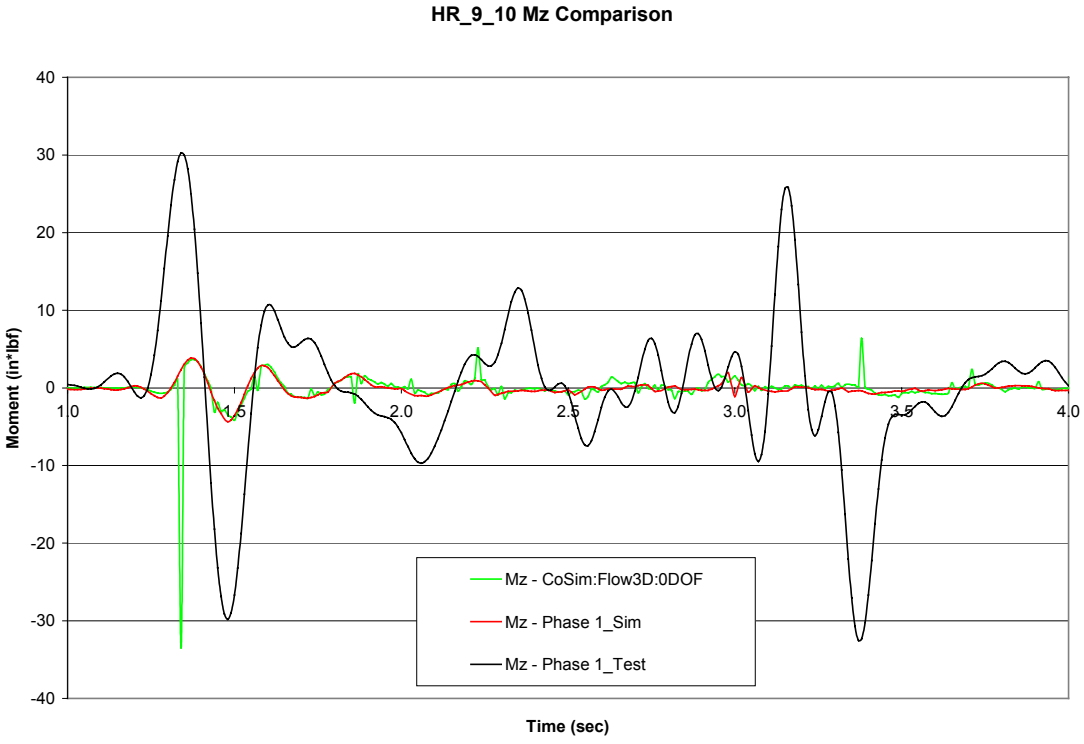

HR_9_10 Fx Comparison
15
10
5
0
-5
-10
-15
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Force (lbf)
Time (sec)
Fx - CoSim:Flow3D:0DOF
Fx - Phase 1_Sim
Fx - Phase 1_Test

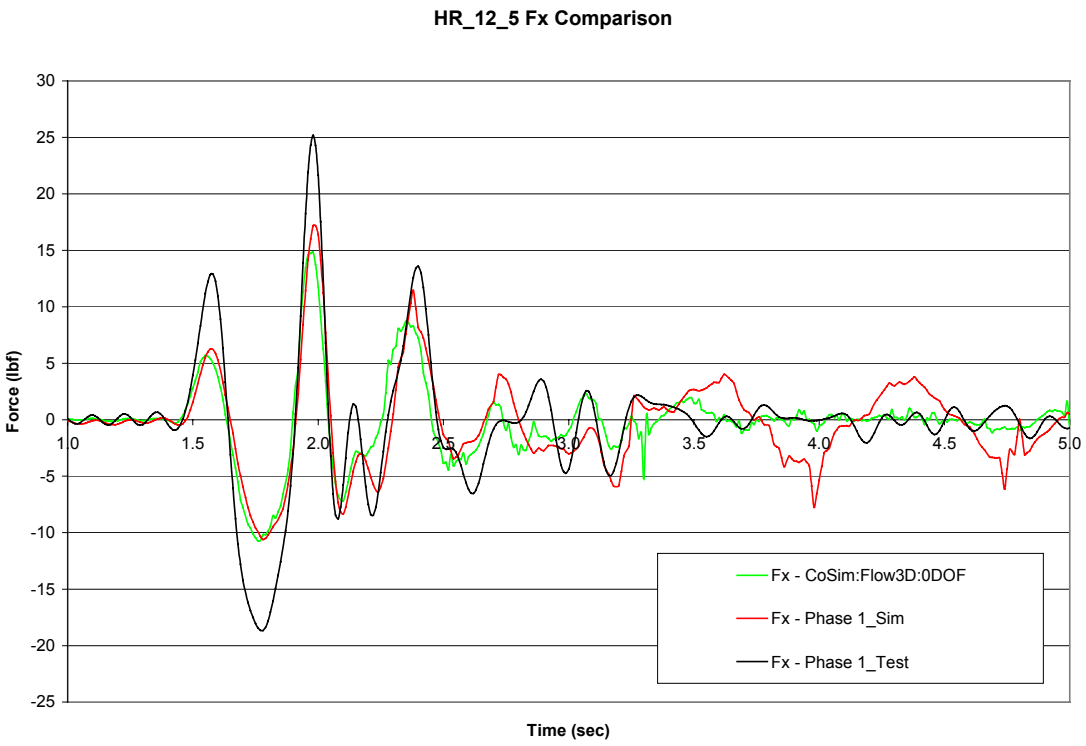

HR_9_10 Fy Comparison
10.0
5.0
0.0
-5.0
-10.0
-15.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Force (lbf)
Time (sec)
Fy - CoSim:Flow3D:0DOF
Fy - Phase 1_Sim
Fy - Phase 1_Test

