Copy No. _____

Defence Research and Development Canada   Recherche et développement pour la défense Canada

DEFENCE   DÉFENSE

CFD Simulations of a Ferry in Head Seas

Eric Thornhill

Defence R&D Canada – Atlantic

Technical Memorandum

DRDC Atlantic TM 2011-030

March 2011

Canada

CFD Simulations of a Ferry in Head Seas

Eric Thornhill

Defence Research and Development Canada – Atlantic
Technical Memorandum
DRDC Atlantic TM 2011-030
March 2011

Abstract

A ship travelling in a seaway can often experience slamming characterized by large sudden loads from impacts with waves. These loads can be large enough to cause local damage, but also induce a whipping response which stresses the primary hull structure and adversely affects the overall fatigue life of the vessel. Most common seakeeping prediction programs use potential flow theory that cannot directly simulate the physics of slamming. This report contains a study using more advanced computational fluid dynamics methods which can reproduce wave impacts. Simulations were conducted for a ferry hull travelling in head seas and compared to existing experimental data. It was found that slamming loads are well predicted, but there can be difficulties in generating and propagating large steep waves inside the simulation.

Résumé

Il arrive souvent qu'un navire voyageant dans une voie maritime fasse l'objet de tossage caractérisé par d'importantes charges soudaines engendrées par l'impact des vagues. Ces charges peuvent être suffisamment importantes non seulement pour provoquer des avaries locales, mais aussi pour entraîner une vibration de frottement qui exerce une contrainte sur la structure de coque primaire et qui a un effet défavorable sur la durée de vie en fatigue globale du navire. La plupart des programmes de prédiction de la tenue en mer les plus couramment utilisés font appel à la théorie de l'écoulement potentiel, laquelle ne peut simuler directement la physique du tossage. Le présent rapport contient les résultats d'une étude faisant appel à des méthodes plus avancées de dynamique computationnelle des fluides qui sont à même de reproduire les impacts des vagues. Des simulations ont été effectuées sur une coque de traversier voyageant en mer debout et ont été comparées aux données expérimentales existantes. Il s'est révélé que les charges attribuables au tossage peuvent être bien prédites, mais qu'il peut être quelque peu difficile de générer et de propager de grandes vagues creuses en contexte de simulation.

This page intentionally left blank.

Executive summary

CFD Simulations of a Ferry in Head Seas

Eric Thornhill; DRDC Atlantic TM 2011-030; Defence Research and Development Canada – Atlantic; March 2011.

Background: A ship travelling in a seaway can often experience slamming characterized by large sudden loads from impacts with waves. These loads can be large enough to cause local damage, but also induce a whipping response which stresses the primary hull structure and adversely affects the overall fatigue life of the vessel. It is important for ship design, operations, as well as life cycle management to quantify and understand the effects of slamming on the ship structure. Most common seakeeping prediction programs use potential flow theory that cannot directly simulate the physics of slamming. This report contains a study using more advanced computational fluid dynamics methods which can reproduce wave impacts. Simulations were conducted for a ferry hull travelling in head seas and compared to existing experimental data.

Principal results: The simulation results were compared with existing experimental data from a ferry hull model that was held captive to a tow carriage as it sailed into oncoming waves. Wave probes attached to the carriage measured the wave profile while forces and pressures were measured on a segment of the bow that was isolated from the main hull. It was found that slamming forces and pressures on this bow segment were well predicted, but there can be difficulties in generating and propagating large steep waves inside the simulation.

Significance of results: The results show that computational fluid dynamics can be used to accurately predict slam impact forces. This simulation technique could therefore be used as a tool to evaluate a hull form with respect to slamming, or as a means of generating data for the development and validation of models used in other types of simulation methods (such as traditional seakeeping codes). However, difficulties in generating large steep waves in this type of simulation may limit the conditions that can be effectively examined using this approach.

Future work: Future work will include simulations on other cases from the model experiments such as forced pitch oscillation tests, and simulations in oblique seas. There will also be work on investigating wave generation/propagation using computational fluid dynamics to improve the simulations or to define limits of applicability.

Sommaire

CFD Simulations of a Ferry in Head Seas

Eric Thornhill ; DRDC Atlantic TM 2011-030 ; Recherche et développement pour la défense Canada – Atlantique ; mars 2011.

Contexte : Il arrive souvent qu'un navire voyageant dans une voie maritime fasse l'objet de tossage caractérisé par d'importantes charges soudaines engendrées par l'impact des vagues. Ces charges peuvent être suffisamment importantes non seulement pour provoquer des avaries locales, mais aussi pour entraîner une vibration de frottement qui exerce une contrainte sur la structure de coque primaire et qui a un effet défavorable sur la durée de vie en fatigue globale du navire. Il est important, pour la conception, l'exploitation et la gestion du cycle de vie d'un navire, de pouvoir quantifier et comprendre les effets du tossage sur sa structure. La plupart des programmes de prédiction de la tenue en mer les plus couramment utilisés font appel à la théorie de l'écoulement potentiel, laquelle ne peut simuler directement la physique du tossage. Le présent rapport contient les résultats d'une étude faisant appel à des méthodes plus avancées de dynamique computationnelle des fluides qui sont à même de reproduire les impacts des vagues. Des simulations ont été effectuées sur une coque de traversier voyageant en mer debout et ont été comparées aux données expérimentales existantes.

Résultats principaux : Les résultats des simulations ont été comparés aux données expérimentales existantes provenant d'un modèle de coque de traversier captif d'un chariot de remorquage qui naviguait dans les vagues venant en sens inverse. Des enregistreurs de vagues ont été fixés au chariot pour mesurer le profil de la houle alors que les forces et les pressions exercées ont été mesurées sur un segment d'étrave isolé de la coque principale. Il s'est révélé que les forces et les pressions engendrées par le tossage sur ce segment d'étrave ont été bien prédites, mais qu'il peut être quelque peu difficile de générer et de propager de grandes vagues creuses en contexte de simulation.

Portée des résultats : Les résultats démontrent que la dynamique computationnelle des fluides peut servir à prédire avec précision les forces d'impact du tossage. Cette technique de simulation pourrait donc servir d'outil pour évaluer la forme d'une coque relativement au tossage, ou encore de moyen de générer des données pour l'élaboration et la validation de modàles utilisés avec d'autres types de méthodes de simulation (p. ex. les codes de tenue en mer classiques). Il est toutefois quelque peu difficile de générer de grandes vagues creuses dans ce type de simulations, ce qui pourrait limiter les conditions pouvant être efficacement étudiées à l'aide de cette approche.

Recherches futures : Les recherches futures comprendront des simulations portant sur d'autres cas provenant des expériences réalisées sur maquettes, comme les essais d'oscillation en tangage forcé, et des simulations dans des mers obliques. Des travaux de recherche seront également menés sur la production et la propagation de vagues par dynamique computationnelle des fluides pour améliorer les simulations faisant appel à cette méthode ou pour en définir des limites d'applicabilité.

This page intentionally left blank.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire iv

Table of contents vii

List of figures ix

List of tables xi

1 Introduction 1

2 Model Tests 2

3 Computational Fluid Dynamics (CFD) Simulations 5

3.1 Geometry 5

3.2 Mesh 9

3.3 Set-up 11

3.4 Analysis 12

4 Results 13

5 Summary and Conclusions 17

References 18

Annex A: Wave Probe Analysis 19

Annex B: CFX Mesh Settings 25

Annex C: CFX Solver Settings 29

Annex D: CFX User Fortran: initval 43

Annex E: CFX User Fortran: getirregwave 47

Annex F: Pressure Tap Locations 53

Annex G: Virtual Wave Probes 57

Annex H: Pressure Transducer Results 59

Annex I: ELAST Ferry CFD Results 81

Acronyms 87

List of figures

Figure 1: ELAST body plan 3

Figure 2: ELAST ferry model showing bow segment and pressure transducers . . 4

Figure 3: ELAST ferry model during tests 4

Figure 4: Original hull surfaces 6

Figure 5: Hull surfaces without skeg used to form solid 6

Figure 6: Hull solid with skeg solid 7

Figure 7: Location of bow segment 7

Figure 8: Simulation domain 8

Figure 9: Mesh on symmetry and inlet planes 10

Figure 10: Mesh around bow segment 10

Figure 11: Early run with poor grid resolution 13

Figure 12: Later run with proper grid resolution 13

Figure 13: X-Force on bow segment 14

Figure 14: X-force on bow segment 15

Figure 15: Wave elevations at probe #16 for run 211002 16

Figure 16: Pressure transducer #1 data 16

Figure A.1: Wave probe wave elevation data 21

Figure A.2: Wave reconstructions at probe 16 location 22

Figure A.3: Wave probe array orientations 23

Figure H.1: Pressure transducer #1 data 59

Figure H.2: Pressure transducer #2 data 60

Figure H.3: Pressure transducer #3 data 61

Figure H.4: Pressure transducer #4 data 62
Figure H.5: Pressure transducer #5 data 63
Figure H.6: Pressure transducer #6 data 64
Figure H.7: Pressure transducer #7 data 65
Figure H.8: Pressure transducer #8 data 66
Figure H.9: Pressure transducer #9 data 67
Figure H.10: Pressure transducer #10 data 68
Figure H.11: Pressure transducer #11 data 69
Figure H.12: Pressure transducer #12 data 70
Figure H.13: Pressure transducer #13 data 71
Figure H.14: Pressure transducer #14 data 72
Figure H.15: Pressure transducer #15 data 73
Figure H.16: Pressure transducer #16 data 74
Figure H.17: Pressure transducer #17 data 75
Figure H.18: Pressure transducer #18 data 76
Figure H.19: Pressure transducer #19 data 77
Figure H.20: Pressure transducer #20 data 78
Figure H.21: Pressure transducer #21 data 79
Figure H.22: Pressure transducer #22 data 80
Figure I.1: X-force on bow segment 81
Figure I.2: Z-force on bow segment 82
Figure I.3: Wave probe #1 data 83
Figure I.4: Wave probe #2 data 84
Figure I.5: Wave probe #10 data 85

Figure I.6: Wave probe #16 data 86

List of tables

Table 1: Main dimensions of ELAST ferry 3

Table 2: Mesh information 9

Table A.1: Main dimensions of ELAST ferry 22

Table F.1: Pressure gauge locations 53

This page intentionally left blank.

1 Introduction

A ship travelling in a seaway can often experience slamming characterized by large sudden loads from impacts with waves. These loads can be large enough to cause local damage, but also induce a whipping response which stresses the primary hull structure and adversely affects the overall fatigue life of the vessel. Most common seakeeping prediction programs use potential flow theory that cannot directly simulate the physics of slamming. If they account for slamming at all, it is done through additional approaches such as custom models that rely on empirical corrections. Accurate evaluation of slamming impact loads for a specific hull form therefore relies on model experiments which are not always available.

This report contains a study using Reynolds-averaged Navier-Stokes (RANS) CFD methods which can simulate slamming [1]. This approach, although much more computationally expensive than potential flow methods, does capture the aspects of fluid behaviour that most influence impact loads. Successful validation of this approach would mean that it could be used as a tool to evaluate slam impact loads for specific conditions, or even to generate data to validate and develop models for more traditional potential flow based seakeeping prediction codes.

The simulations were set up to match conditions from model tests performed by the Co-operative Research Ships (CRS) WHIP working group. The physical experiments were conducted at MARIN on a 1:36 scale model of a Ropax ferry that was previously used by the DAMA and ELAST CRS Working Groups (hull form will be referred to as the 'ELAST ferry'). The model was rigidly attached to the tow carriage with an isolated bow segment where wave forces could be measured independently of the hull. Several wave probes were used to measure the wave's profile and velocity near the bow segment. Tests were conducted at several speeds and headings into both regular and irregular waves. Most runs were performed with the model held captive to the carriage except for a series where the model was forced in a pitching motion by a mechanical oscillator. Also during the CRS WHIP project, validation studies were carried out with its own custom software using model test data. To facilitate this, cases from the large model test data set were selected as having good representative examples of slamming. The current study focuses on run 207004 from the model tests.

Test 207004 was a captive model test running at 20 knots (full scale speed) into head seas (180° heading to the incident wave direction). The incoming waves were generated as a 'wave sweep' which starts with longer wave length waves that gradually get shorter while maintaining the same amplitude. The waves therefore get progressively steeper during the run, usually ensuring many bow flare slam events.

2 Model Tests

A special series of model experiments were carried out in 2007 by the CRS WHIP Working Group for validation purposes. In order to control the impact conditions as much as possible, the vessel was held captive for one set of tests, and oscillated with prescribed pitch in an additional set.

The 1:36 scale model is of a ferry which has been used in earlier experiments on bow impacts [2]. The main dimensions are given in Table 1 and the hull lines are shown in Figure 1. Originally this model had a bulbous bow, but in order to simplify the impact conditions on the bow of the ship, the bulbous bow had been removed.

The interest was on the impulsive forces on the bow, so the model was constructed in two parts: the main hull body, and an isolated bow segment. The bow segment (shown in Figure 2) extended forward of Station 19.5 and upwards from the waterline and was separated from the model by a six-component strain gauge system. The connection was made stiff enough that the model would behave as a rigid body. The segment was also fitted with 23 pressure taps to record local hull pressures. The pressure tap locations are given Annex H.

Three series of experiments were performed. Series 1 of the experiments was carried out with a wave train consisting of waves with constant wave amplitude but decreasing length. This results in a series of different impacts starting mildly and increasing as the waves get steeper. Series 2 consisted of one single large wave (referred to as a focused wave) and hence one single severe impact. Series 3 consisted of forced pitch oscillation tests in regular waves. Experiments were performed for several forward speeds and included tests in oblique waves up to 120° (head seas was defined as 180°).

The incoming wave profiles were measured by a wave probe array attached to the carriage. The 16 capacitance wave probe array was always oriented parallel to the incoming wave direction ahead of the bow (just far ahead to be unaffected by the bow wave or splash). The data from these probes was combined with linear wave theory to create the incoming bow wave used by the CFD simulations. A description of this procedure is given in Annex A.

The current study focuses on run 207004 which is from Series 1 of the experiments. The model was held fixed to the carriage (no roll, pitch, or heave) at 20 knots (full scale speed) into head seas.

Table 1: *Main dimensions of ELAST ferry*

	Full Scale	Model Scale
Length (LBP)	173 m	4.806 m
Beam	26 m	0.722 m
Draft	6.3 m	0.175 m
Depth	17.5 m	0.486 m
Displacement	15,500 t	332 kg

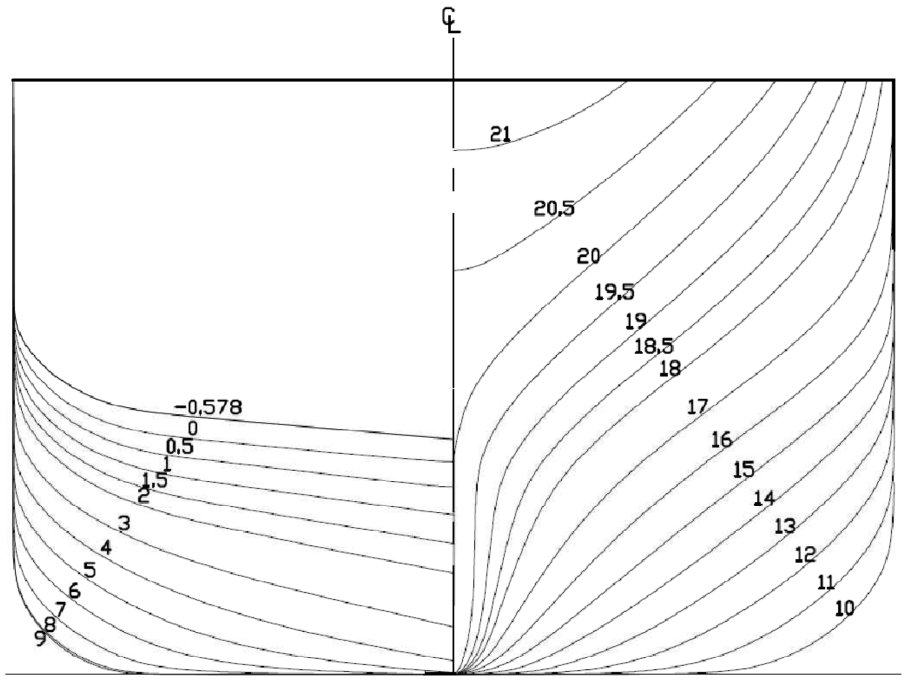

Figure 1: *ELAST body plan*

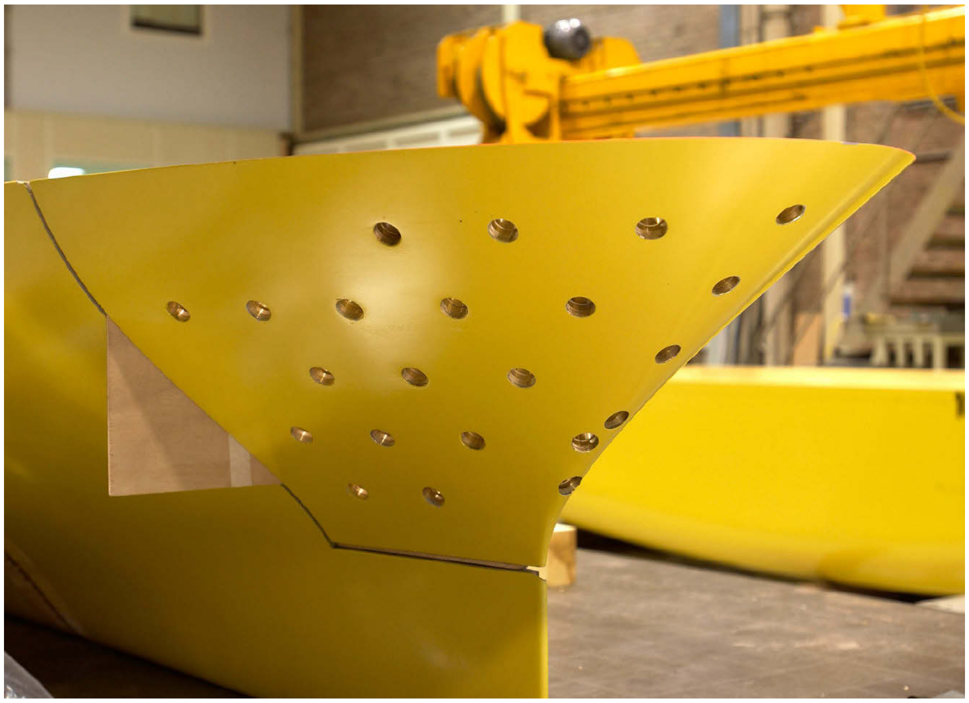

Figure 2: *ELAST ferry model showing bow segment and pressure transducers*

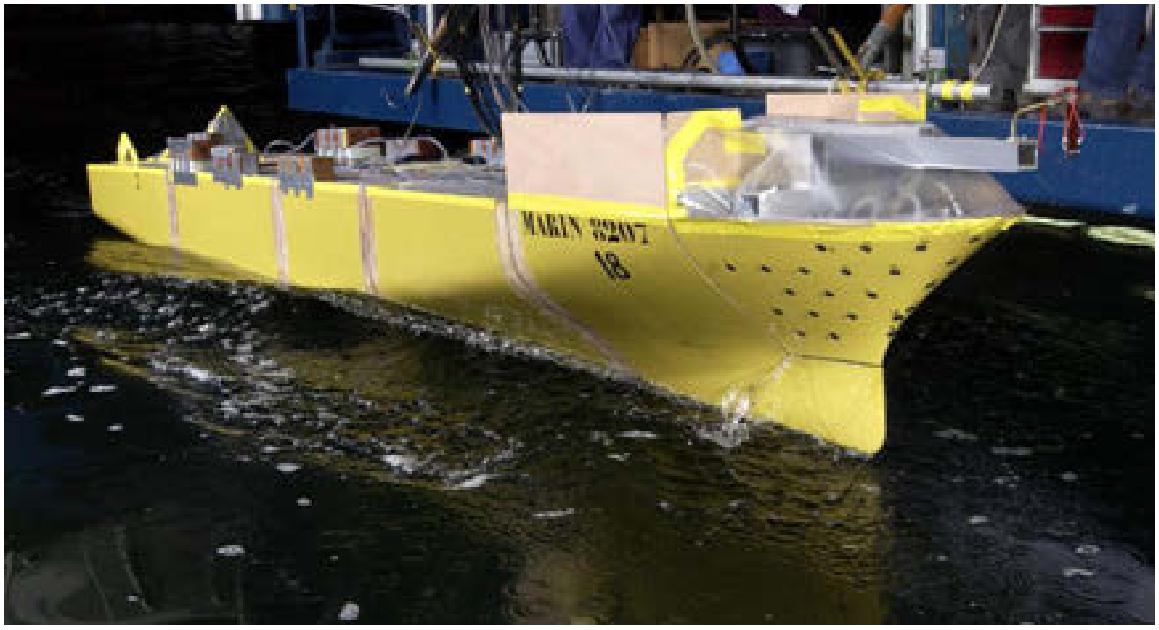

Figure 3: *ELAST ferry model during tests*

3 CFD Simulations

CFD simulations were performed with ANSYS® CFX® 11.0 on an unstructured mesh using the volume-of-fluid method for the free surface interface. The simulations were conducted at the same model scale (1:36) as the experiments for run 207004.