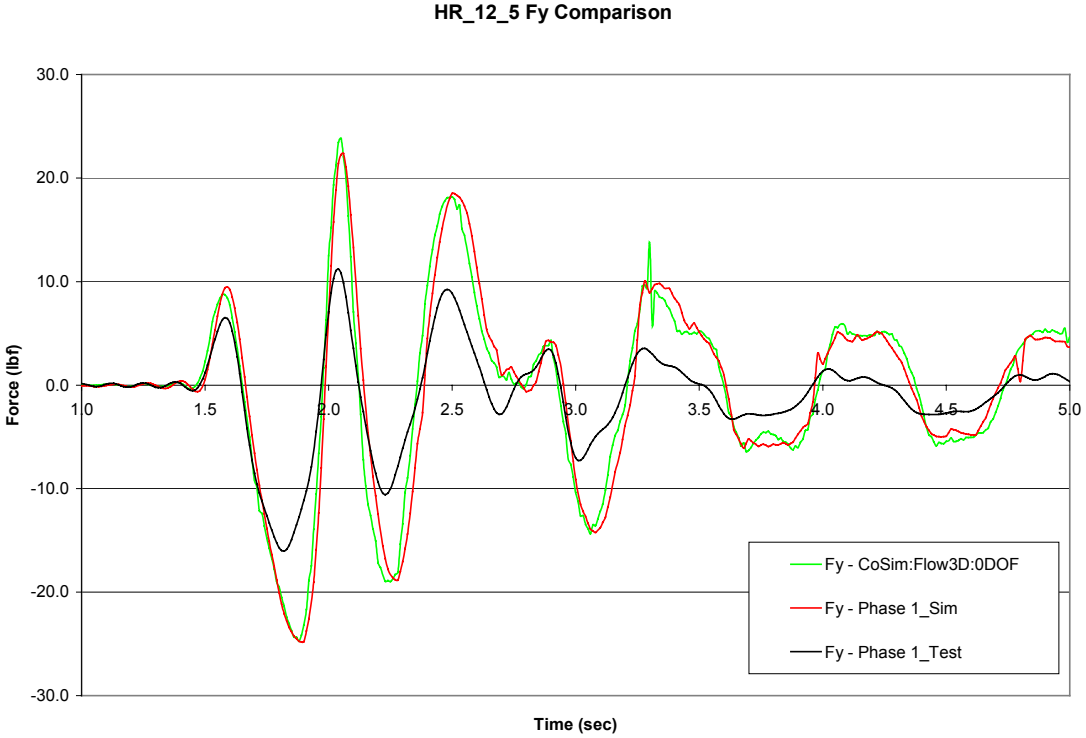

HR_9_10 Fz Comparison
50.0
40.0
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
-40.0
-50.0
-60.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Force (lbf)
Time (sec)
Fz - CoSim:Flow3D:0DOF
Fz - Phase 1_Sim
Fz - Phase 1_Test

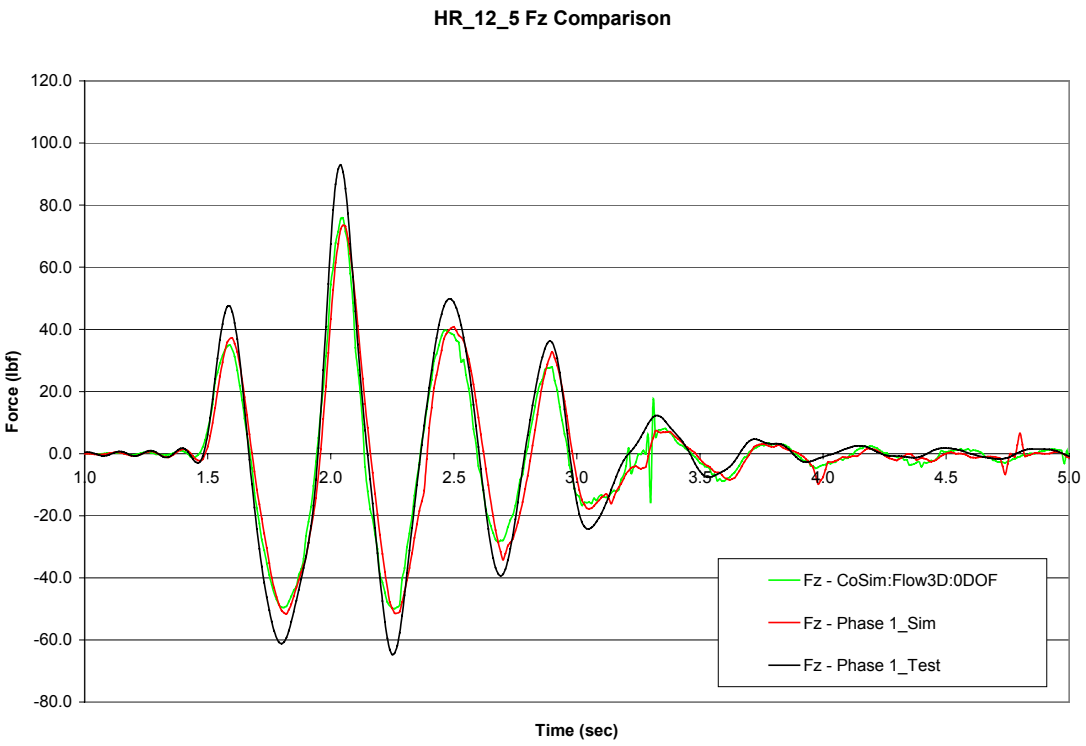

HR_9_10 Mx Comparison
80
60
40
20
0
-20
-40
-60
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Moment (in*lbf)
Time (sec)
Mx - CoSim:Flow3D:0DOF
Mx - Phase 1_Sim
Mx - Phase 1_Test

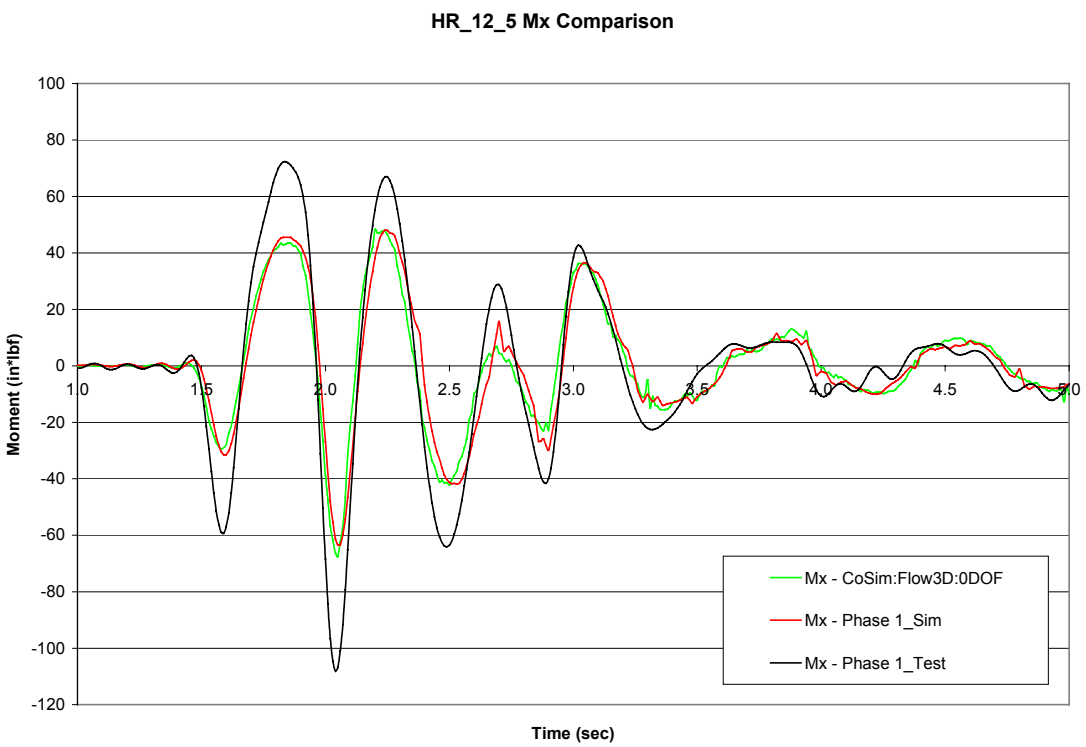

HR_9_10 My Comparison
Moment (in*lbf)
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase 1_Sim
My - Phase 1_Test

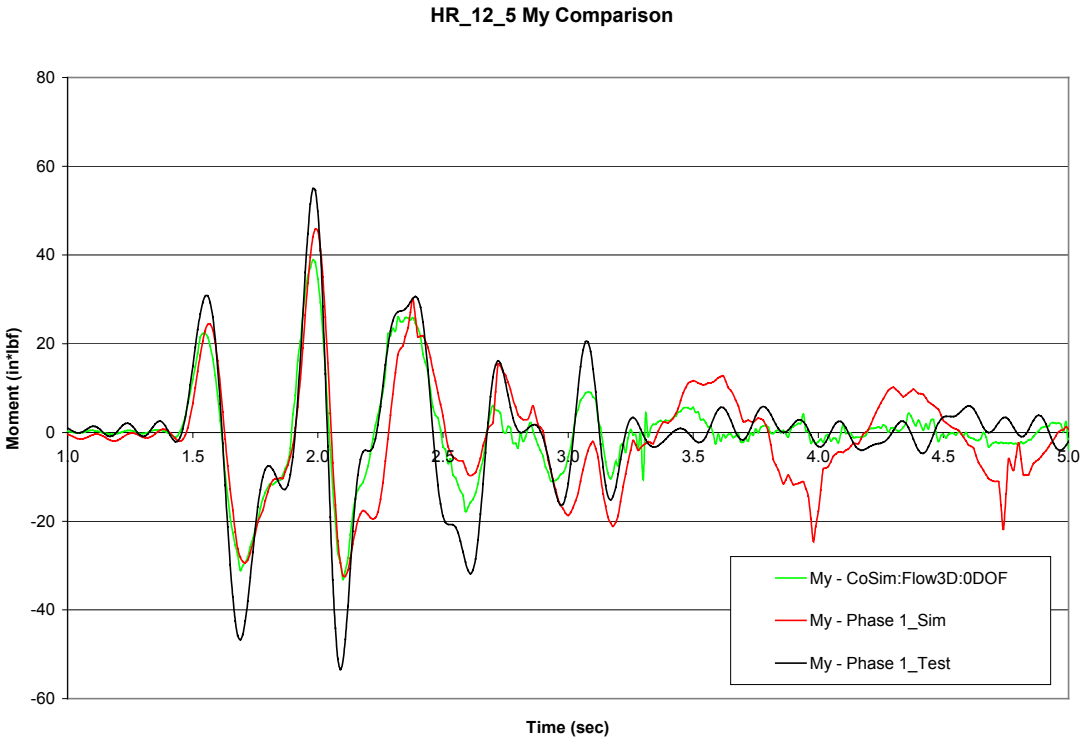

HR_9_10 Mz Comparison
Moment (in*lbf)
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase 1_Sim
Mz - Phase 1_Test

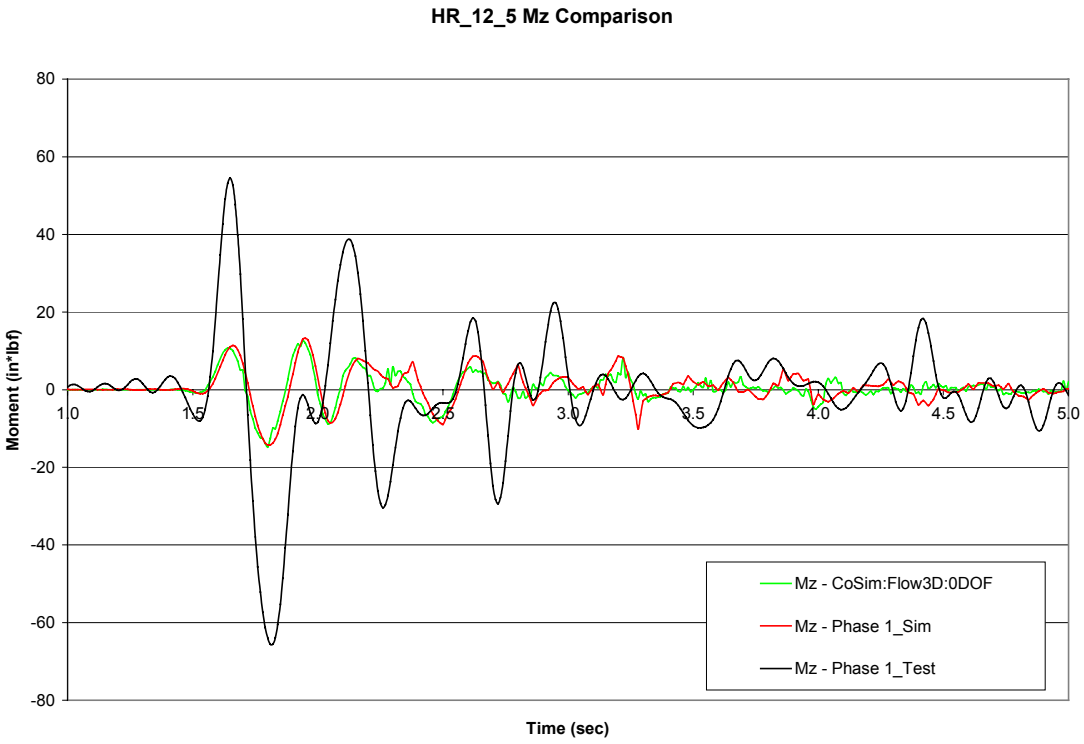

HR_12_5 Fx Comparison
Force (lbf)
Time (sec)
Fx - CoSim:Flow3D:0DOF
Fx - Phase 1_Sim
Fx - Phase 1_Test