3.1 Geometry

The geometry was supplied to DRDC by MARIN in Initial Graphics Exchange Specification (IGES) format which defined the hull as a single surface with the skeg as an independent surface (see Figure 4). In order to use the CFX meshing tools, the geometry had to be converted into a solid format (wherein all surfaces connect at their edges and are 'airtight'). There was some difficulty merging the skeg surface cleanly into the hull surface as they were overlapping (as opposed to meeting at edges).

Ultimately, the final geometry was created in parts. First the hull surface without skeg was mirrored across its symmetry plane and given a top deck surface and transom surface (both were planar surfaces defined by the edges of the hull surface). These were then formed into a solid as shown in Figure 5. Next the skeg surface was used as a basis for defining a new set of surfaces which formed a separate skeg solid that extended into the volume of the hull solid (Figure 6). These two solids were then united with a Boolean operation to form a single solid hull with skeg.

The next step was to create an isolated bow segment to match the physical model. The bow segment was a section forward of station 19.5 and above the waterline as shown in Figure 7. This was created by slicing the hull solid at the appropriate lines. The hull geometry then consisted of two solids; the main hull and bow segment. By separating the bow segment as with its own surfaces, it becomes easy in the solver and post-processor to extract forces, moments, and pressures. In retrospect, more sections should have been made near the bow.

The last step was to create a domain solid (a simple rectangular block) and subtract the hull solids as shown in Figure 8. This defines the air/water space where flow calculations take place. As only head sea cases were simulated, a symmetry plane was used along the ship centreline to reduce the mesh size. The domain was 2.78 m (100.8 m FS) wide from the side wall to the symmetry plane, 20 m (720 m FS) long, and 3.5 m (126 m FS) tall. The calm water water depth was 2.425 m (87.5 m FS). The hull was located so that its bow was approximately one ship length from the inlet. The distance to the inlet was an important parameter for these simulations. The ship had to be far enough away so that the generated waves at the inlet could form properly, but close enough to minimize any degradation of the wave profile.

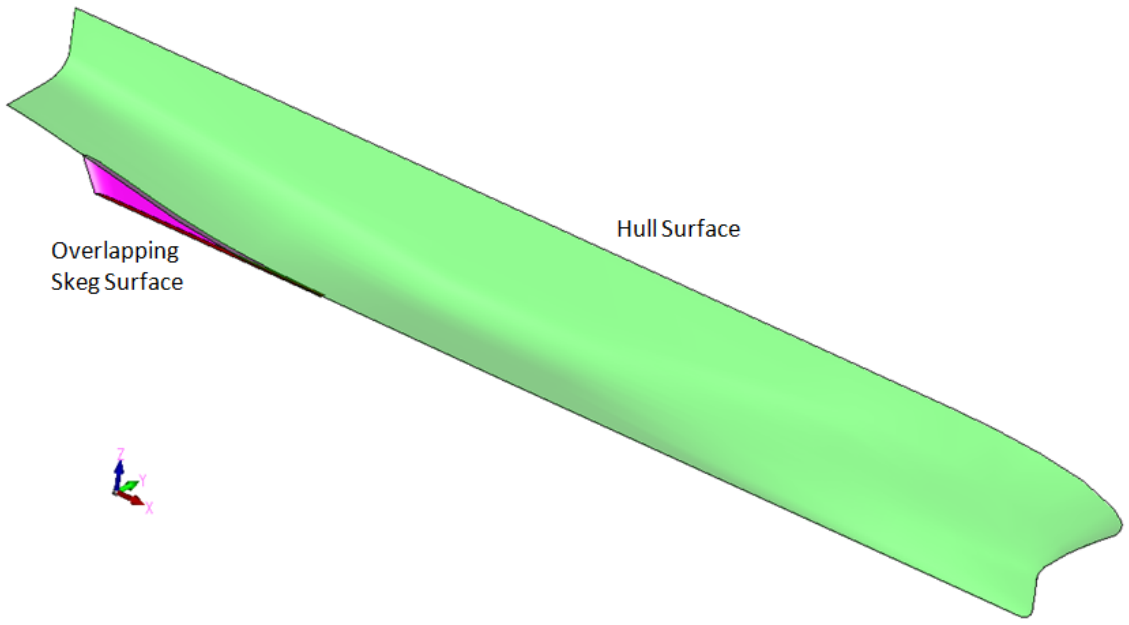

Figure 4: *Original hull surfaces*

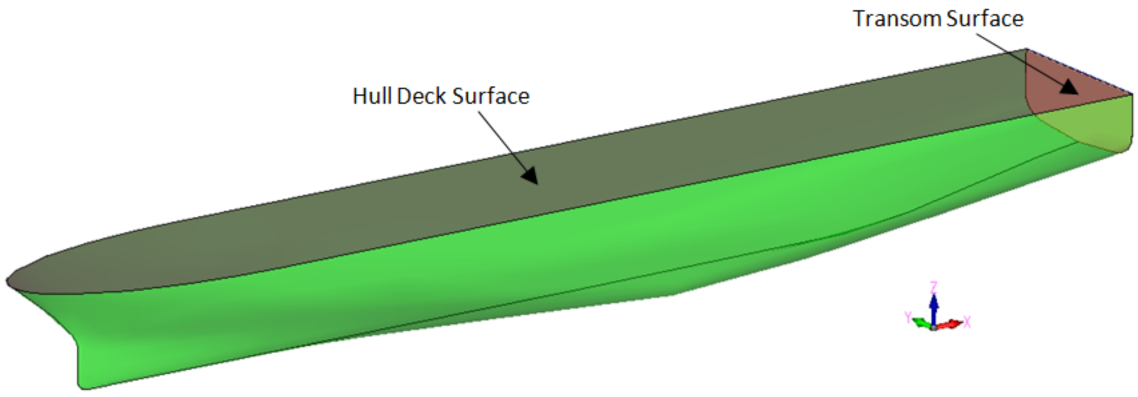

Figure 5: *Hull surfaces without skeg used to form solid*

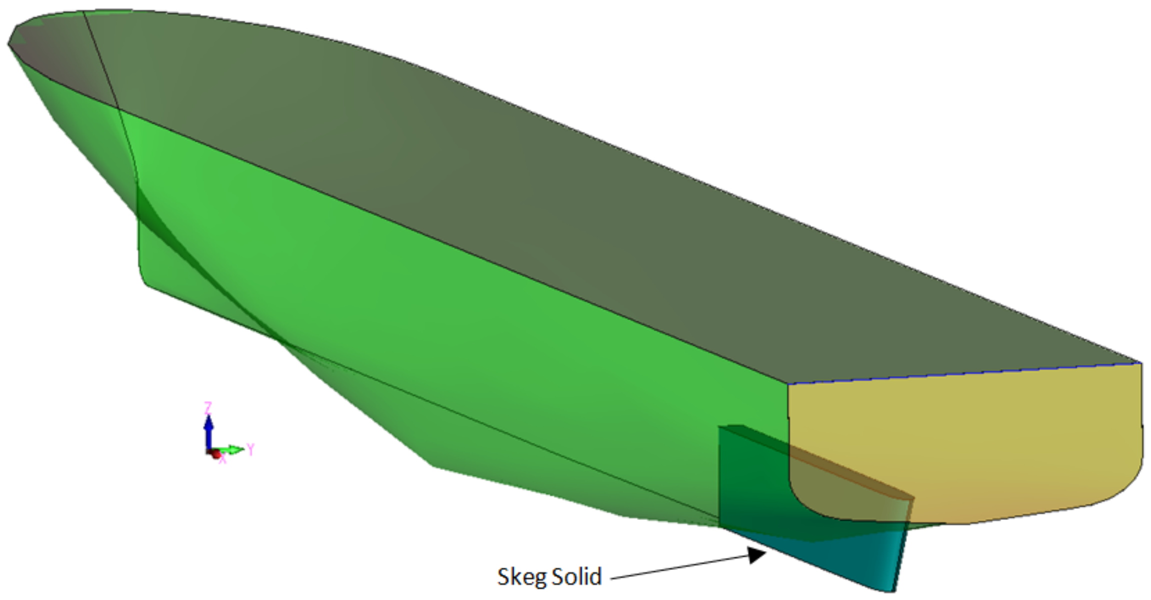

Figure 6: *Hull solid with skeg solid*

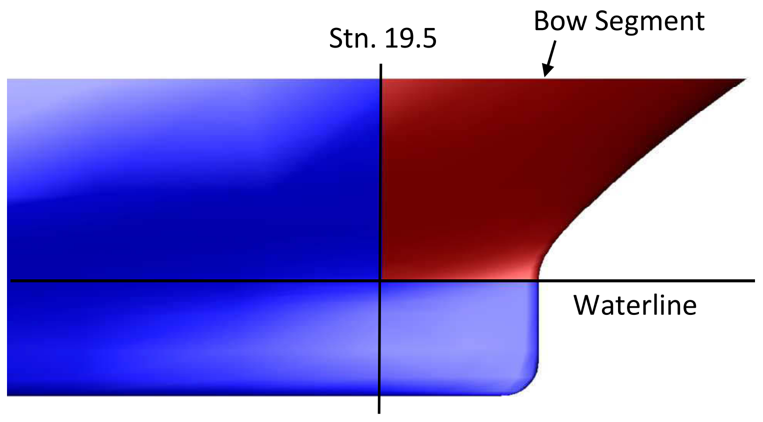

Figure 7: *Location of bow segment*

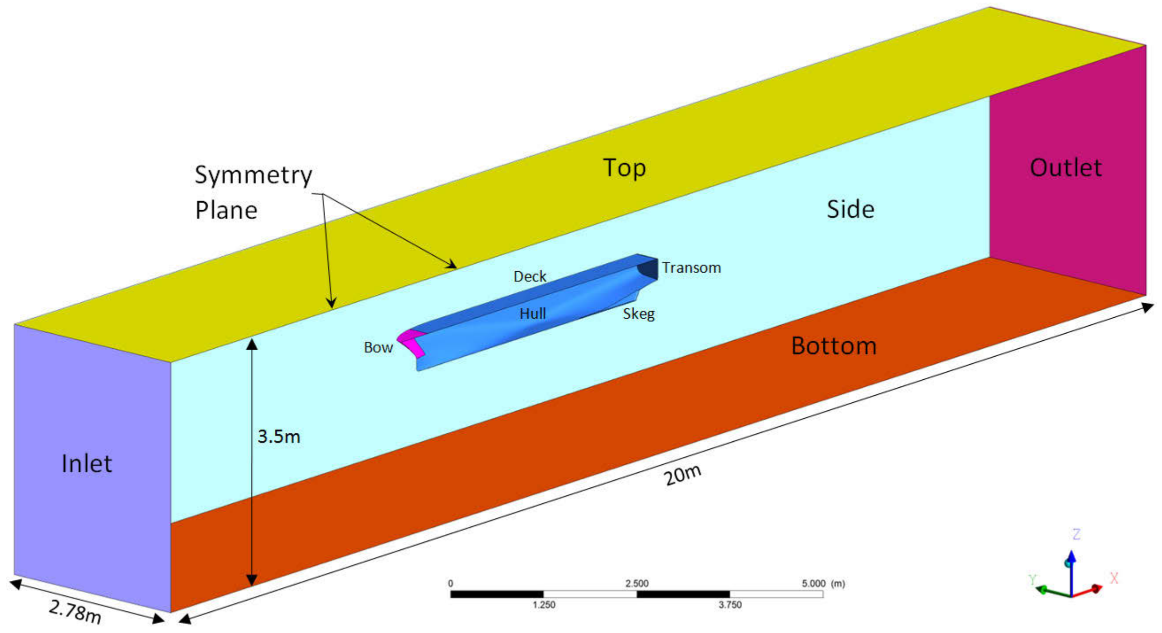

Figure 8: *Simulation domain*

3.2 Mesh

The mesh was created using CFX–Mesh® (part of ANSYS WorkBench® 11.0). It was primarily built from unstructured tetrahedrals except for a prismatic boundary layer and transition cells near the no-slip surfaces of the ship hull. The mesh density was heavily concentrated near the hull of the ship with extra refinement near the bow segment. The mesh also concentrated near the air/water interface. The ability of the volume-of-fluid (VOF) method to yield good results for free surface calculations depends on the available mesh density at the interface. The demand for mesh density is further increased with the presence of waves. There must be sufficient mesh density not just at the still water plane but in the full extent from the lowest trough to the highest crest. These demands result in a grid that can be very computationally expensive. As a compromise for the present simulations, a fine wave capturing mesh was used from the domain inlet to about midway along the ship's length. From there the density gradually decreases towards the outlet. The loss of wave fidelity past amidships was not a concern as the primary objective of the simulations was to predict the forces and pressures from slamming at the bow segment.

The CFX–Mesh parameters used to create the mesh are given in Annex B. Figure 9 shows the mesh on the symmetry and inlet boundaries of the domain where the increased resolution along the waterplane can be seen. Figure 10 shows a close up view of the mesh near the bow segment. The minimum edge size of this mesh is ~5 mm. Mesh information is given in Table 2.

Table 2: *Mesh information*

Nodes	3,141,149
Elements	17,897,581
Tetrahedra	17,795,626
Wedges	98,853
Pyramids	3,102

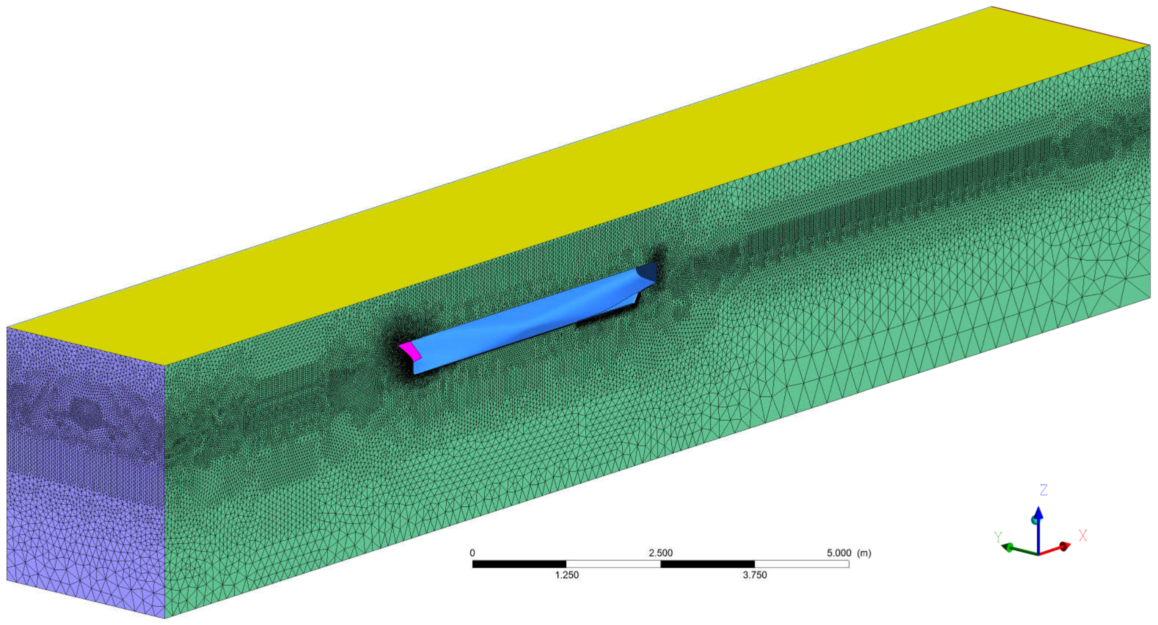

Figure 9: *Mesh on symmetry and inlet planes*

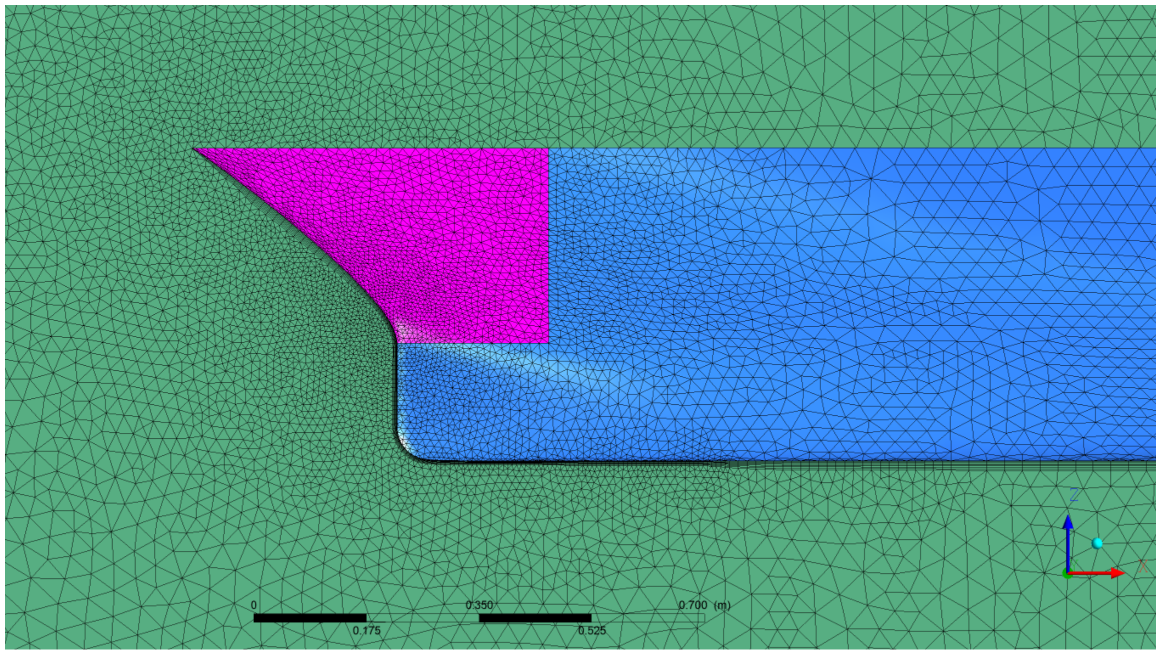

Figure 10: *Mesh around bow segment*

3.3 Set-up

The following boundary conditions were applied (see Figure 8). There was a specialized inlet boundary condition at the front of the domain, a symmetry plane at the vessel's centreline, and openings with relative pressure entrainment at the back and top of the domain. The sides and bottom of the domain were free-slip walls. The ship surfaces were no-slip walls except for the transom and deck of the vessel which were free-slip (these surfaces were not 'wet' during the simulation).

The present simulations were only conducted for the captive tests in head waves. For these tests the vessel was held fixed to the carriage while moving through the wave field. This simplifies the comparison as vessel motions do not need to be calculated or reproduced in the computational domain.