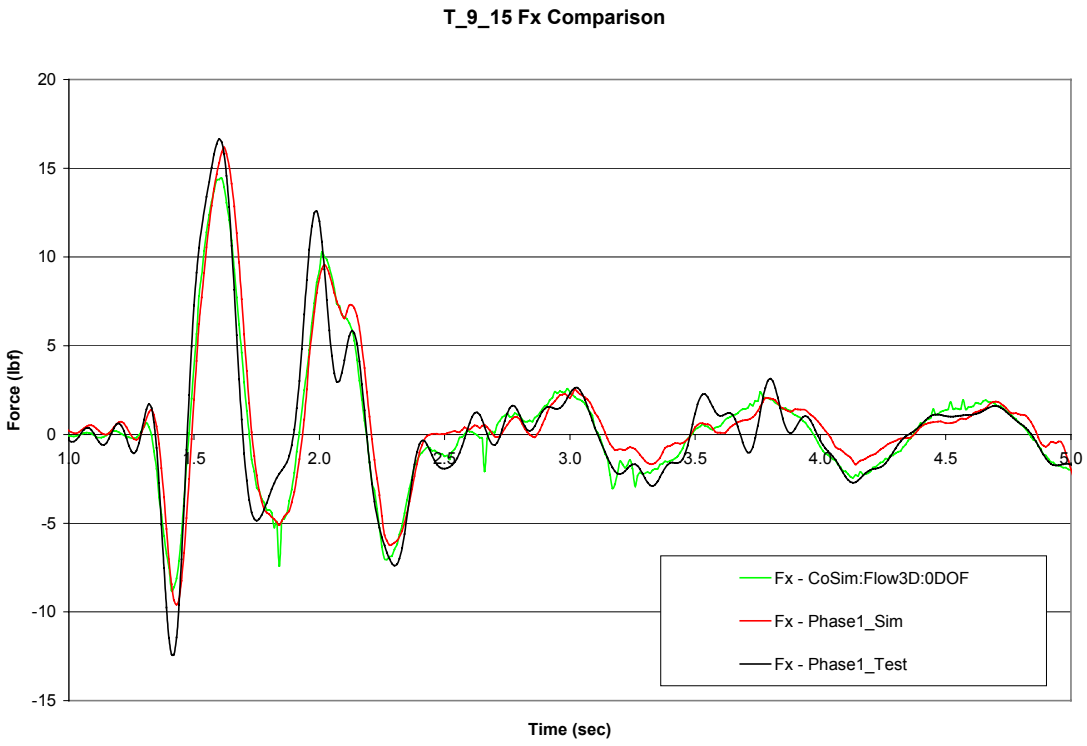

HR_12_5 Fy Comparison
Force (lbf)
Time (sec)
Fy - CoSim:Flow3D:0DOF
Fy - Phase 1_Sim
Fy - Phase 1_Test

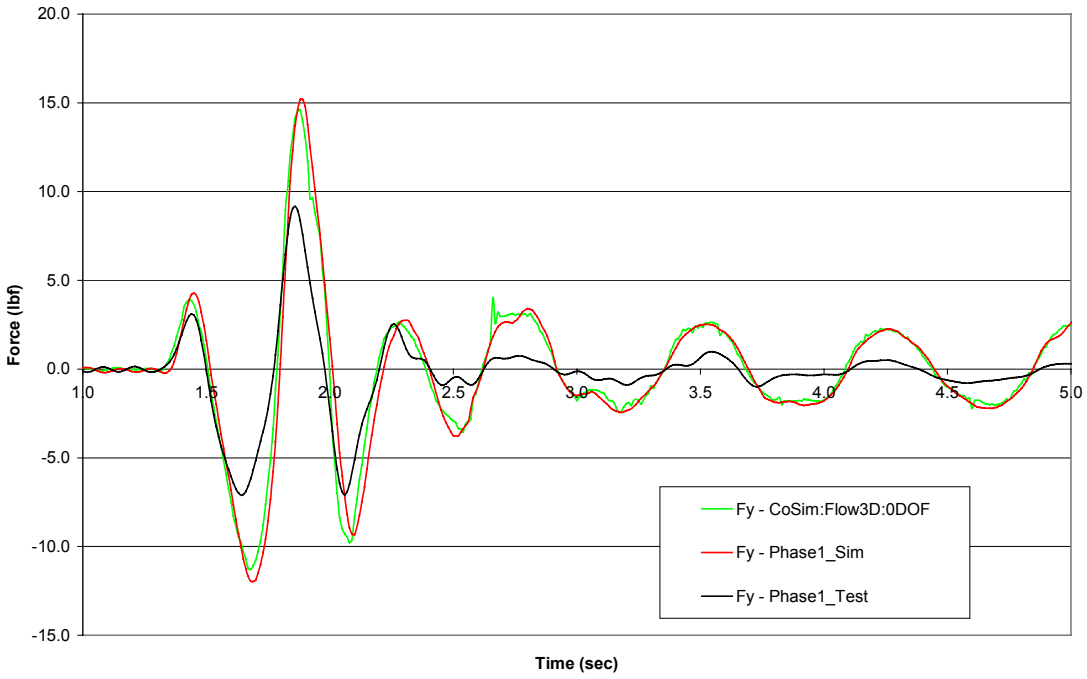

HR_12_5 Fz Comparison
120.0
100.0
80.0
60.0
40.0
20.0
0.0
-20.0
-40.0
-60.0
-80.0
Force (lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Fz - CoSim:Flow3D:0DOF
Fz - Phase 1_Sim
Fz - Phase 1_Test