Although these simulations were run using a Shear Stress Transport (SST) turbulence model, the nature of the bow flare slamming impacts was insensitive to turbulence. This was confirmed by DNV who performed similar simulations with STAR–CCM+® and achieved good slam impact results without turbulence modelling disabled.

The inlet boundary was set to define velocity components and VOF of the incoming flow. This is a combination of the ships forward speed, and the incident wave field. In order to define a specific irregular wave field, specialized functions had to be created that could read in external wave data, and calculate the appropriate velocity and wave height data for any time and position at the inlet. The wave elevations from the model tests were recorded and converted to a fast Fourier transform (FFT) representation (see Annex A). This was saved as a text file containing the frequency, amplitude and phase data for each component of the FFT. At run time, an initialization routine would read this file and store the data to memory. Then, at any point in the simulation a function call could be made that would return the appropriate wave orbital velocities and wave height from the reconstructed wave for the given position and time. These functions were created using CFX User Fortran [3] and are given in Annex E and Annex D.

Transient simulations have the added complication of requiring a lot of disk space, as information at each time step needs to be saved for later analysis. The file size required to save all solution variables at a given timestep for this simulation is approximately 8.75 GB. This makes saving all information at every timestep (1200 in total) impractical. Instead, selected variables such as VOF, velocity, and pressure were saved for every 3rd timestep. The transient files with selected variables were still a respectable 250 MB each. Reducing transient file size and frequency also speeds up the simulation, as writing the large files stalls computations. It was found that the given timestep size and saving frequency were near the minimum necessary to produce smooth animations of the simulation in post-processing.

The simulations were run using a second order transient scheme with a constant time step size of 0.05 s. In order to avoid problems with start-up transients, the simulation would run for a few timesteps (0.5 s model scale) before the wave generation would start. Without this delay, simulations were prone to crash. A 60 s (360 s full scale) simulation for a 20 knot forward speed run into irregular head waves took approximately 7 days using 40 Itanium 1.6 GHz processors in a shared memory SGI Altix 4700 computer.

3.4 Analysis

As mentioned in Section 3.3, only selected field variables (pressure, velocity and volume fraction) were saved in transient files every 3rd timestep. This was sufficient for producing animations, but could skip over features in key data such as bow forces. For these, CFX provides 'monitors' which report desired data at every timestep to a log file. Monitors were set for hull and bow forces and moments, pressures at the pressure transducer locations on the bow segment[1], and volume fractions at points ahead of the bow used to calculate wave elevations (virtual wave probes).

The virtual wave probes consisted of multiple points in a vertical column that monitored volume fraction. As many were required, a Matlab® script was used to generate a supplementary CFX Command Language (CCL) file which appends to the simulation definition file at the beginning of a run. Details on this CCL file and the procedure for extracting wave elevation from the monitor points data is given in Annex G. Four virtual wave probes were used that corresponded to the positions of probes 1, 2, 10, and 16 in the experiments (probe positions are given in Annex A).

[1]One of the pressure transducers was located on the port side of the bow segment in the physical model. This was mirrored to the starboard side in simulations as only half the ship was represented. Actual locations used in simulation are given in Appendix F.

4 Results

One of the important findings from this study was in relation to wave generation in RANS simulations. The ability to extract hydrodynamic loads on the vessel strongly depended on the quality of these generated waves. For example, in an early attempt at simulating run 207004, a grid was created which concentrated mesh density at the free surface directly ahead of the hull but allowed the density to grow coarser toward the sides of the domain. This grid produced erroneous hump waves as shown in Figure 11 (Time = 180 s full scale). Once this grid resolution was both increased and made uniform across the domain (doubling the element count), the waves improved significantly as shown in Figure 12.

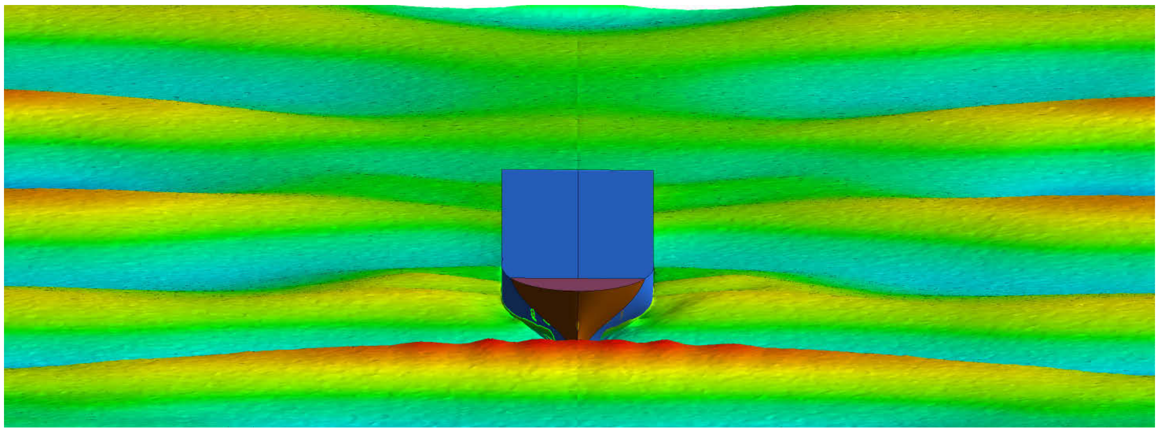

Figure 11: *Early run with poor grid resolution*

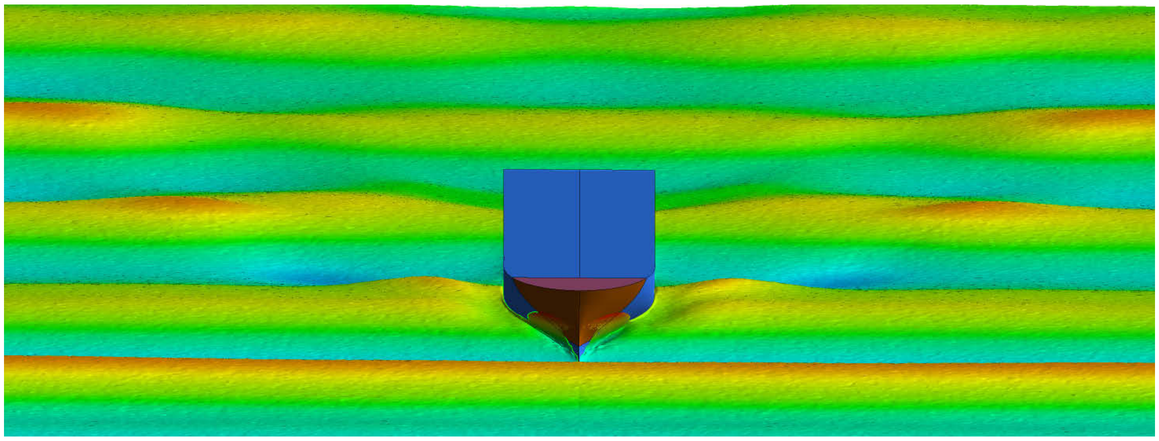

Figure 12: *Later run with proper grid resolution*

In general, the simulations were able to accurately reproduce the slam force impacts on the bow segment of the model as compared with the model tests. Figure 13 shows the x-forces on the bow segment from the experiments, from the CRS WHIP software 2D-BEM, and the CFD simulations. Also shown in the figure are the wave elevations from probe 16 (the closest wave probe to the bow). The simulated impacts show good agreement with experiments and are considerably better than those predicted by the CRS WHIP software. It turned out that predicting slam impacts with RANS CFD was fairly straightforward, the difficult part was the wave generation.

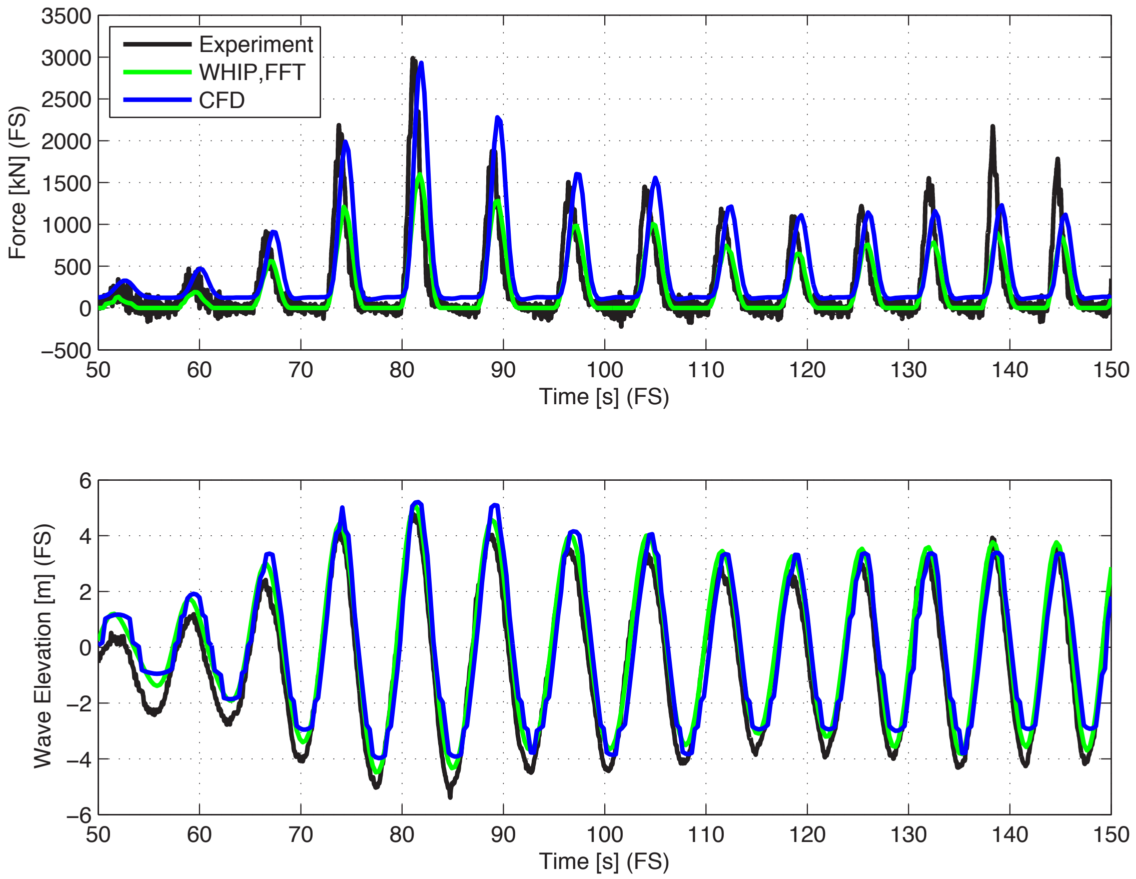

Figure 13: *X-Force on bow segment*

Experiment run 207004 simulated here employed a wave sweep function which starts with longer wavelength waves that gradually get smaller while retaining the same wave height. The waves therefore get steeper as the run progresses. Figure 13 shows the early part of the run where the CFD was able to generate the required waves reasonably well. However, later in the run when the waves became steeper, the simulated waves began to break down and could not propagate with the desired steepness. Figure 14 shows a time segment later in the run where the highest slam impact was measured during the experiment (t=193 s). The CFD generated waves in the plot are all of smaller amplitude (and steepness) than the experiments and input FFT function. All of the resulting slam impacts therefore under-predict the experiments. Full plots of the results for this run are available in Annex I.

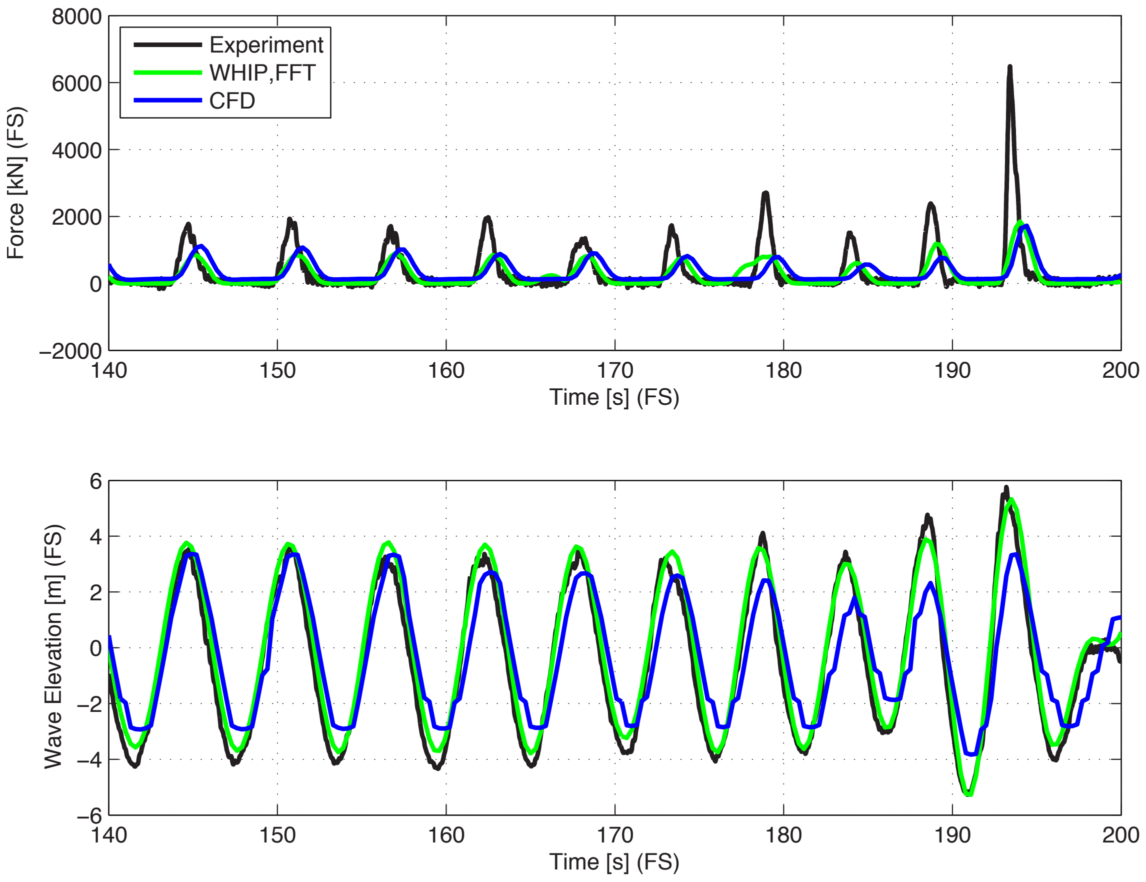

Figure 14: *X-force on bow segment*

The difficulties in generating the steeper waves lead to simulating another of the runs from model tests that employed a focused wave. For this run a single wave group was generated by the wave-maker that coalesce into one large steep wave at a time and position to impact the model. Shown in Figure 15 is the wave profile from run 211002 showing the large steep focused wave just before impact. The resulting CFD generated wave was not even close. Several attempts were made adjusting the components of the input FFT as well as solver parameters such as timestep size (using the same mesh as the previous simulations) but no significant improvement could be made. Further work is required to either find ways of generating and propagating steep waves in RANS CFD, or, to identify the limits of what can be achieved using these methods.

The CFD simulations also monitored pressures at points on the bow segment corresponding to pressure transducer locations on the physical model. These results, as with the force results, were in good agreement with the experimental data as shown in Figure 16. Complete results for all pressure transducer locations are given in Annex H.

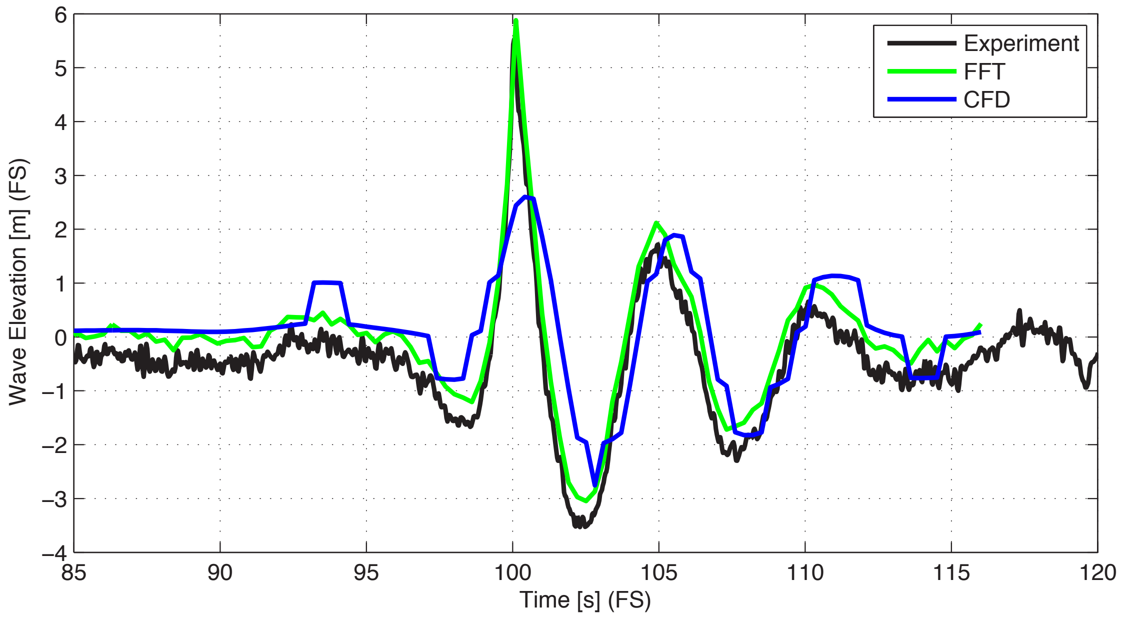

Figure 15: *Wave elevations at probe #16 for run 211002*

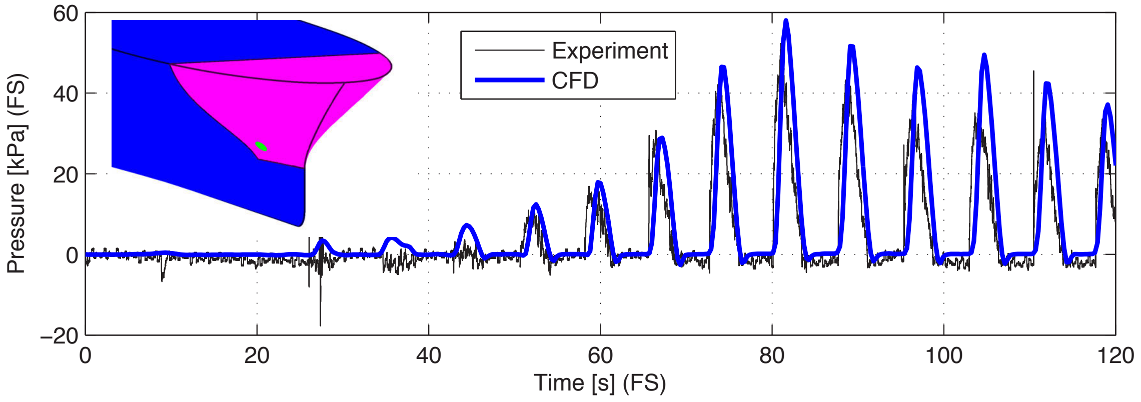

Figure 16: *Pressure transducer #1 data*

5 Summary and Conclusions

A ship sailing in a seaway can often experience large sudden loading from wave impacts called slamming. It is important for ship design, operation, and life cycle management to be able to quantify the effects of slamming on the vessel's structure. Common seakeeping prediction tools used for determining hydrodynamic loading of ship in a seaway cannot calculate slamming directly and either omit its effects or attempts to estimate them through other means. This study uses a more advanced computational method to simulate wave impacts that has been shown to be in good agreement with experiments with respect to bow flare slamming loads. Although computational expensive , RANS CFD can be used to examine specific slamming scenarios, that along with model tests, could help develop and validate models in general seakeeping prediction codes.

Incident waves in the simulation proved to be a key factor in the ability to correctly predict slamming. Large steep waves could not be correctly generated/propagated in the current setup. Further work is needed to overcome this, or, to determine the limits of applicability for this approach.

References

[1] Kapsenberg, G. and Thornhill, E. (2010), A Practical Approach to Ship Slamming in Waves, In *Proceedings of the 28th Symposium on Naval Hydrodynamics*, ONR, Pasadena, California.

[2] Sames, P., Kapsenberg, G., and Corrignan, P. (2001), Prediction of Bow Door Loads in Extreme Wave Conditions, In *Proceeding of Conference on Design and Operation for Abnormal Conditions II*, RINA, London.

[3] (2009), ANSYS CFX-Solver Modeling Guide (Release 12.1), ANSYS.

Annex A: Wave Probe Analysis

During the physical model experiments of the ELAST ferry in 2007 for the WHIP working group, a wave probe array was used to record the wave elevation histories. The array consisted of 16 wave probes in a line oriented with the direction of wave propagation. The array was fixed to the tow carriage and moved with the hull for each run. The probes were numbered from 1 to 16 with probe 16 closest to the hull. The spacing (ship scale) of the probes in the array are given in Table A.1. The distances of the harp to a reference point on the centreline at Station 19.5 for each wave heading are given in Figure A.3.

In order to reproduce the same wave field in the numerical simulations, this wave data was converted to a summation of cosine waves with various amplitudes, frequencies, and phase angles according to equation A.1.

$$z(x,t) = \sum_{i=0}^{n} A_i cos(\omega_i - k_i x + \varphi_i) \tag{A.1}$$

where,

z	$=$ is the wave elevation
A_i	$=$ is a component wave amplitude
ω_i	$=$ is a component wave frequency
k_i	$=$ is a component wave number
φ_i	$=$ is a component phase angle
x	$=$ is a location in the direction of wave propagation
t	$=$ is time
n	$=$ is the total number of cosine wave components

Simulations may require that the wave input be given for a wave probe fixed at a specific location (such as the vessel origin). The experimental data must therefore be translated for a moving point near the bow to the desired fixed point at t=0. First, the wave elevation data set for each probe is run through a FFT. As the probe is moving, the FFT produces the encounter frequencies and not the wave frequencies.

$$[A_n, \omega_{e_i}, \varphi_n] = FFT(z(t)_{Probe\#}) \tag{A.2}$$

To determine the wave frequencies, the wave number (equation A.3) is substituted into the equation A.4 for encounter frequency to yield equation A.5. Solving this for wave frequency yields equation A.6.

$$k = \frac{\omega^2}{g} \tag{A.3}$$

$$\omega_e = \omega - k\, v_s\, cos(\mu) \tag{A.4}$$

$$\omega_e^2\, v_s\, g^{-1}\, cos(\mu) - \omega_e - \omega = 0 \tag{A.5}$$

$$\omega = \frac{g - \sqrt{g^2 - 4\, v_s\, \omega_e\, cos(\mu)}}{2\, v_s\, cos(\mu)} \tag{A.6}$$

where,

k	= is the wave number
g	= is the acceleration due to gravity
v_s	= is the ship velocity
μ	= is the wave direction (180°= head seas)
ω_e	= is the wave encounter frequency

In order to improve the input, an average was taken of the wave field measured by the 16 probes in the array. This was done by reconstructing the wave data for each probe at the probe 16 location using equation A.7 and calculating the resulting average wave. Figure A.1 shows the a sample of raw data from the wave probe harp during Test 207004, while Figure A.2 shows reconstructed data for each probe at the probe 16 location, along with the average wave in red.

$$z(x,t)_{p16} = \sum_{i=0}^{n} A_i\, cos\left(\omega_i t - k_i\left(v_s t\, cos(\mu) + x_{p16}\right) + \varphi_i\right) \tag{A.7}$$

where,

z_{p16}	= is the wave elevation at the probe 16 position
x_{p16}	= is the distance in the direction of wave propagation from the given wave probe to probe 16

A FFT was then performed on this average wave. The wave frequencies were calculated using equation A.6 and the wave field was translated from the probe 16 location to the origin by adjusting the phase angles according to equation **??**. The FFT produces $n/2$ sets of $\{\omega, A, \varphi\}$ (where n is the number of data points in the time history). This number is usually reduced to no more than 200 by excluding the high frequency components before being used in a simulation.

$$\varphi_0 = \varphi - kx_0 \tag{A.8}$$

where,

φ_0 = is the phase angle for a stationary wave probe at the origin
x_0 = is the distance in the direction of wave propagation from the given wave probe to the origin

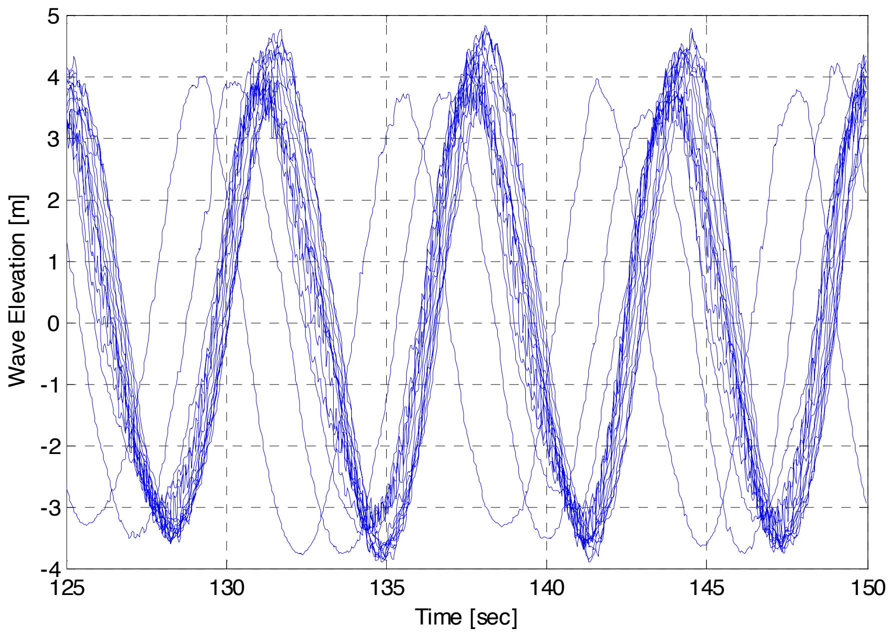

Figure A.1: *Wave probe wave elevation data*

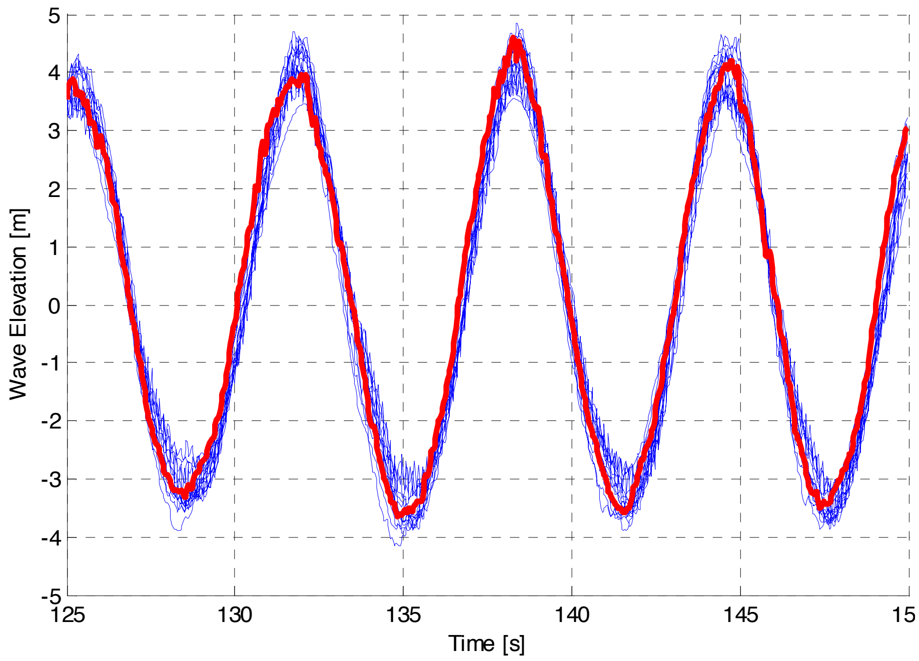

Figure A.2: *Wave reconstructions at probe 16 location*

Table A.1: *Main dimensions of ELAST ferry*

Probe #	Distance to Probe 1 [m]	Probe #	Distance to Probe 1 [m]
Probe 1	0.0	Probe 9	59.4
Probe 2	36.0	Probe 10	61.2
Probe 3	48.6	Probe 11	63.0
Probe 4	50.4	Probe 12	64.8
Probe 5	52.2	Probe 13	66.6
Probe 6	54.0	Probe 14	68.4
Probe 7	55.8	Probe 15	70.2
Probe 8	57.6	Probe 16	72.0

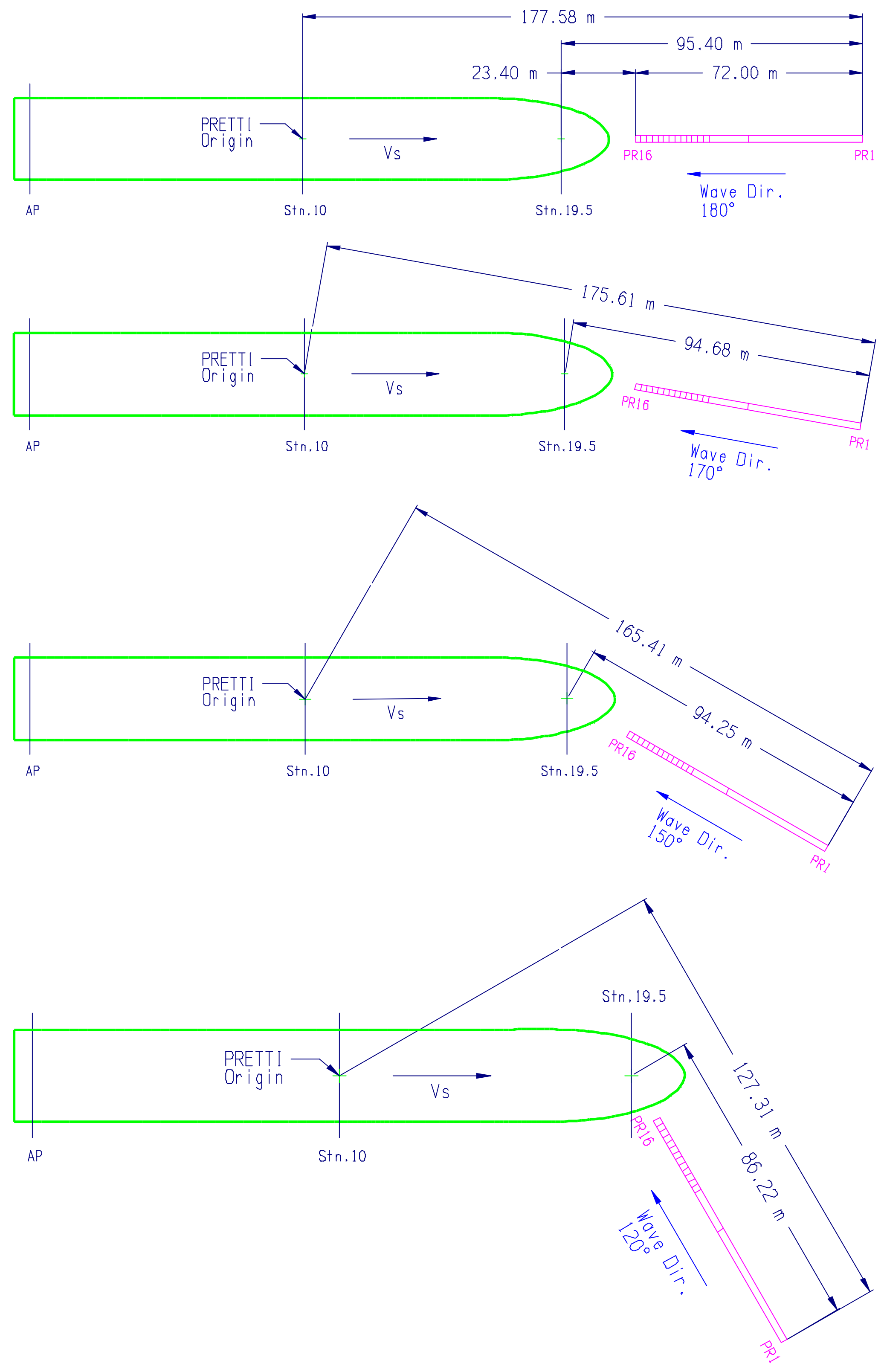

Figure A.3: *Wave probe array orientations*

This page intentionally left blank.

Annex B: CFX Mesh Settings

The following are the CFX Mesh parameters used for the ELAST ferry in head seas simulations. They are given in the ANSYS CCL format that can be imported/exported as plain text files by CFX. Note that the special character ‘#’ indicates a comment line and the character ‘\’ indicates the given line continues onto the next.

```
COMMAND FILE:
  Meshing CCL Version = 12.1
END
MESHING ACTION: CADCheck
  Meshing Problem = Default
  Option = Check Geometry
  Short Edge Limit = 0.02 [m]
  Sliver Factor Limit = 25
END
MESHING STRATEGY: Default
  BODY MESHER:
    Option = Advancing Front
  END
  CONTROLS:
    # The mesh at the waterline had to be fairly dense.
    # But not so dense as to make it too big.
    # Two mesh controls were used to refine the waterplane
    # mesh. The first was fine and extended from the inlet
    # to the stern (using two triangle controls that
    # form a rectangle over the desired area). The
    # second waterplane control was the rest of the domain
    # but at a reduced resolution as the interesting bit
    # had already occured and loss of resolution was an issue.
    POINT SPACING: waterline
      Expansion Factor = 1.05
      Length Scale = 0.03 [m]
      Radius of Influence = 0.15 [m]
    END
    POINT SPACING: waterline2
      Expansion Factor = 1.1
      Length Scale = 0.03 [m]
      Radius of Influence = 0.11 [m]
    END
    TRIANGLE CONTROL: waterplane1
      Option = Uniform
      Point 1 = -5 [m], 2.8 [m], 0 [m]
      Point 2 = -5 [m], 0 [m], 0 [m]
      Point 3 = 5 [m], 0 [m], 0 [m]
      Point Spacing = waterline
    END
    TRIANGLE CONTROL: waterplane2
      Option = Uniform
      Point 1 = -5 [m], 2.8 [m], 0 [m]
      Point 2 = 5 [m], 2.8 [m], 0 [m]
      Point 3 = 5 [m], 0 [m], 0 [m]
      Point Spacing = waterline
    END
    TRIANGLE CONTROL: waterplane3
      Option = Uniform
      Point 1 = 5 [m], 2.8 [m], 0 [m]
      Point 2 = 5 [m], 0 [m], 0 [m]
      Point 3 = 15 [m], 0 [m], 0 [m]
      Point Spacing = waterline2
    END
    TRIANGLE CONTROL: waterplane4
      Option = Uniform
      Point 1 = 5 [m], 2.8 [m], 0 [m]
      Point 2 = 15 [m], 2.8 [m], 0 [m]
      Point 3 = 15 [m], 0 [m], 0 [m]
      Point Spacing = waterline2
    END
  END
  EXCLUDED REGIONS:
    Short Edge Tolerance = 0.02 [m]
  END
  FACE MESHER:
    Option = Delaunay
  END
  Global Scaling = 1
  # An inflation layer was applied to all wetted parts of the
  # hull, excluding the transom and deck.
  INFLATION:
    Expansion Factor = 1.1
    INFLATED BOUNDARY: Inflated Boundary 1
      Location = F42.40,F43.40,F50.40
      Maximum Thickness = 0.015 [m]
    END
    Inflation Thickness Multiplier = 1
    Minimum External Angle = 10.0 [deg]
    Minimum Internal Angle = 2.5 [deg]
    Number of Inflated Layers = 8
    Number of Spreading Iterations = 0
    Option = Total Thickness
  END
  Option = Advancing Front and Inflation 3D
  PROXIMITY:
    Edge Proximity = Yes
    Face Proximity = No
  END
  SPACING:
    BODY SPACING: Default Body Spacing
      Constant Edge Length = 1 [m]
      Option = Constant
    END
    Default Body Spacing = Default Body Spacing
    Default Face Spacing = Default Face Spacing
    FACE SPACING: Default Face Spacing
      Angular Resolution = 30 [deg]
      Maximum Edge Length = 1 [m]
      Minimum Edge Length = 0.005 [m]
      Option = Angular Resolution
    END
    # Bow section was given extra resolution.
    FACE SPACING: bowsection
      Angular Resolution = 18 [deg]
      Location = F50.40
      Maximum Edge Length = 0.015 [m]
      Minimum Edge Length = 0.0075 [m]
      Option = Angular Resolution
      VOLUMETRIC EFFECT:
        Expansion Factor = 1.05
        Radius of Influence = 0.05 [m]
      END
    END
    # The skeg had some tight features that also required
    # extra resolution.
    FACE SPACING: skeg
      Angular Resolution = 18 [deg]
      Location = F42.40
      Maximum Edge Length = 0.05 [m]
      Minimum Edge Length = 0.005 [m]
      Option = Angular Resolution
      VOLUMETRIC EFFECT:
        Expansion Factor = 1.2
        Radius of Influence = 0.03 [m]
      END
    END
    FACE SPACING: skegbottom
      Constant Edge Length = 0.01 [m]
      Location = F82.40
      Option = Constant
      VOLUMETRIC EFFECT:
        Expansion Factor = 1.2
        Radius of Influence = 0.05 [m]
      END
    END
    FACE SPACING: transom
      Constant Edge Length = 0.015 [m]
      Location = F44.40
      Option = Constant
      VOLUMETRIC EFFECT:
        Expansion Factor = 1.1
        Radius of Influence = 0.05 [m]
      END
    END
  END
END
```