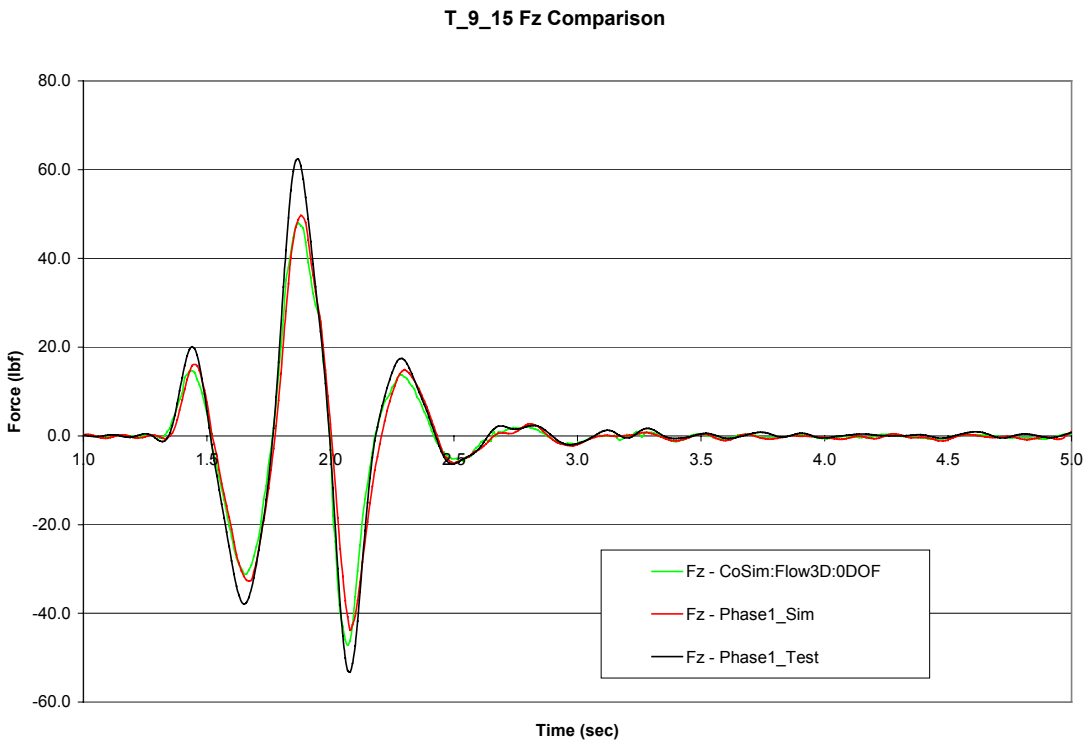

HR_12_5 Mx Comparison
100
80
60
40
20
0
-20
-40
-60
-80
-100
-120
Moment (in*lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Mx - CoSim:Flow3D:0DOF
Mx - Phase 1_Sim
Mx - Phase 1_Test

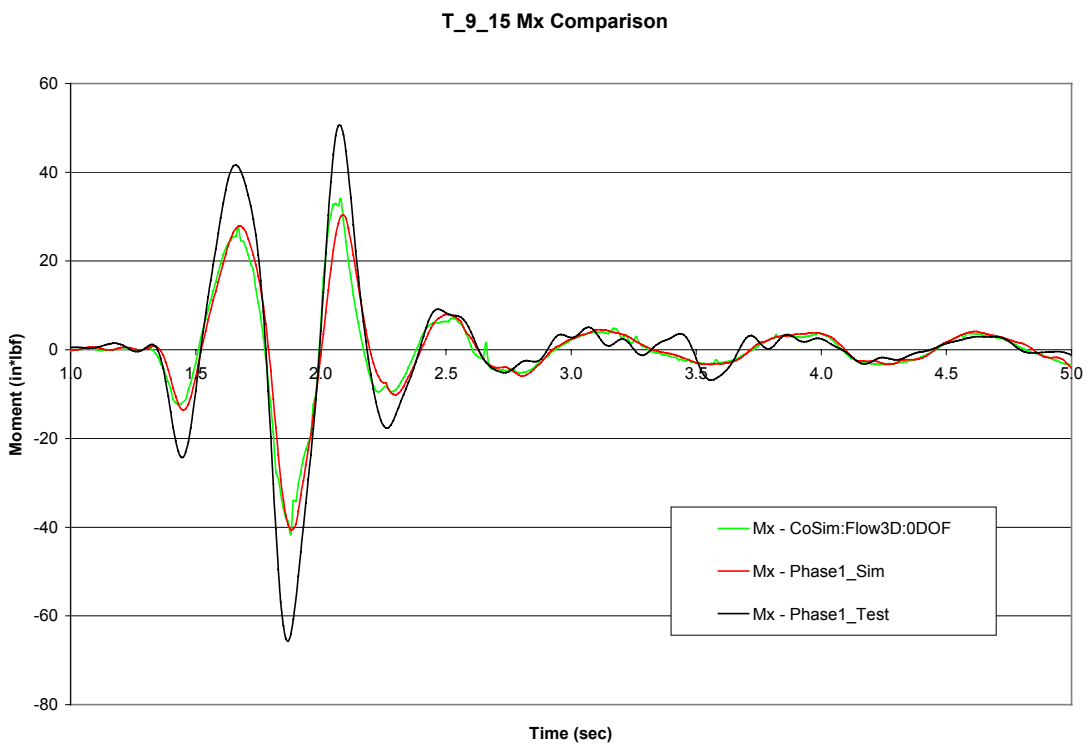

HR_12_5 My Comparison
Moment (in*lbf)
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase 1_Sim
My - Phase 1_Test

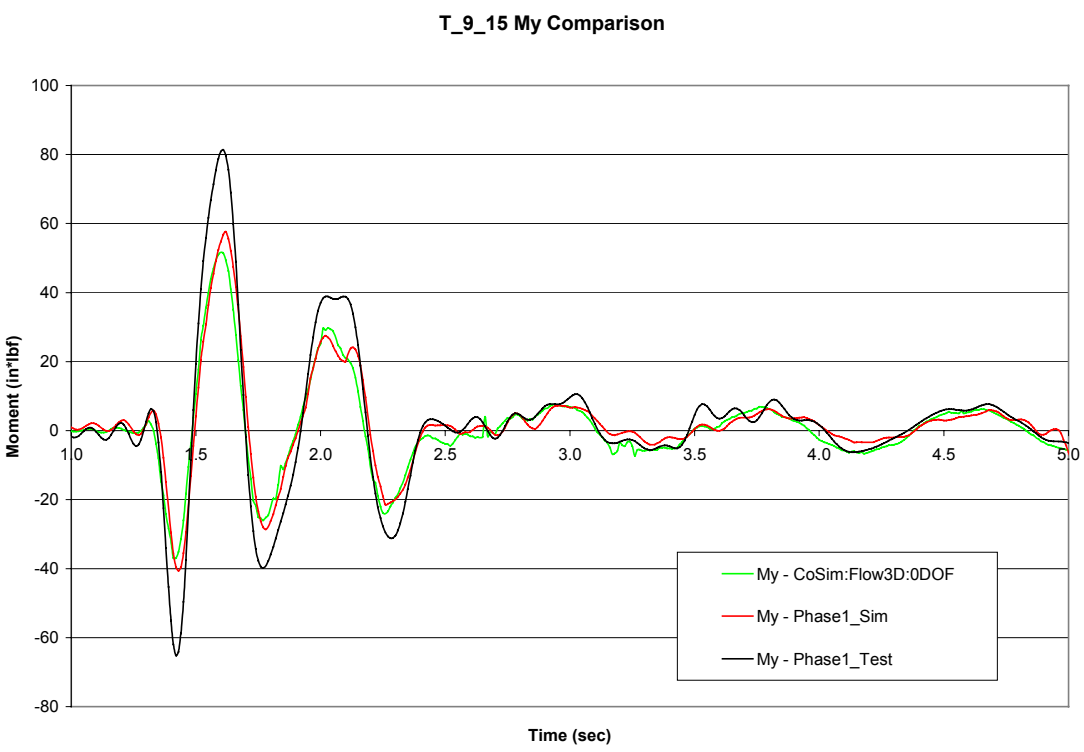

HR_12_5 Mz Comparison
Moment (in*lbf)
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase 1_Sim
Mz - Phase 1_Test

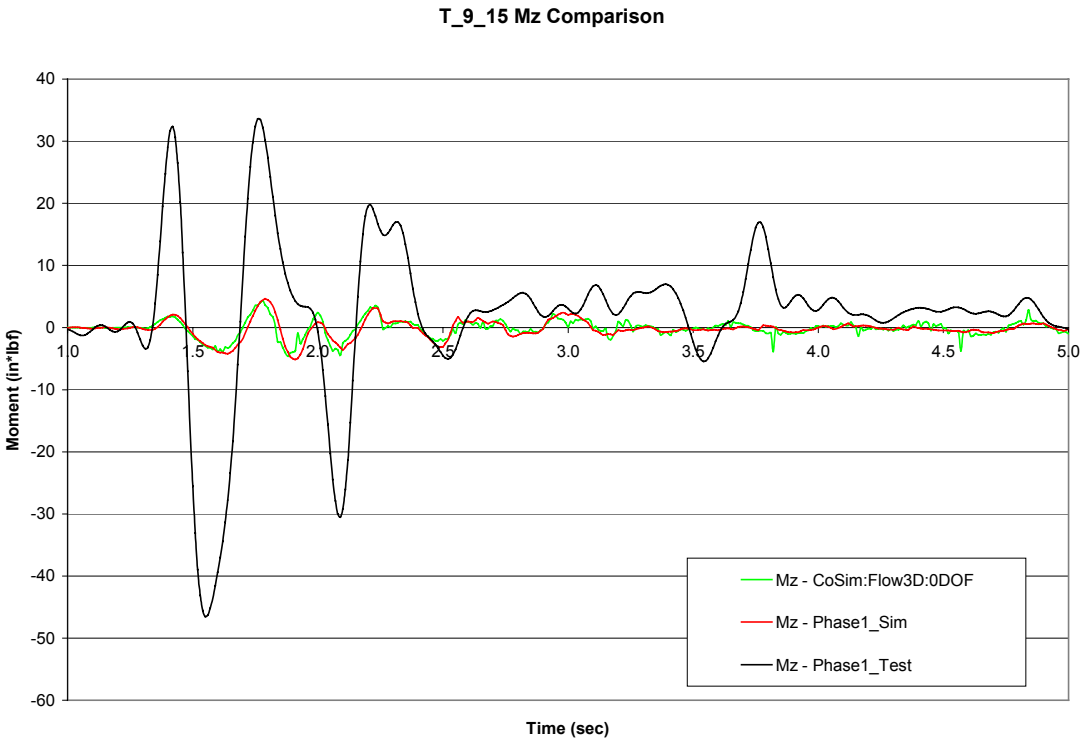

T_9_15 Fx Comparison
Force (lbf)
Time (sec)
Fx - CoSim:Flow3D:0DOF
Fx - Phase1_Sim
Fx - Phase1_Test

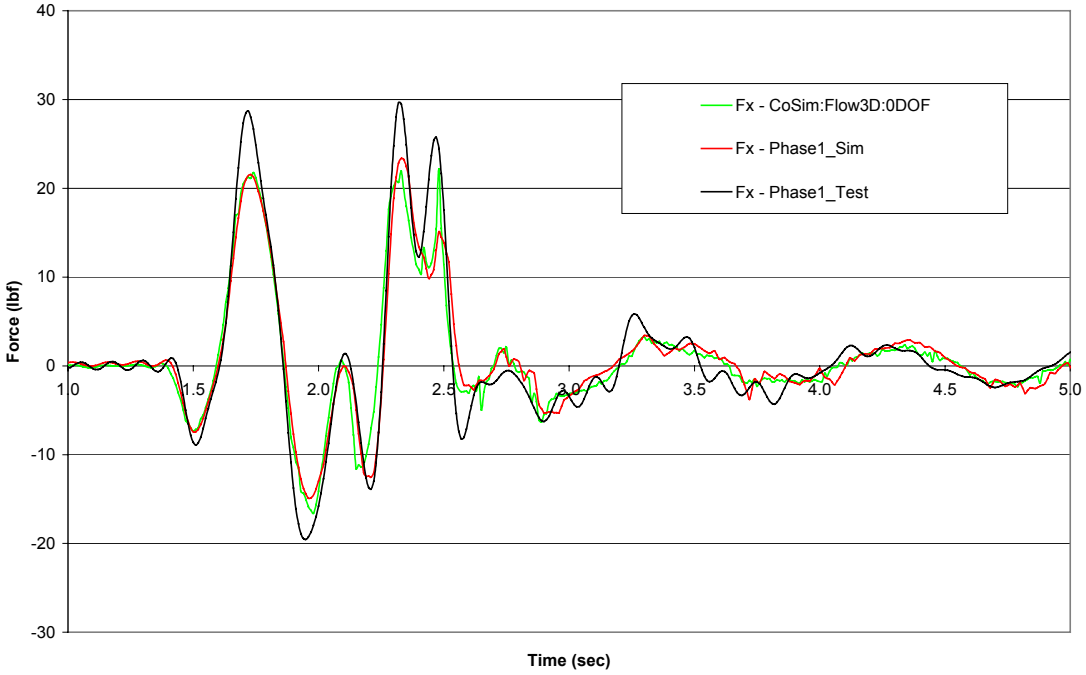

T_9_15 Fy Comparison
Force (lbf)
Time (sec)
Fy - CoSim:Flow3D:0DOF
Fy - Phase1_Sim
Fy - Phase1_Test