Annex C: CFX Solver Settings

The following are the solver parameters used for the ELAST ferry in head seas simulations. They are given in the ANSYS CCL format that can be imported/exported as plain text files by CFX. Note that the special character '#' indicates a comment line and the character '\' indicates the given line continues onto the next.

```
LIBRARY:
  CEL:
    EXPRESSIONS:

      # Experimental run number being simulated (for reference)
      runname = 207004

      # full scale / model scale ratio
      scale = 36

      # full scale ship velocity (simulation is at model scale)
      shipvel = 20[knot]

      # convert ship knots to model scale m/s
      boatvel = shipvel*(0.514444[m/s/knot]/sqrt(scale))

      # how long the simulation runs in full scale time
      fullscaletime = 360[s]

      # total simulation time in model scale time
      totaltime = fullscaletime/sqrt(scale)

      # simulation time step size (model scale time)
      tstep = 0.05 [s]

      # EdVisRat and FractI are used to set/initialize
      # turbulence at the inlet, outlet, domain
      EdVisRat = 30.0
      FractI = 0.01

      # Functions used to set volume-of-fluid fraction
      VFAir = step((z-waterlevelz)/1[m])
      VFWater = 1-VFAir

      # calm water hydrostatic pressure function
      hydropres = waterdensity*g*VFWater*(waterlevelz-z)
      airdensity = 1.185[kg m^-3]
      waterdensity = 997[kg m^-3]

      # the z coordinate value of the still waterline
      waterlevelz = 0[m]
      # z coordinate of domain bottom, can be increased to
      # eliminate bottom effects in wave equations
      bottomz = 4*(-2.425[m])

      # position of inlet relative to c.g. in direction of
      # wave propagation.
      # i.e. waves traveling bow to stern, inlet is located
      # 5m ahead of the bow (model scale), and the c.g. is
      # at station 9.5, Lpp=173m
      inletx = -5.0[m]-(9.5/20*173[m]/scale)

      # water volume fraction at inlet, accounting for
      # incident wave heights
      watinlvf = step(1.0[m^-1]*(waterlevelz+waveh-z))

      # delay from start of simulation to when waves will start.
      # a small time is required to settle out initial
      # simulations transients, otherwise it may crash.
      wavedelay = 0.5[s]

      # used to scale incident waves
      wavefudge = 1

      # wave height, and orbital velocities at inlet
      # see description of user defined getwave() function.
      waveh = \
         wavefudge*step(1[s^-1]*(t-wavedelay))*\
         getwave(1,(t-wavedelay),z,inletx,waterlevelz,\
         bottomz,boatvel,scale)*1[m]
      wavu = \
         wavefudge*step(1[s^-1]*(t-wavedelay))*\
         getwave(2,(t-wavedelay),z,inletx,waterlevelz,\
         bottomz,boatvel,scale)*1[m/s]
      wavw = \
         wavefudge*step(1[s^-1]*(t-wavedelay))*\
         getwave(3,(t-wavedelay),z,inletx,waterlevelz,\
         bottomz,boatvel,scale)*1[m/s]

    END
    FUNCTION: getwave
      Argument Units = [],[s],[m],[m],[m],[m],[m/s],[]
      Option = User Function
      Result Units = []
      User Routine Name = getwaveout
    END
  END
  MATERIAL: Air at 25 C
    Material Description = Air at 25 C and 1 atm (dry)
    Material Group = Air Data, Constant Property Gases
    Option = Pure Substance
    Thermodynamic State = Gas
```

```
    PROPERTIES:
      Option = General Material
      EQUATION OF STATE:
        Density = 1.185 [kg m^-3]
        Molar Mass = 28.96 [kg kmol^-1]
        Option = Value
      END
      DYNAMIC VISCOSITY:
        Dynamic Viscosity = 1.831E-05 [kg m^-1 s^-1]
        Option = Value
      END
    END
  END
  MATERIAL: Water
    Material Description = Water (liquid)
    Material Group = Water Data, Constant Property Liquids
    Option = Pure Substance
    Thermodynamic State = Liquid
    PROPERTIES:
      Option = General Material
      EQUATION OF STATE:
        Density = 997.0 [kg m^-3]
        Molar Mass = 18.02 [kg kmol^-1]
        Option = Value
      END
      DYNAMIC VISCOSITY:
        Dynamic Viscosity = 8.899E-4 [kg m^-1 s^-1]
        Option = Value
      END
    END
  END
  # Both the getirregwave and initval functions are located
  # in the same file mdofv14.F which is compiled and linked
  # before running the solver (see CFX manual for details
  # on how user fortran for more details).
  USER ROUTINE DEFINITIONS:
    USER ROUTINE: getwaverout
      Calling Name = getirregwave
      Library Name = mdofv14
      Library Path = /usr/people/thornhil/userfortran
      Option = User CEL Function
    END
    USER ROUTINE: initvalrout
      Calling Name = initval
      Junction Box Location = User Input
      Library Name = mdofv14
      Library Path = /usr/people/thornhil/userfortran
      Option = Junction Box Routine
    END
  END
END
```

```
FLOW: Flow Analysis 1
  SOLUTION UNITS:
    Angle Units = [rad]
    Length Units = [m]
    Mass Units = [kg]
    Solid Angle Units = [sr]
    Temperature Units = [K]
    Time Units = [s]
  END
  ANALYSIS TYPE:
    Option = Transient
    EXTERNAL SOLVER COUPLING:
      Option = None
    END
    INITIAL TIME:
      Option = Automatic with Value
      Time = 0 [s]
    END
    TIME DURATION:
      Option = Total Time
      Total Time = totaltime
    END
    TIME STEPS:
      Option = Timesteps
      Timesteps = tstep
    END
  END
  DOMAIN: Domain 1
    Coord Frame = Coord 0
    Domain Type = Fluid
    Location = Assembly
    BOUNDARY: bottom
      Boundary Type = WALL
      Location = F39.30
      BOUNDARY CONDITIONS:
        MASS AND MOMENTUM:
          Option = Free Slip Wall
        END
      END
    END
    BOUNDARY: bow
      Boundary Type = WALL
      Location = F41.30
      BOUNDARY CONDITIONS:
        MASS AND MOMENTUM:
          Option = No Slip Wall
        END
        WALL ROUGHNESS:
          Option = Smooth Wall
        END
      END
```

```
END
BOUNDARY: bowdeck
  Boundary Type = WALL
  Location = F31.30
  BOUNDARY CONDITIONS:
    MASS AND MOMENTUM:
      Option = Free Slip Wall
    END
  END
END
BOUNDARY: deck
  Boundary Type = WALL
  Location = F32.30
  BOUNDARY CONDITIONS:
    MASS AND MOMENTUM:
      Option = Free Slip Wall
    END
  END
END
BOUNDARY: hull
  Boundary Type = WALL
  Location = F35.30
  BOUNDARY CONDITIONS:
    MASS AND MOMENTUM:
      Option = No Slip Wall
    END
    WALL ROUGHNESS:
      Option = Smooth Wall
    END
  END
END
BOUNDARY: inlet
  Boundary Type = INLET
  Location = F36.30
  BOUNDARY CONDITIONS:
    FLOW REGIME:
      Option = Subsonic
    END
    MASS AND MOMENTUM:
      Option = Cartesian Velocity Components
      U = wavu*watinlvf+boatvel
      V = 0 [m s^-1]
      W = wavw*watinlvf
    END
    TURBULENCE:
      Eddy Viscosity Ratio = EdVisRat
      Fractional Intensity = FractI
      Option = Intensity and Eddy Viscosity Ratio
    END
  END
  FLUID: Air at 25 C
```

```
      BOUNDARY CONDITIONS:
        VOLUME FRACTION:
          Option = Value
          Volume Fraction = 1.0-watinlvf
        END
      END
    END
    FLUID: Water
      BOUNDARY CONDITIONS:
        VOLUME FRACTION:
          Option = Value
          Volume Fraction = watinlvf
        END
      END
    END
  END
  BOUNDARY: outlet
    Boundary Type = OPENING
    Location = F37.30
    BOUNDARY CONDITIONS:
      FLOW REGIME:
        Option = Subsonic
      END
      MASS AND MOMENTUM:
        Option = Entrainment
        Relative Pressure = hydropres
      END
      TURBULENCE:
        Eddy Viscosity Ratio = EdVisRat
        Fractional Intensity = FractI
        Option = Intensity and Eddy Viscosity Ratio
      END
    END
    FLUID: Air at 25 C
      BOUNDARY CONDITIONS:
        VOLUME FRACTION:
          Option = Zero Gradient
        END
      END
    END
    FLUID: Water
      BOUNDARY CONDITIONS:
        VOLUME FRACTION:
          Option = Zero Gradient
        END
      END
    END
  END
  BOUNDARY: side
    Boundary Type = SYMMETRY
    Location = F40.30
```

```
END
BOUNDARY: sym
  Boundary Type = SYMMETRY
  Location = F33.30
END
BOUNDARY: top
  Boundary Type = OPENING
  Location = F38.30
  BOUNDARY CONDITIONS:
    FLOW REGIME:
      Option = Subsonic
    END
    MASS AND MOMENTUM:
      Option = Entrainment
      Relative Pressure = hydropres
    END
    TURBULENCE:
      Option = Low Intensity and Eddy Viscosity Ratio
    END
  END
  FLUID: Air at 25 C
    BOUNDARY CONDITIONS:
      VOLUME FRACTION:
        Option = Value
        Volume Fraction = VFAir
      END
    END
  END
  FLUID: Water
    BOUNDARY CONDITIONS:
      VOLUME FRACTION:
        Option = Value
        Volume Fraction = VFWater
      END
    END
  END
END
BOUNDARY: transom
  Boundary Type = WALL
  Location = F34.30
  BOUNDARY CONDITIONS:
    MASS AND MOMENTUM:
      Option = Free Slip Wall
    END
  END
END
DOMAIN MODELS:
  BUOYANCY MODEL:
    Buoyancy Reference Density = 1.185 [kg m^-3]
    Gravity X Component = 0 [m s^-2]
    Gravity Y Component = 0 [m s^-2]
```

```
      Gravity Z Component = -g
      Option = Buoyant
      BUOYANCY REFERENCE LOCATION:
        Cartesian Coordinates = -3 [m], 0 [m], 2 [m]
        Option = Cartesian Coordinates
      END
    END
    DOMAIN MOTION:
      Option = Stationary
    END
    MESH DEFORMATION:
      Option = None
    END
    REFERENCE PRESSURE:
      Reference Pressure = 1 [atm]
    END
  END
  FLUID DEFINITION: Air at 25 C
    Material = Air at 25 C
    Option = Material Library
    MORPHOLOGY:
      Option = Continuous Fluid
    END
  END
  FLUID DEFINITION: Water
    Material = Water
    Option = Material Library
    MORPHOLOGY:
      Option = Continuous Fluid
    END
  END
  FLUID MODELS:
    COMBUSTION MODEL:
      Option = None
    END
    FLUID: Air at 25 C
      FLUID BUOYANCY MODEL:
        Option = Density Difference
      END
    END
    FLUID: Water
      FLUID BUOYANCY MODEL:
        Option = Density Difference
      END
    END
    HEAT TRANSFER MODEL:
      Homogeneous Model = True
      Option = None
    END
    THERMAL RADIATION MODEL:
      Option = None
    END
    TURBULENCE MODEL:
      Option = SST
      BUOYANCY TURBULENCE:
        Option = None
      END
    END
    TURBULENT WALL FUNCTIONS:
      Option = Automatic
    END
  END
  FLUID PAIR: Air at 25 C | Water
    INTERPHASE TRANSFER MODEL:
      Option = Free Surface
    END
    MASS TRANSFER:
      Option = None
    END
    SURFACE TENSION MODEL:
      Option = None
    END
  END
  MULTIPHASE MODELS:
    Homogeneous Model = On
    FREE SURFACE MODEL:
      Interface Compression Level = 2
      Option = Standard
    END
  END
END
INITIALISATION:
  Option = Automatic
  FLUID: Air at 25 C
    INITIAL CONDITIONS:
      VOLUME FRACTION:
        Option = Automatic with Value
        Volume Fraction = VFAir
      END
    END
  END
  FLUID: Water
    INITIAL CONDITIONS:
      VOLUME FRACTION:
        Option = Automatic with Value
        Volume Fraction = VFWater
      END
    END
  END
  INITIAL CONDITIONS:
    Velocity Type = Cartesian
    CARTESIAN VELOCITY COMPONENTS:
```

```
      Option = Automatic with Value
      U = boatvel
      V = 0 [m s^-1]
      W = 0 [m s^-1]
    END
    STATIC PRESSURE:
      Option = Automatic with Value
      Relative Pressure = hydropres
    END
    TURBULENCE INITIAL CONDITIONS:
      Option = Intensity and Eddy Viscosity Ratio
      EDDY VISCOSITY RATIO:
        Eddy Viscosity Ratio = EdVisRat
        Option = Automatic with Value
      END
      FRACTIONAL INTENSITY:
        Fractional Intensity = FractI
        Option = Automatic with Value
      END
    END
  END
END
OUTPUT CONTROL:
  BACKUP RESULTS: Backup Results 1
    File Compression Level = Default
    Option = Standard
    OUTPUT FREQUENCY:
      Option = Timestep Interval
      Timestep Interval = 100
    END
  END
  MONITOR OBJECTS:
    Monitor Coefficient Loop Convergence = False
    MONITOR BALANCES:
      Option = Full
    END
    MONITOR FORCES:
      Option = Full
    END
    MONITOR PARTICLES:
      Option = Full
    END
    MONITOR POINT: bowdeckmom
      Expression Value = -2*torque_y()@bowdeck
      Option = Expression
    END
    MONITOR POINT: bowdeckx
      Expression Value = 2*force_x()@bowdeck
      Option = Expression
    END
    MONITOR POINT: bowdeckz
      Expression Value = 2*force_z()@bowdeck
      Option = Expression
    END
    MONITOR POINT: bowmom
      Expression Value = -2*torque_y()@bow
      Option = Expression
    END
    MONITOR POINT: bowx
      Expression Value = 2*force_x()@bow
      Option = Expression
    END
    MONITOR POINT: bowz
      Expression Value = 2*force_z()@bow
      Option = Expression
    END
    MONITOR POINT: deckmom
      Expression Value = -2*torque_y()@deck
      Option = Expression
    END
    MONITOR POINT: deckx
      Expression Value = 2*force_x()@deck
      Option = Expression
    END
    MONITOR POINT: deckz
      Expression Value = 2*force_z()@deck
      Option = Expression
    END
    MONITOR POINT: fullscaletimelog
      Expression Value = t*sqrt(scale)
      Option = Expression
    END
    MONITOR POINT: hullmom
      Expression Value = -2*torque_y()@hull
      Option = Expression
    END
    MONITOR POINT: hullx
      Expression Value = 2*force_x()@hull
      Option = Expression
    END
    MONITOR POINT: hullz
      Expression Value = 2*force_z()@hull
      Option = Expression
    END
    MONITOR POINT: timelog
      Expression Value = t
      Option = Expression
    END
    MONITOR POINT: transommom
      Expression Value = -2*torque_y()@transom
      Option = Expression
    END
```

```
      MONITOR POINT: transomx
        Expression Value = 2*force_x()@transom
        Option = Expression
      END
      MONITOR POINT: transomz
        Expression Value = 2*force_z()@transom
        Option = Expression
      END
      MONITOR RESIDUALS:
        Option = Full
      END
      MONITOR TOTALS:
        Option = Full
      END
    END
    RESULTS:
      File Compression Level = Default
      Option = Standard
    END
    TRANSIENT RESULTS: Transient Results 1
      File Compression Level = Default
      Include Mesh = No
      Option = Selected Variables
      Output Variables List = Pressure,Velocity,\
      Water.Volume Fraction
      OUTPUT FREQUENCY:
        Option = Time Interval
        Time Interval = tstep*5
      END
    END
  END
  SOLVER CONTROL:
    Turbulence Numerics = First Order
    ADVECTION SCHEME:
      Option = High Resolution
    END
    CONVERGENCE CONTROL:
      Maximum Number of Coefficient Loops = 5
      Minimum Number of Coefficient Loops = 3
      Timescale Control = Coefficient Loops
    END
    CONVERGENCE CRITERIA:
      Residual Target = 1.E-2
      Residual Type = MAX
    END
    JUNCTION BOX ROUTINES:
      Junction Box Routine List = initvalrout
    END
    MULTIPHASE CONTROL:
      Volume Fraction Coupling = Coupled
    END
    PRESSURE LEVEL INFORMATION:
      Cartesian Coordinates = 0 [m], 0 [m], 2 [m]
      Option = Cartesian Coordinates
      Pressure Level = 0 [atm]
    END
    TRANSIENT SCHEME:
      Option = Second Order Backward Euler
      TIMESTEP INITIALISATION:
        Option = Automatic
      END
    END
  END
  EXPERT PARAMETERS:
    backup file at zero = t
    topology estimate factor = 1.2
  END
END
COMMAND FILE:
  Results Version = 11.0
  Version = 12.1
END
```

This page intentionally left blank.

Annex D: CFX User Fortran: initval

```
C========1=========2=========3=========4=========5=========6=========7
C CFX-Fortran Routine for ANSYS CFX-11. This file must be compiled
C and linked before function(s) can be called by the solver.
C Eric Thornhill, February, 2006
C
C This #include is needed to allow access to CFX functions which can
C look-at or modify solver variables/parameters. See the CFX manual
C for User Fortran for more information.
      #include "cfx5ext.h"
C
C==========BEGINNING OF ROUTINE DEFINITION============================
C dllexport is the CFX function to generate the appropriate dll used
C by the solver.
      dllexport(initval)
      SUBROUTINE INITVAL(CZ,DZ,IZ,LZ,RZ)

C User routine: this function is called at the beginning of a solver
C run to intialize various variables. This particular initval function
C is meant for simulations involving irregular waves. The function
C will look for a text file 'irregwave.wif' in the run directory of
C the simulation. This file will contain the amplitude,frequency,phase
C information needed to recontruct the irregular wave using linear
C wave theory. The file will have n rows corresponding to n linear
C wave components to be combined to form the irregular wave. Each row
C will have three numbers seperated by white space (or tabs):
C amplitude (m), frequency (rad/s), and phase (rad). The file will be
C read at the beginning of the simulation and the data stored in
C \USER_DATA memory, to be accessed by the "getirregwave" function
C which is used to recontruct the wave at return data such as wave
C height and orbital velocities.

C ------------------------------
C     Preprocessor includes
C ------------------------------
      #include "stack_point.h"
      #include "MMS.h"

C ------------------------------
C        Argument list
C ------------------------------
      INTEGER I,IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)

C ------------------------------
C        Local Parameters
C ------------------------------
      CHARACTER*(*) ROUTIN
      PARAMETER (ROUTIN='INITVAL')
      CHARACTER*(4) CRESLT

C ------------------------------
C        Local Variables
C ------------------------------
      REAL INIVAL, WAVEINI
      CHARACTER  wavefile*16
      INTEGER I,WIUNIT, IOstatus
      REAL wcnt,w,A,p

```

```
C ------------------------------
C        Stack pointers
C ------------------------------
      __stack_point__ pOMEGA
      __stack_point__ pAMPLITUDE
      __stack_point__ pPHASE
      __stack_point__ pWAVENUM

C-----------------------------------------------------------------------
C    Executable Statements
C-----------------------------------------------------------------------
C Send any diagnostic messages via master process.
      CALL MESAGE('WRITE-ASIS', '    ')
      CALL MESAGE('WRITE','Initializing Junction Box Routine!')

C Set initial value
      INIVAL = 0

C Ensure that directory /USER_DATA exists and then make it the
C current directory.
      CALL PSHDIR('/USER_DATA', 'STOP', CRESLT)

C Create space for array ARRAY for irregular wave data
      CALL MAKDAT('NW', 'REAL', 'STOP', 1, pVAR1, CRESLT)
      CALL MAKDAT('OMEGA', 'REAL', 'STOP', 200, pARRAY, CRESLT)
      CALL MAKDAT('AMPLITUDE', 'REAL', 'STOP', 200, pARRAY, CRESLT)
      CALL MAKDAT('PHASE', 'REAL', 'STOP', 200, pARRAY, CRESLT)
      CALL MAKDAT('WAVENUM', 'REAL', 'STOP', 200, pARRAY, CRESLT)

C Initilize wave parameter data to -1, so a check can be done to
C see if it loaded properly. Using a max of 200 wave components.
      CALL POKER('NW', 1, INIVAL, 'STOP', CRESLT, RZ)
      DO I = 1,200
        WAVEINI = -1;
        CALL POKER ('OMEGA', I, WAVEINI, 'STOP', CRESLT, RZ)
        CALL POKER ('AMPLITUDE', I, WAVEINI, 'STOP', CRESLT, RZ)
        CALL POKER ('PHASE', I, WAVEINI, 'STOP', CRESLT, RZ)
        CALL POKER ('WAVENUM', I, WAVEINI, 'STOP', CRESLT, RZ)
      END DO

C========1=========2=========3=========4=========5=========6=========7
C See if irregwave.wif exits
      wavefile = '../irregwave.wif'

      CALL MESAGE( 'WRITE-ASIS', 'Reading irregular wave data from:'
     +        wavefile)

      WIUNIT = 5
      OPEN(unit=WIUNIT, file=wavefile, status='OLD')

C Initialize wcnt, the wave component counter
      wcnt = 0

C Read data from file to a maximum of 200 components, and store in
C \USER_DATA
      DO I = 1,200
        READ(WIUNIT,*,IOSTAT=IOstatus),w,A,p
        IF ((IOstatus .GT. 0) .OR. (IOstatus .LT. 0)) THEN
          EXIT
        ENDIF
        wcnt = wcnt+1;
        CALL POKER ('OMEGA', I, w, 'STOP', CRESLT, RZ)
        CALL POKER ('AMPLITUDE', I, A, 'STOP', CRESLT, RZ)
        CALL POKER ('PHASE', I, p, 'STOP', CRESLT, RZ)
      END DO

      CLOSE(WIUNIT)

      CALL POKER('NW', 1, wcnt, 'STOP', CRESLT, RZ)
      CALL USER_PRINT_REAL('Wave components read (max=200):',
     +      wcnt*1.0)
      CALL MESAGE( 'WRITE-ASIS', '    ')

      CALL POPDIR('STOP', CRESLT)

      END
C========1=========2=========3=========4=========5=========6=========7
```

This page intentionally left blank.