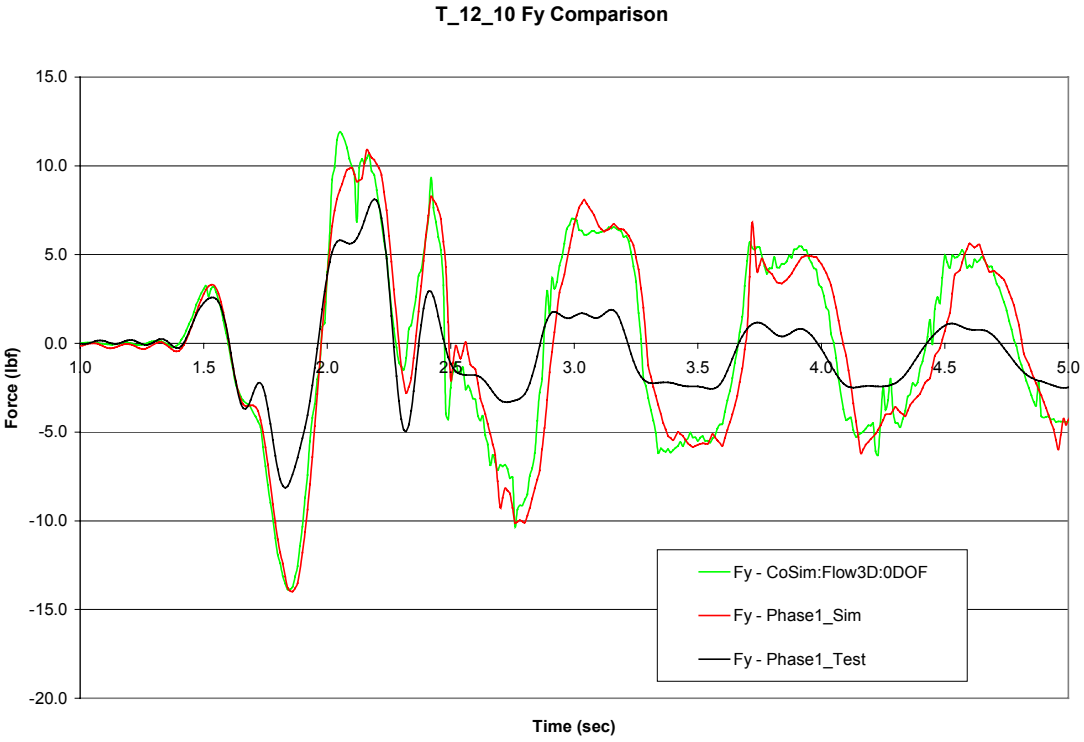

T_9_15 Fz Comparison
80.0
60.0
40.0
20.0
0.0
-20.0
-40.0
-60.0
Force (lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Fz - CoSim:Flow3D:0DOF
Fz - Phase1_Sim
Fz - Phase1_Test

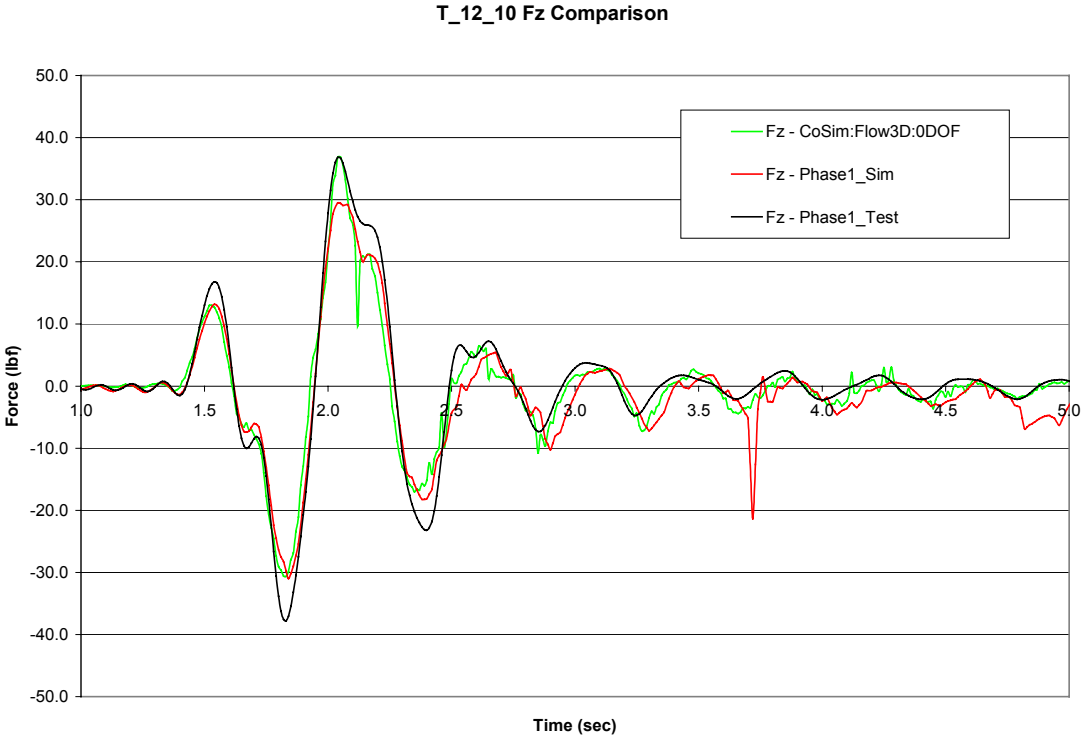

T_9_15 Mx Comparison
60
40
20
0
-20
-40
-60
-80
Moment (in*lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Mx - CoSim:Flow3D:0DOF
Mx - Phase1_Sim
Mx - Phase1_Test

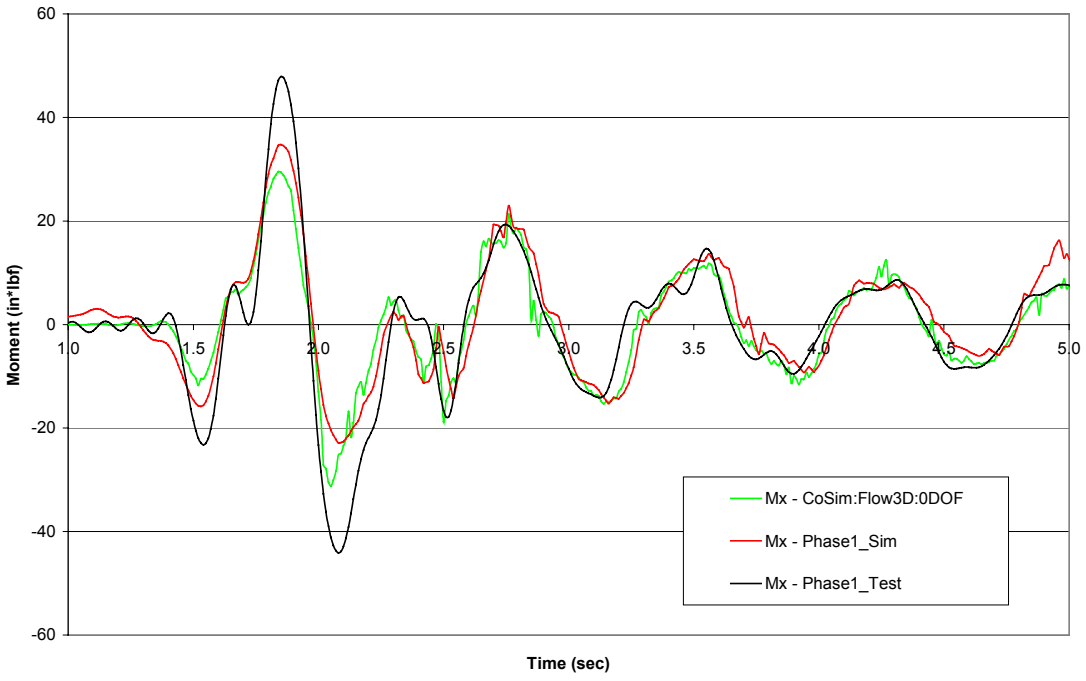

T_9_15 My Comparison
Moment (in*lbf)
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase1_Sim
My - Phase1_Test