Annex E: CFX User Fortran: getirregwave

```
C========1=========2=========3=========4=========5=========6=========7
C CFX-Fortran Routine for ANSYS CFX-11. This file must be compiled
C and linked before function(s) can be called by the solver.
C Eric Thornhill, February, 2006
C
C This #include is needed to allow access to CFX functions which can
C look-at or modify solver variables/parameters.
      #include "cfx5ext.h"
C
C==========BEGINNING OF ROUTINE DEFINITION===========================
C
C dllexport is the CFX function to generate the appropriate dll used
C by the solver.
      dllexport(getirregwave)
      SUBROUTINE getirregwave(
     &  NLOC, NRET, NARG, RET, ARGS, CRESLT, CZ,DZ,IZ,LZ,RZ )
C
C User routine: this function returns either wave elevation,
C u velocity or v velocity for an irregular wave translating in the
C positive x-direction. The irregular wave is generated by summing a
C series of linear wave components with amplitudes, omegas, and phases
C as listed in the file 'irregwave.wif' that was read by the
C initialization junction box routine 'initval' (also user-fortran).
C The data is then stored in 'USER_DATA' memory.
C
C --------------------
C        Input
C --------------------
C  NLOC   - size of current locale
C  NRET   - number of components in result
C  NARG   - number of arguments in call
C  ARGS() - (NLOC,NARG) argument values
C
C
C --------------------
C        Output
C --------------------
C  RET()  - (NLOC,NRET) return values
C  CRESLT - 'GOOD' for success
C
C ------------------------------
C     Preprocessor includes
C ------------------------------
      #include "MMS.h"
      #include "stack_point.h"
C
C ------------------------------
C        Argument list
C ------------------------------
      INTEGER NLOC,NARG,NRET
      CHARACTER CRESLT*(*)
      REAL ARGS(NLOC,NARG), RET(NLOC,NRET)
      INTEGER IZ(*)
      CHARACTER CZ(*)*(1)
      DOUBLE PRECISION DZ(*)
      LOGICAL LZ(*)
      REAL RZ(*)
C
C ------------------------------
C        Local Variables
C ------------------------------
```

```
      INTEGER I,ILOC
      REAL param, t, z, x, nw, waterlevelz, scale
      REAL bottomz, boatvel
      REAL g, A, k, w, h, xd, eta, u, v, out
      REAL etol, cnt, k1, f1, k2, f2, k3
C
C----------------------------------------------------------------------
C    Executable Statements
C----------------------------------------------------------------------
C
C     Get data from argument inputs
      DO ILOC = 1,NLOC
        param =       ARGS(ILOC,1)
        t =           ARGS(ILOC,2)
        z =           ARGS(ILOC,3)
        x =           ARGS(ILOC,4)
        waterlevelz = ARGS(ILOC,5)
        bottomz =     ARGS(ILOC,6)
        boatvel =     ARGS(ILOC,7)
        scale =       ARGS(ILOC,8)
      END DO
C
C param: type of data to return. 1 = wave elevation, 2 = u velocity,
C         3 = v velocity
C t: current simulation time [s]
C z: z location of data [m]
C x: x location of data [m]
C waterlevelz: z coordinate of still water level [m]
C bottomz: z coordinate of domain bottom [m]
C boatvel: if simulation of moving ship, this is the ship
C           speed [m/s]
C scale: put in if wave data is at full scale, and simulation at
C         model scale. e.g. if model is 1/10 scale, and wave data is
C         full scale, user scale=10
C         NOTE: This is not currently active in this function
C
C Helpful Formulas:
C ---------------------------------------------------------------
C k = 2*pi/lambda     k=wavenumber, lambda=wavelength
C w = 2*pi/T          w (omega)= rad frequency , T= wave period
C h = 2*A             h=wave height, A=wave amplitude
C w^2=g*k*tanh(k*h)   dispersion relation
C w^2=g*k             dispersion relation (deep water)
C eta = A*cos(k*x-w*t+p)   wave elevation equation p = phase
C pressure = rho*g*A*cosh(ky+kh)/cosh(kh)*cos(kx-wt)
C
C Wave equations can vary place to place. This function using:
C phi = -Ag/w * cosh(kz-kh)/cosh(kh) *  sin(wt-kx)
C
C because also using the condition eta = -1/g * d(phi)/dt
C it gives: eta=A*cos(wt-kx) which is what precal/pretti uses (?).
C
C So, using u = d(phi)/dx  and v = d(phi)/dz
C
C You get:  u = Agk/w * cosh(kz+kh)/cosh(kh) * cos(wt-kx)
C           v = -Agk/w * sinh(kz+kh)/cosh(kh) * sin(wt-kx)
C
C Using the deep water assumptions, this simplifies to:
C           u = exp(k*z) * Aw * cos(wt-kx)
C           v = -exp(k*x) * Aw * sin(wt-kx)

C========1=========2=========3=========4=========5=========6=========7
C Ensure that directory /USER_DATA exists and then make it the current
C directory.
```

```
      CALL PSHDIR('/USER_DATA','STOP',CRESLT)

C Get nw: number of wave components in 'irregwave.wif'
      CALL PEEKR('NW', 1, nw, 'STOP', CRESLT, RZ)

      IF (nw .LE. 0.0) THEN
        CALL MESAGE('WRITE-ASIS','getirregwave: nw=0')
      ENDIF

      g = 9.807
      pi = 3.1415926535897932
      z = z-waterlevelz
      h = waterlevelz-bottomz

      IF (h .LE. 0.0) THEN
        CALL MESAGE('WRITE-ASIS','getirregwave: h<=0')
      ENDIF

C========1=========2=========3=========4=========5=========6=========7
C Wave numbers, k, are not using the deep water assumption so their
C calculation requires an iterative process that need only be done
C once and then saved to USER_DATA. This section checks the array
C WAVENUM to see if the first number is still the initialized value
C of -1, if so, it will calculate the wave numbers for the given
C omegas and water depth.
      CALL PEEKR('WAVENUM', 1, k, 'STOP', CRESLT, RZ)

C Use scale to scale down irregular waves (if given in full scale).
      IF (k .LT. 0.0) THEN
        CALL MESAGE( 'WRITE-ASIS',
     +              'getirregwave: scaling wave parameters')
        DO I=1,nw
          CALL PEEKR('OMEGA', I, w, 'STOP', CRESLT, RZ)
          CALL PEEKR('AMPLITUDE', I, A, 'STOP', CRESLT, RZ)
          w = w*sqrt(scale)
          A = A/scale
          CALL POKER ('OMEGA', I, w, 'STOP', CRESLT, RZ)
          CALL POKER ('AMPLITUDE', I, A, 'STOP', CRESLT, RZ)
        ENDDO
      ENDIF

C Calculate wave numbers if not already calculated.
      IF (k .LT. 0.0) THEN
        CALL MESAGE( 'WRITE-ASIS',
     +              'getirregwave: calculating wave numbers')

C Calculate wave numbers using secant method
        DO I = 1,nw
          k = -1
          CALL PEEKR('OMEGA', I, w, 'STOP', CRESLT, RZ)
          etol = 1
          cnt = 0
          k1 = 0.01
          f1 = w**2-g*k1*tanh(k1*h)
          k2 = 1
          f2 = w**2-g*k2*tanh(k2*h)

          DO WHILE ((etol .GT. 0.0001) .AND. (cnt .LT. 100))
            cnt = cnt+1
            k3 = k2-(k2-k1)/(f2-f1)*f2
            etol = abs(f2-f1)
            k1 = k2
            f1 = f2
            k2 = k3
            f2 = w**2-g*k2*tanh(k2*h)
          ENDDO

          k=k2
          IF (cnt .GT. 98) THEN
            CALL MESAGE( 'WRITE-ASIS',
     +      'getirregwave: no secant k convergence,using deep water')
            k = w**2/g
          ENDIF

          CALL POKER ('WAVENUM', I, k, 'STOP', CRESLT, RZ)
          CALL USER_PRINT_REAL('k = ',k)
        ENDDO

      ENDIF

C========1=========2=========3=========4=========5=========6=========7
C  Calculate wave amplitude and velocities.
      eta = 0
      u = 0
      v = 0

      DO I=1,nw
        CALL PEEKR('OMEGA', I, w, 'STOP', CRESLT, RZ)
        CALL PEEKR('AMPLITUDE', I, A, 'STOP', CRESLT, RZ)
        CALL PEEKR('PHASE', I, p, 'STOP', CRESLT, RZ)
        CALL PEEKR('WAVENUM', I, k, 'STOP', CRESLT, RZ)

        IF (A .LE. 0.0) THEN
          CALL MESAGE( 'WRITE-ASIS','getirregwave: waveheight<=0')
        ENDIF

        IF (w .LE. 0.0) THEN
          CALL MESAGE( 'WRITE-ASIS','getirregwave: omega<=0')
        ENDIF

        IF (k .LE. 0.0) THEN
          CALL MESAGE( 'WRITE-ASIS','getirregwave: k<=0')
        ENDIF

        xd = -boatvel*t + x
        eta = eta+A*cos(w*t-k*(xd)+p)
        u = u + A*g*k/w * cosh(k*z+k*h)/cosh(k*h) * cos(w*t-k*xd+p)
        v = v - A*g*k/w * sinh(k*z+k*h)/cosh(k*h) * sin(w*t-k*xd+p)
      ENDDO

      IF (param .EQ. 1.0) THEN
        out=eta
      ELSE IF (param .EQ. 2.0) THEN
        out=u
      ELSE IF (param .EQ. 3.0) THEN
        out=v
      ELSE
        out=0
        CALL MESAGE( 'WRITE-ASIS','getirregwave: Unrecognized Param')
      ENDIF

C Switch back to previous directory
      CALL POPDIR('STOP', CRESLT)

C Return delta for new delta position
      DO ILOC = 1,NLOC
        RET(ILOC,1) = out
      END DO
```

```

C Set success flag and end function
      CRESLT = 'GOOD'

      END
C========1=========2=========3=========4=========5=========6=========7
```

This page intentionally left blank.

Annex F: Pressure Tap Locations

The locations of pressure transducers on the ELAST ferry model are given below in Table F.1. Positions are given in full scale meters relative to the forward perpendicular (x-coordinate), centreline (y-coordinate), and keel (z-coordinate).

Table F.1: *Pressure gauge locations*

Guage #	x [m]	y [m]	z [m]	nx [-]	ny [-]	nz [-]	Delauney Area [m2]
1	-5.770	2.527	8.386	0.2238	-0.7054	-0.6726	2.382
2	-5.770	4.454	10.186	0.2417	-0.6431	-0.7266	4.949
3	-5.770	8.369	13.786	0.2502	-0.7065	-0.6621	3.701
4	-3.070	1.662	8.386	0.2497	-0.7376	-0.6274	3.978
5	-3.070	3.397	10.186	0.2692	-0.6563	-0.7049	6.071
6	-3.070	5.397	11.986	0.2755	-0.6396	-0.7176	9.977
7	-3.070	7.342	13.786	0.2728	-0.6786	-0.6820	2.474
8	-0.370	2.211	10.186	0.3247	-0.6688	-0.6688	7.487
9	-0.370	4.151	11.986	0.3129	-0.6276	-0.7129	6.470
10	-0.370	6.170	13.786	0.3050	-0.6366	-0.7083	6.432
11	2.330	0.444	10.186	0.6214	-0.3883	-0.6805	4.182
12	2.330	2.620	11.986	0.3943	-0.5890	-0.7055	7.374
13	2.330	4.780	13.786	0.3439	-0.6129	-0.7114	8.332
14	2.330	6.794	15.586	0.3227	-0.6557	-0.6826	2.582
15	5.030	2.879	13.786	0.4566	-0.4948	-0.7393	5.621
16	5.030	5.285	15.586	0.3687	-0.6089	-0.7024	5.845
17	7.730	2.987	15.586	0.5023	-0.3924	-0.7705	3.048
18	1.240	0.0	8.912	0.7513	0.0000	-0.6600	2.685
19	3.120	0.0	10.815	0.6858	0.0000	-0.7278	2.040
20	5.115	0.0	12.602	0.6534	0.0000	-0.7570	5.644
21	7.180	0.0	14.311	0.6225	0.0000	-0.7826	5.828
22	9.310	0.0	15.937	0.5934	0.0000	-0.8049	1.368
23	2.330	-2.620	11.986	0.3943	0.5890	-0.7055	7.374

The following are the locations of the pressure monitors used for the ELAST ferry CFD simulations in the coordinates of the simulation. They were set to match the locations of the pressure gauges in the bow segment of the physical model. They are given in the ANSYS CCL format that can be imported/exported as plain text files by CFX. Note that the special character '#' indicates a comment line and the character '\' indicates the given line continues onto the next.

```
FLOW:
  OUTPUT CONTROL:
     MONITOR OBJECTS:
       MONITOR POINT:P01
         Cartesian Coordinates = 0.040139[m],0.070194[m],0.057944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P02
         Cartesian Coordinates = 0.040139[m],0.123722[m],0.107944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P03
         Cartesian Coordinates = 0.040139[m],0.232472[m],0.207944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P04
         Cartesian Coordinates = -0.034861[m],0.046167[m],0.057944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P05
         Cartesian Coordinates = -0.034861[m],0.094361[m],0.107944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P06
         Cartesian Coordinates = -0.034861[m],0.149917[m],0.157944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P07
         Cartesian Coordinates = -0.034861[m],0.203944[m],0.207944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
       END
       MONITOR POINT:P08
         Cartesian Coordinates = -0.109861[m],0.061417[m],0.107944[m]
         Option = Cartesian Coordinates
         Output Variables List = Pressure
```

```
END
MONITOR POINT:P09
  Cartesian Coordinates = -0.109861[m],0.115306[m],0.157944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P10
  Cartesian Coordinates = -0.109861[m],0.171389[m],0.207944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P11
  Cartesian Coordinates = -0.184861[m],0.012333[m],0.107944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P12
  Cartesian Coordinates = -0.184861[m],0.072778[m],0.157944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P13
  Cartesian Coordinates = -0.184861[m],0.132778[m],0.207944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P14
  Cartesian Coordinates = -0.184861[m],0.188722[m],0.257944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P15
  Cartesian Coordinates = -0.259861[m],0.079972[m],0.207944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P16
  Cartesian Coordinates = -0.259861[m],0.146806[m],0.257944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P17
  Cartesian Coordinates = -0.334861[m],0.082972[m],0.257944[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
MONITOR POINT:P18
  Cartesian Coordinates = -0.154583[m],0.000000[m],0.072556[m]
  Option = Cartesian Coordinates
  Output Variables List = Pressure
END
```

```
      MONITOR POINT:P19
        Cartesian Coordinates = -0.206806[m],0.000000[m],0.125417[m]
        Option = Cartesian Coordinates
        Output Variables List = Pressure
      END
      MONITOR POINT:P20
        Cartesian Coordinates = -0.262222[m],0.000000[m],0.175056[m]
        Option = Cartesian Coordinates
        Output Variables List = Pressure
      END
      MONITOR POINT:P21
        Cartesian Coordinates = -0.319583[m],0.000000[m],0.222528[m]
        Option = Cartesian Coordinates
        Output Variables List = Pressure
      END
      MONITOR POINT:P22
        Cartesian Coordinates = -0.378750[m],0.000000[m],0.267694[m]
        Option = Cartesian Coordinates
        Output Variables List = Pressure
      END
    END
  END
END
```

Annex G: Virtual Wave Probes

Virtual wave probes are a vertical column of points that monitor volume fraction in the CFD simulation. For the ELAST ferry simulation each virtual probe consisted of 53 points which spanned from 0.181 m below the still waterline to 0.181 m above the still waterline in equal increments of 0.007 m (-6.5 m to 6.5 m by 0.25 m full scale). Four virtual probes were used which corresponded to probes 1, 2, 10, and 16. All probes were located on the ships centreline (y=0). The x positions of the probes in simulation coordinates were: -2.65 m, -1.65 m, -0.95 m, -0.65 m.

After a run was completed, the first step in extracting the wave heights at the virtual wave probes was to check the actual monitor point positions compared with the assigned positions. When CFX begins a run, it automatically re-assigns monitor points to the nearest mesh nodes. These re-assignments are listed in the simulation output file as shown below for point 53 in virtual wave probe #16. Provided the position changes were small, there would be a negligible effect on the reported wave heights.

```
Monitor Point: W16Z53

  Domain: Domain 1
  User specified location (x,y,z) : -6.500E-01,  0.000E+00,  1.806E-01
  Assigned vertex location (x,y,z): -6.538E-01,  0.000E+00,  1.809E-01
  Distance to specified location  :  3.830E-03

  Valid variables from output variable list:
                       Water.Volume Fraction
```

Once the time series of volume fraction for each point of a virtual wave probe was read in the monitor output file[2]. A Matlab script was used to determine the z-coordinate were the volume fraction equalled 0.5. Volume fraction is a scalar field variable used by the VOF method for multiphase flows. Every cell is assigned a value ranging from 0 (cell is completely full of air) to 1 (cell is completely full of water). Volume fraction equal to 0.5 is used to define the air/water interface (free surface).

The Matlab script interpolates the volume fraction data to determine the z-coordinate of the free surface. This is done for each virtual wave probe at every time step. If the virtual wave probe determines multiple free surfaces, or no free surface within its z-coordinate range, it returns a null value.

[2]Monitor data can be exported to comma-delimited text files from the CFX Solver Manager GUI, or extracted from a results file at the command line by using the `cfx5dfile` utility.
e.g. `cfx5dfile filename.res -read-monitor > outputname`

This page intentionally left blank.

Annex H: Pressure Transducer Results

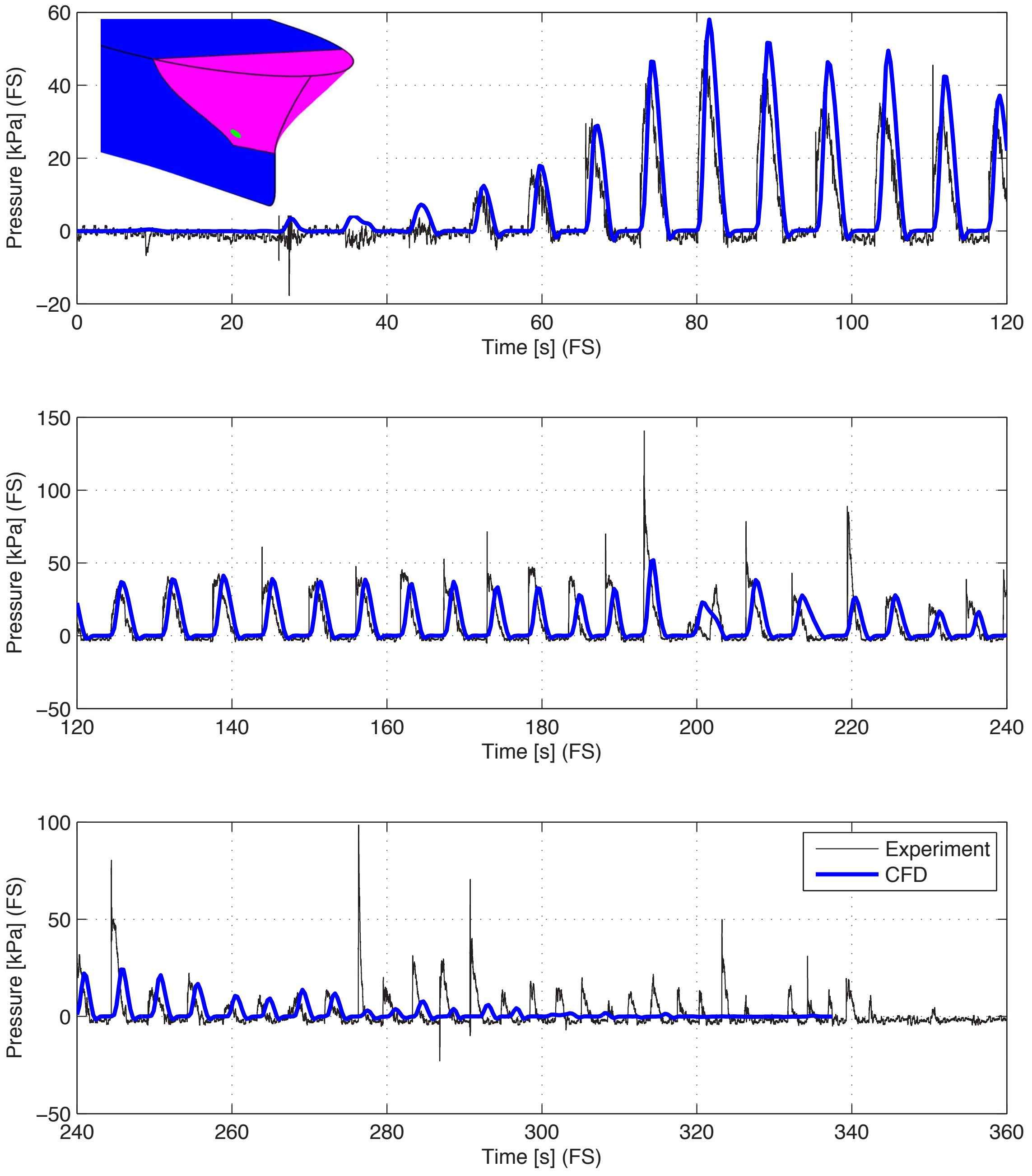

Figure H.1: *Pressure transducer #1 data*

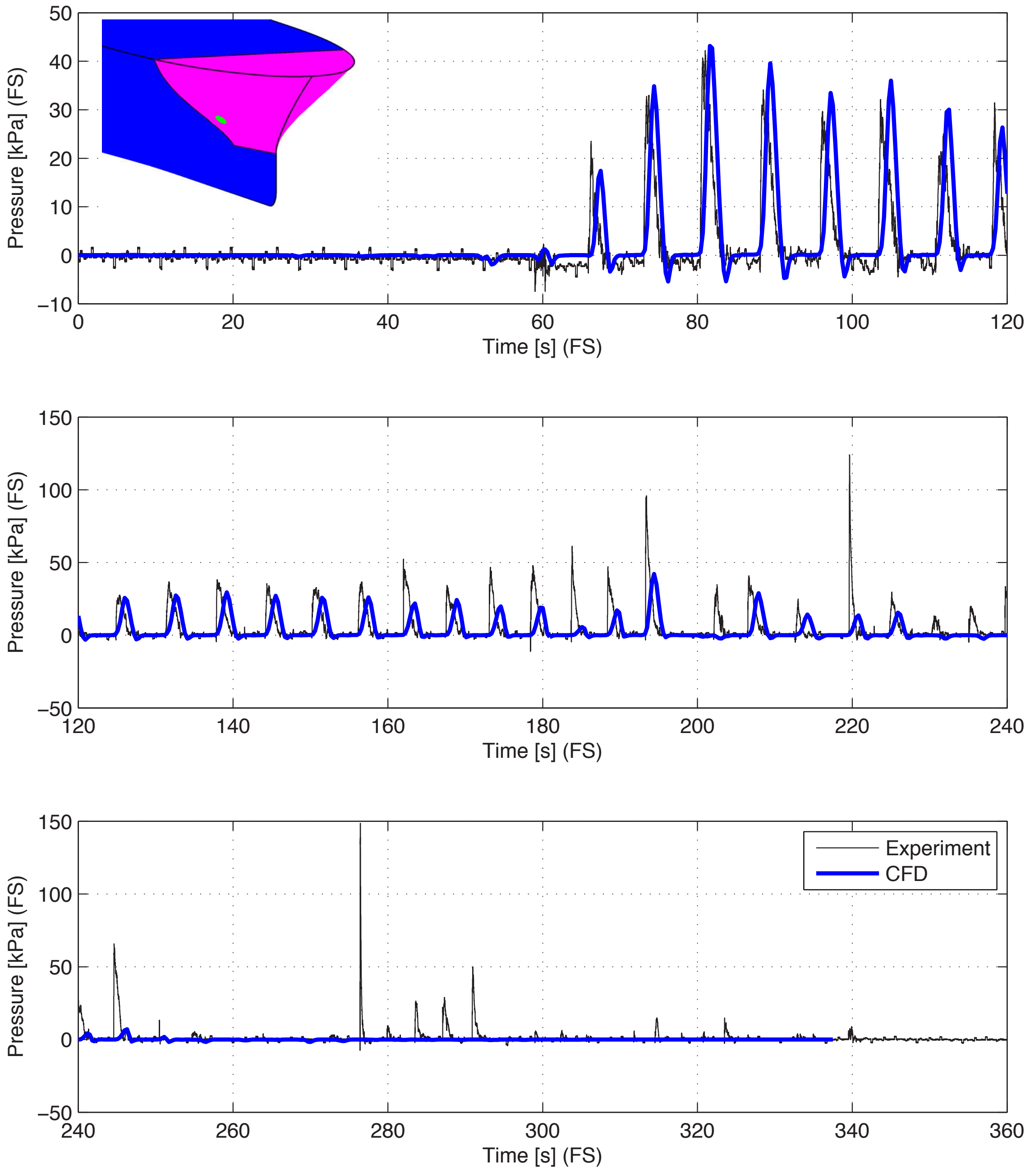

Figure H.2: *Pressure transducer #2 data*

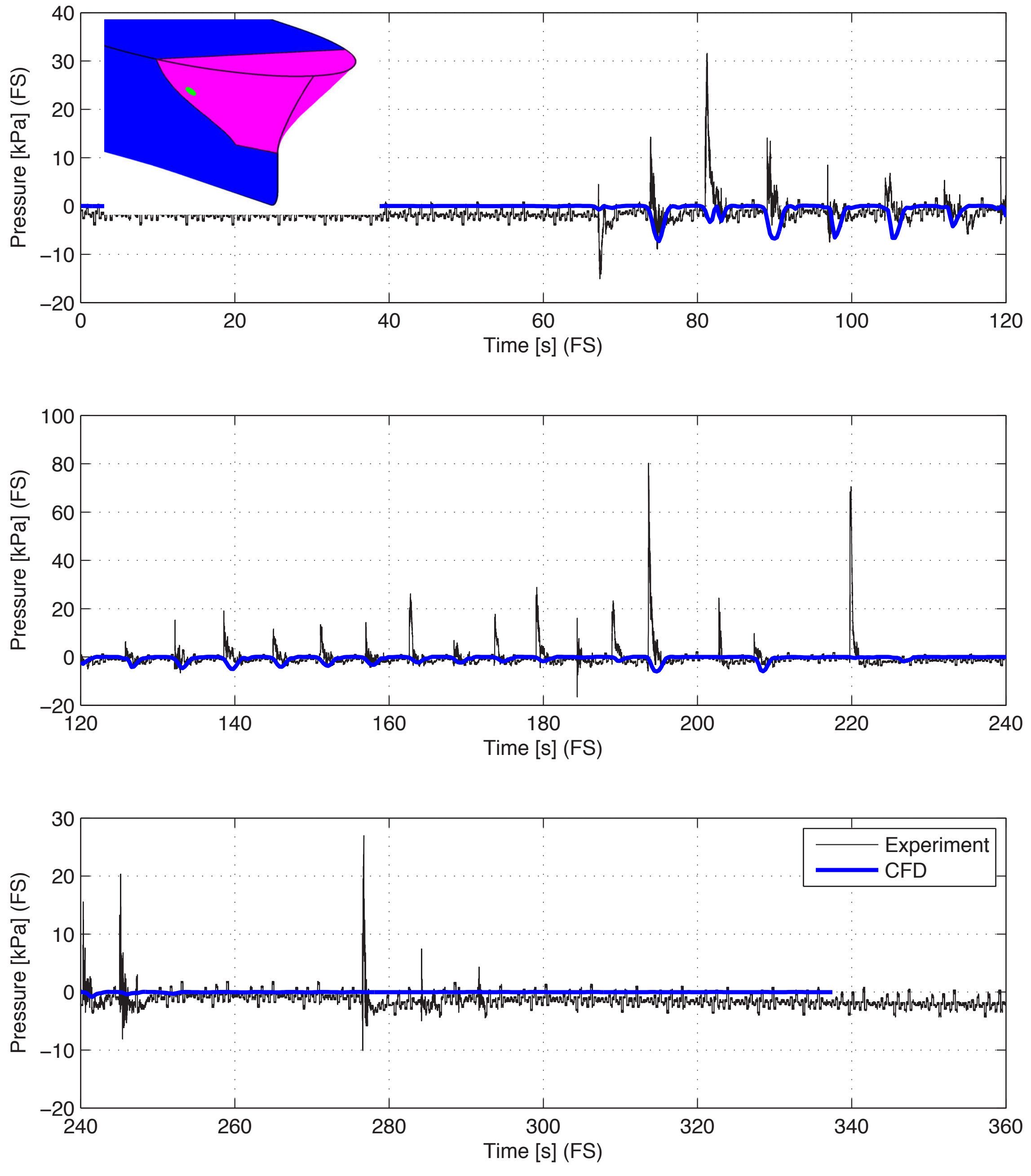

Figure H.3: *Pressure transducer #3 data*

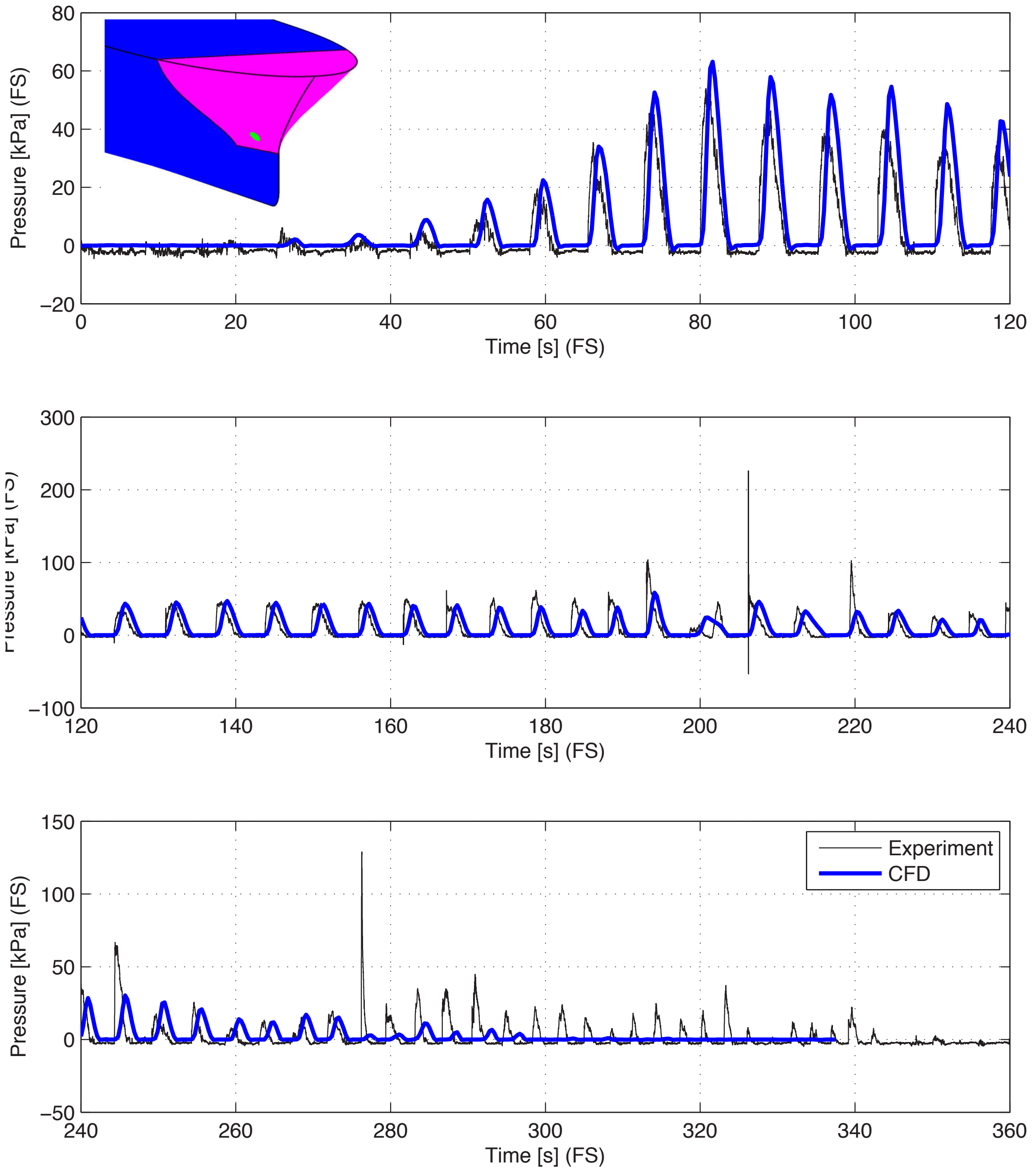

Figure H.4: *Pressure transducer #4 data*

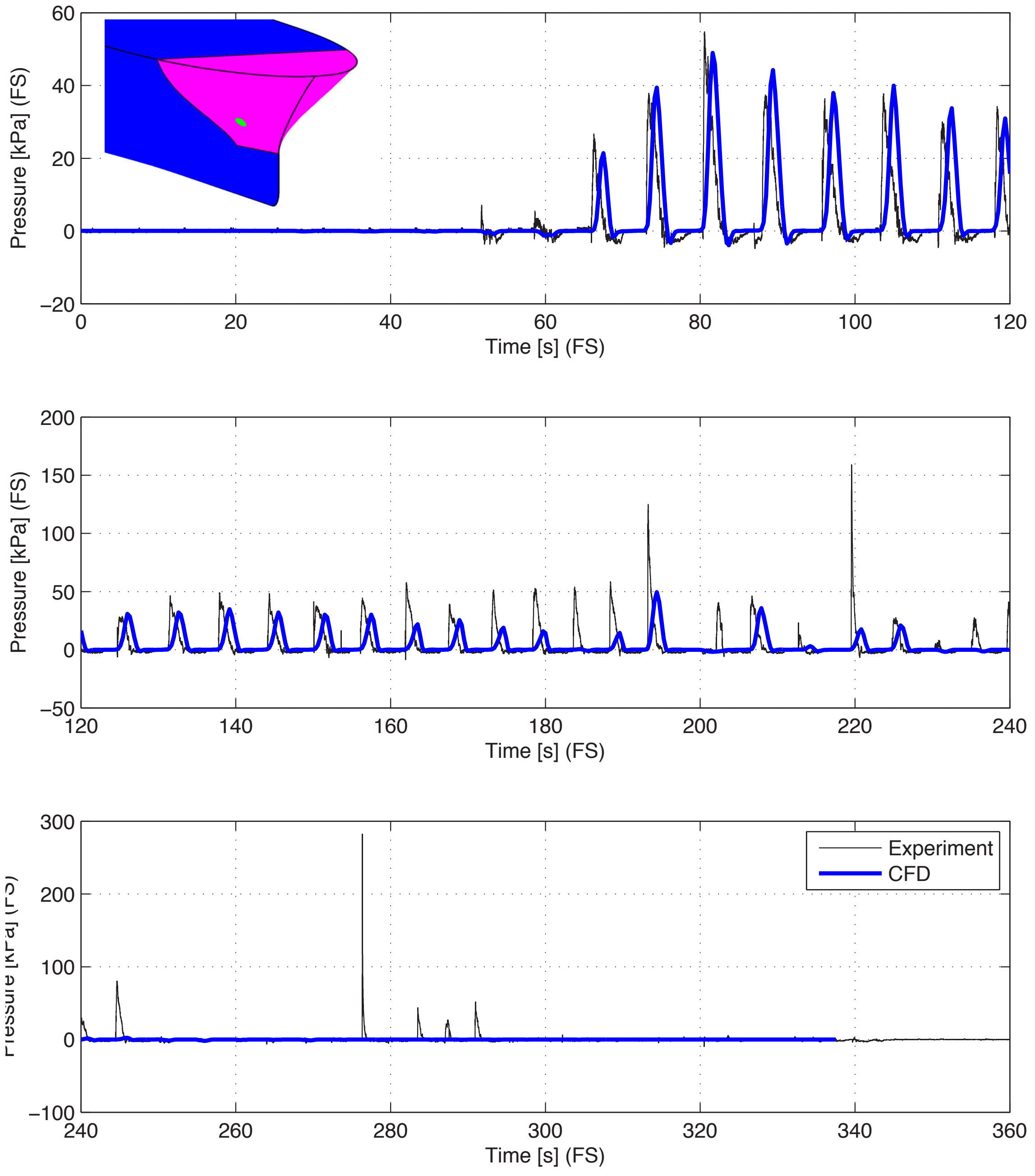

Figure H.5: *Pressure transducer #5 data*

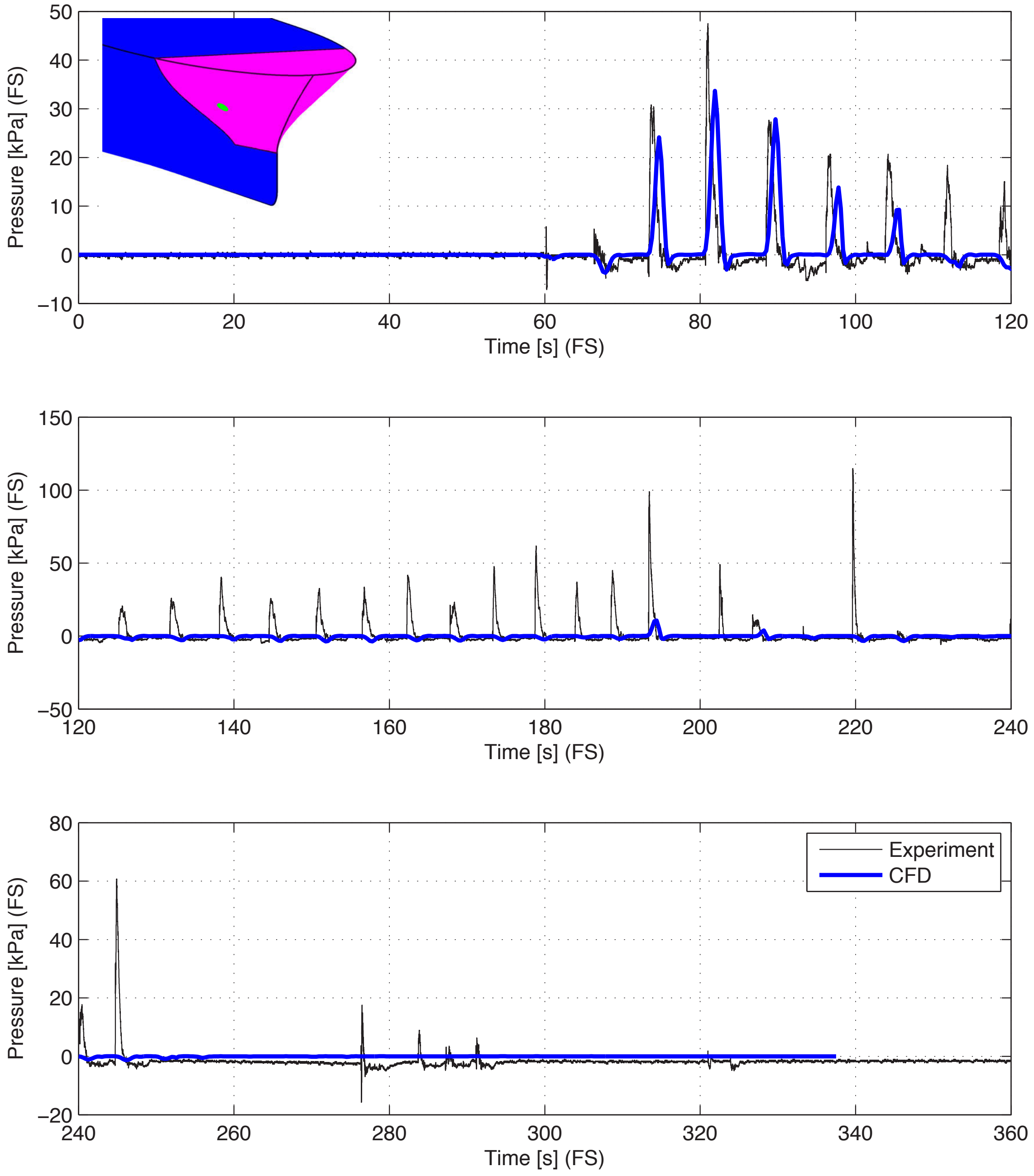