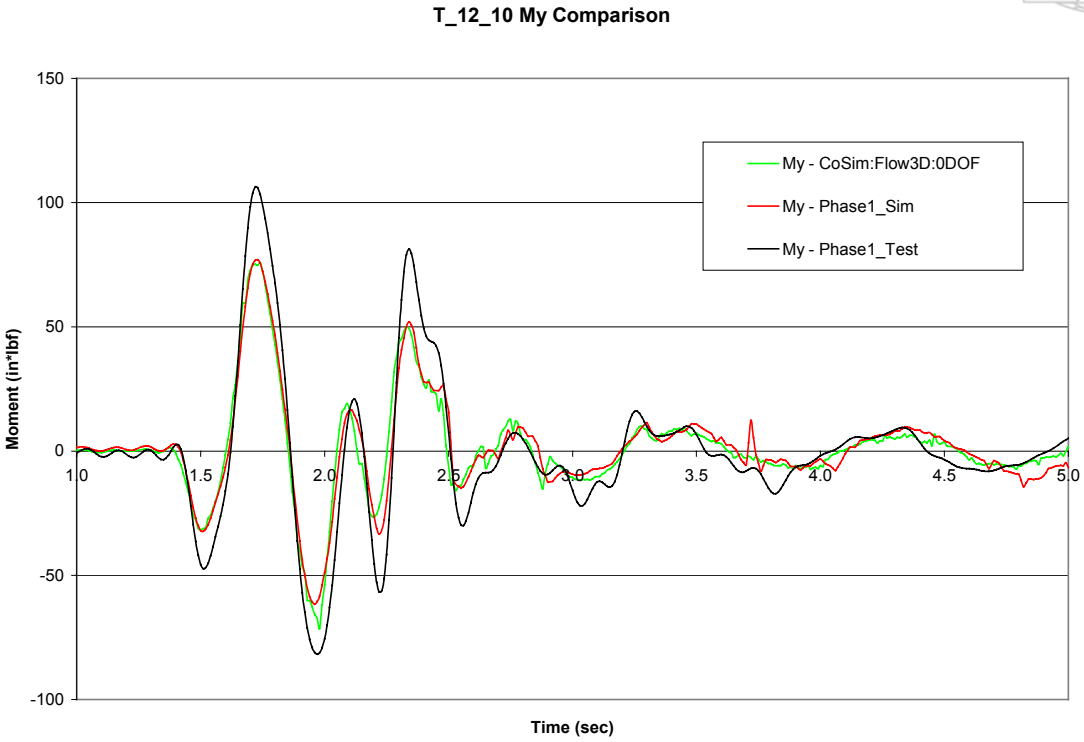

T_9_15 Mz Comparison
Moment (in*lbf)
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase1_Sim
Mz - Phase1_Test

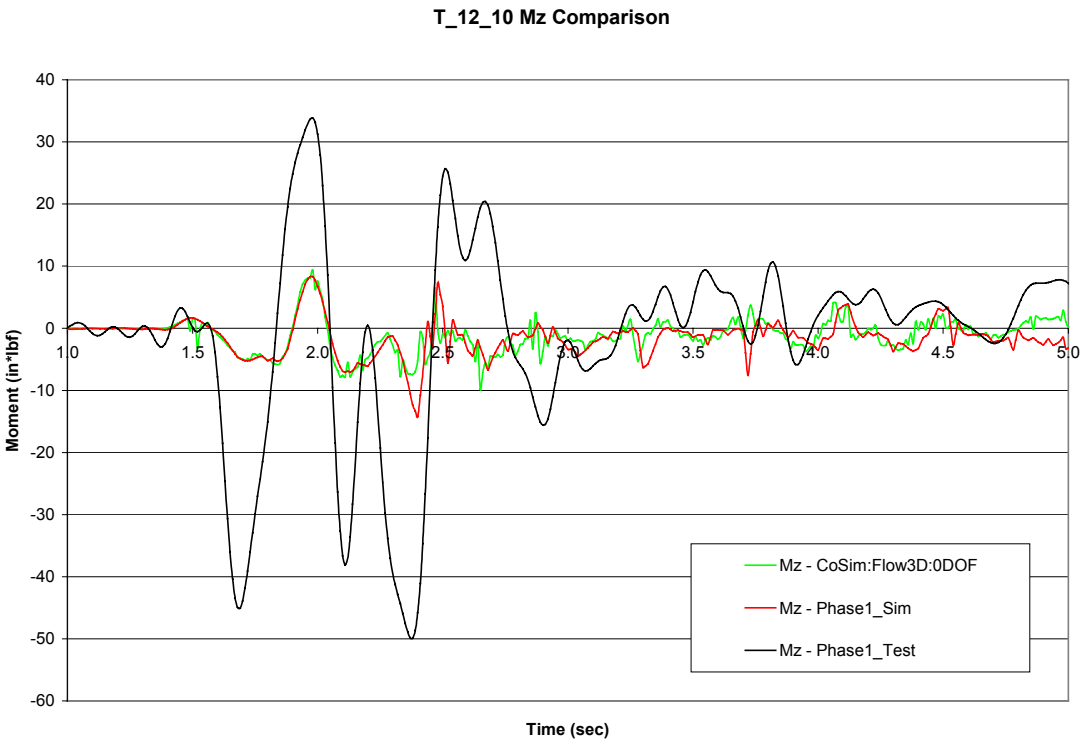

T_12_10 Fx Comparison
40
30
20
10
0
-10
-20
-30
Force (lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Fx - CoSim:Flow3D:0DOF
Fx - Phase1_Sim
Fx - Phase1_Test


T_12_10 Fy Comparison
15.0
10.0
5.0
0.0
-5.0
-10.0
-15.0
-20.0
Force (lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Fy - CoSim:Flow3D:0DOF
Fy - Phase1_Sim
Fy - Phase1_Test


T_12_10 Fz Comparison
Fz - CoSim:Flow3D:0DOF
Fz - Phase1_Sim
Fz - Phase1_Test
Force (lbf)
Time (sec)
50.0
40.0
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
-40.0
-50.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


T_12_10 Mx Comparison
Mx - CoSim:Flow3D:0DOF
Mx - Phase1_Sim
Mx - Phase1_Test
Moment (in*lbf)
Time (sec)
60
40
20
0
-20
-40
-60
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


T_12_10 My Comparison
150
100
50
0
-50
-100
Moment (in*lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
My - CoSim:Flow3D:0DOF
My - Phase1_Sim
My - Phase1_Test


T_12_10 Mz Comparison
40
30
20
10
0
-10
-20
-30
-40
-50
-60
Moment (in*lbf)
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time (sec)
Mz - CoSim:Flow3D:0DOF
Mz - Phase1_Sim
Mz - Phase1_Test