Figure H.6: *Pressure transducer #6 data*

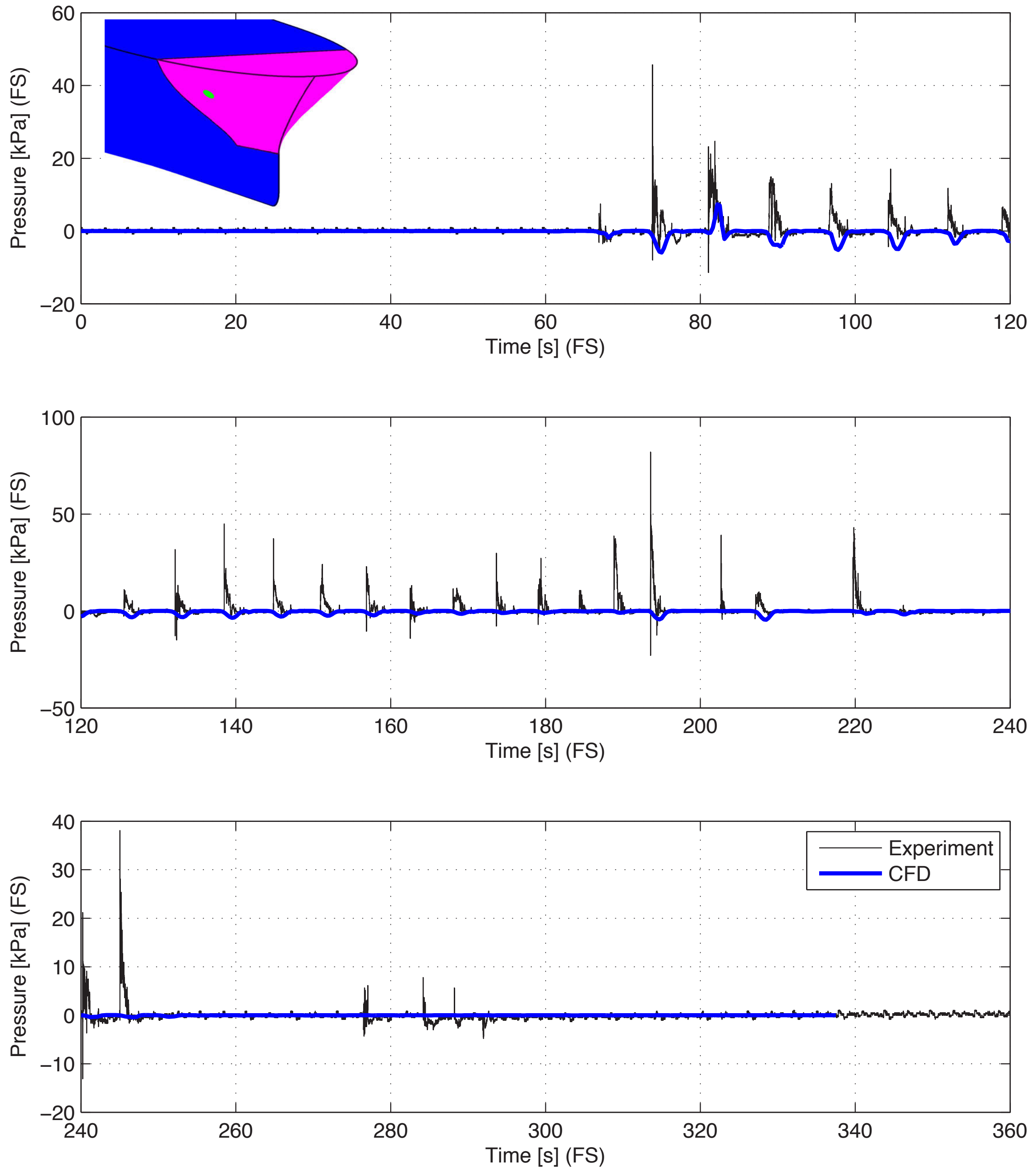

Figure H.7: *Pressure transducer #7 data*

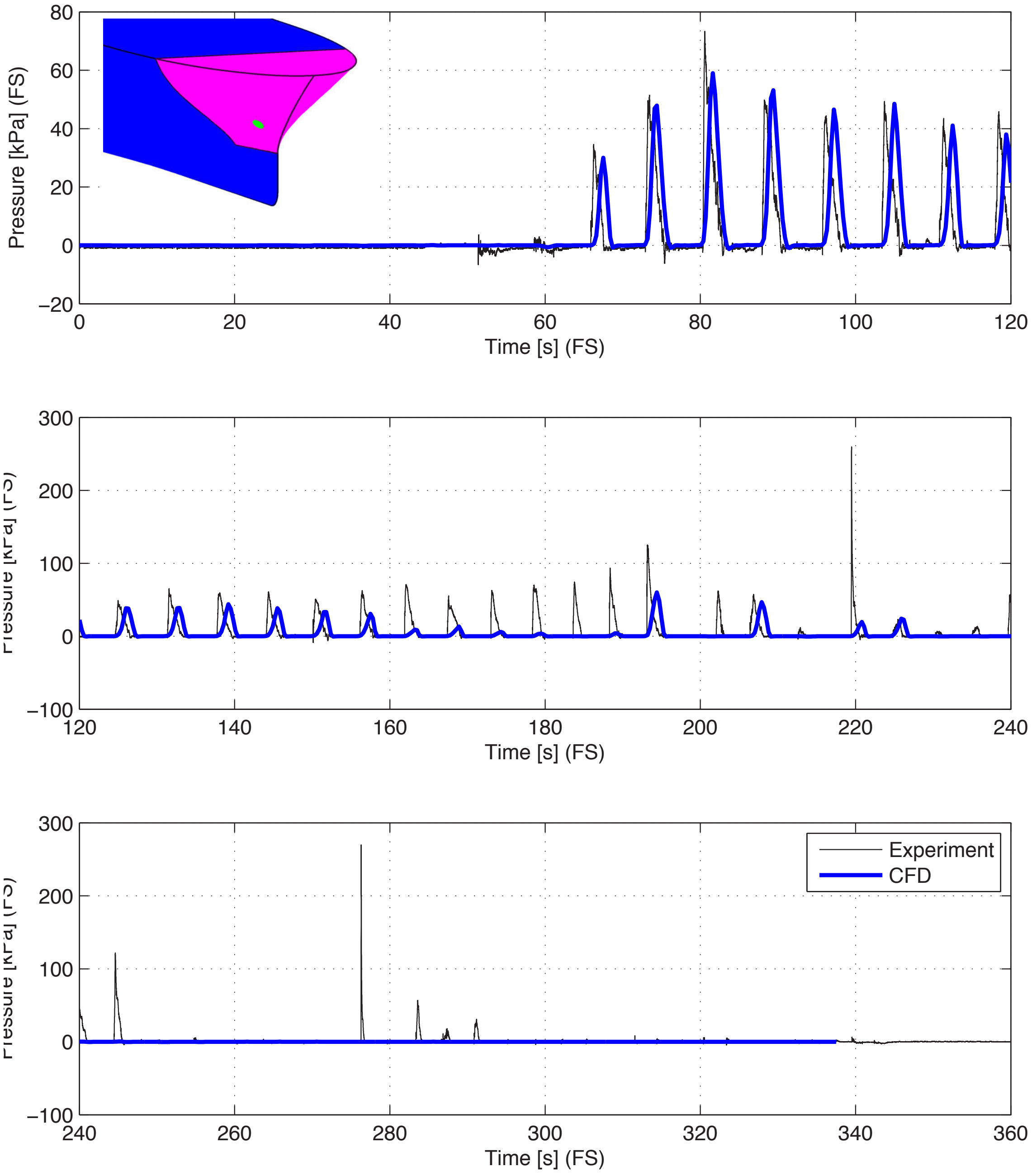

Figure H.8: *Pressure transducer #8 data*

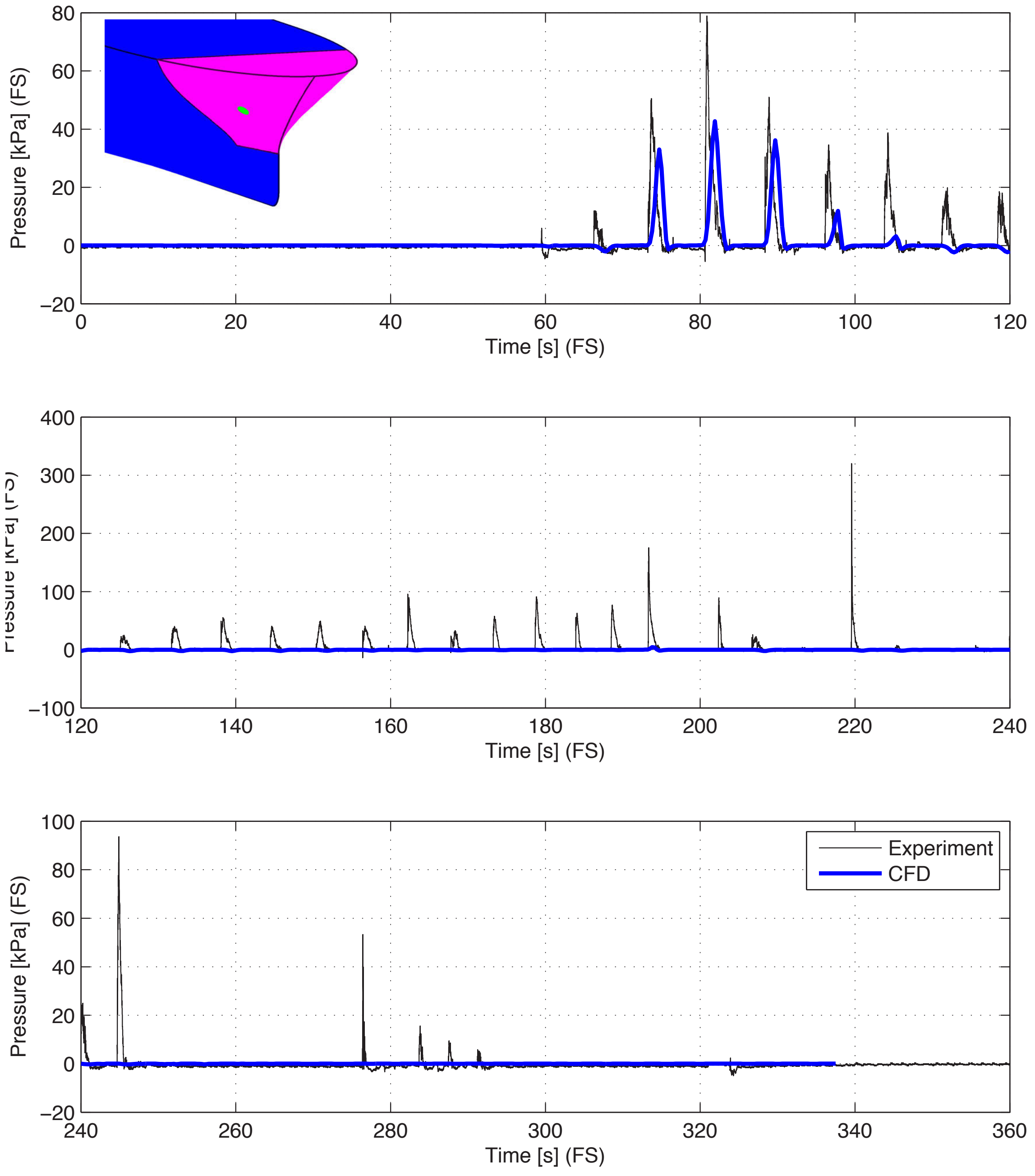

Figure H.9: *Pressure transducer #9 data*

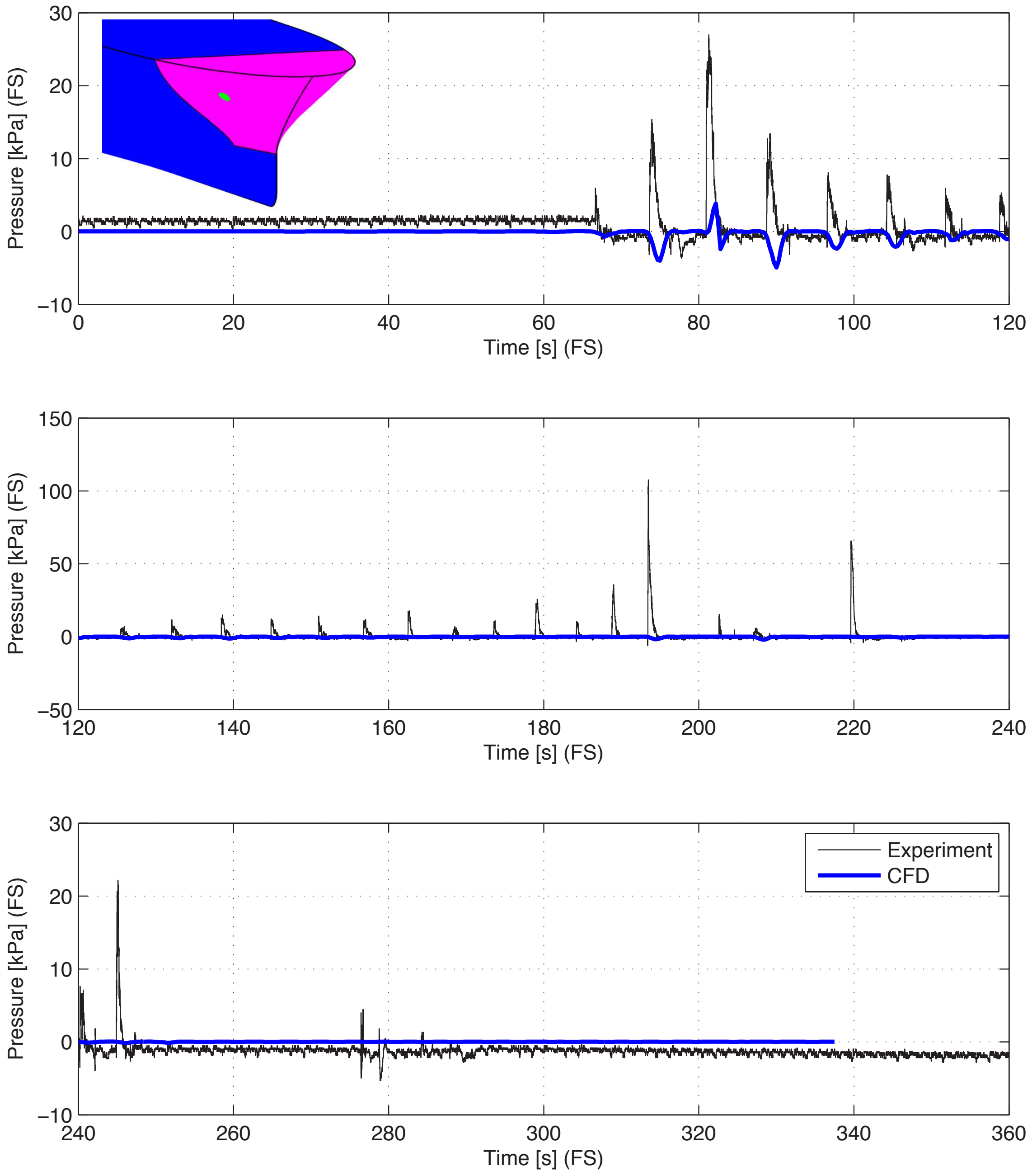

Figure H.10: *Pressure transducer #10 data*

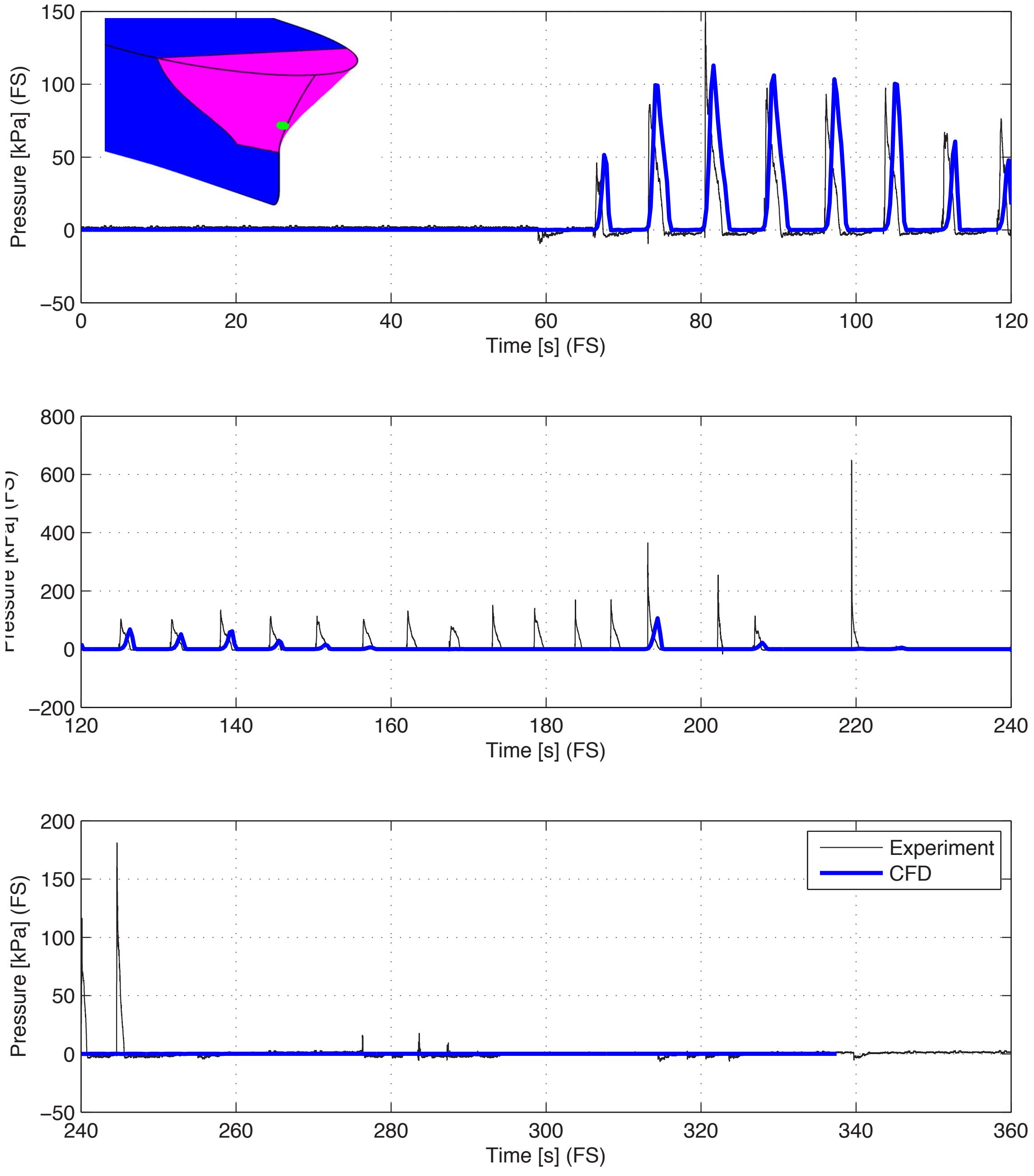

Figure H.11: *Pressure transducer #11 data*

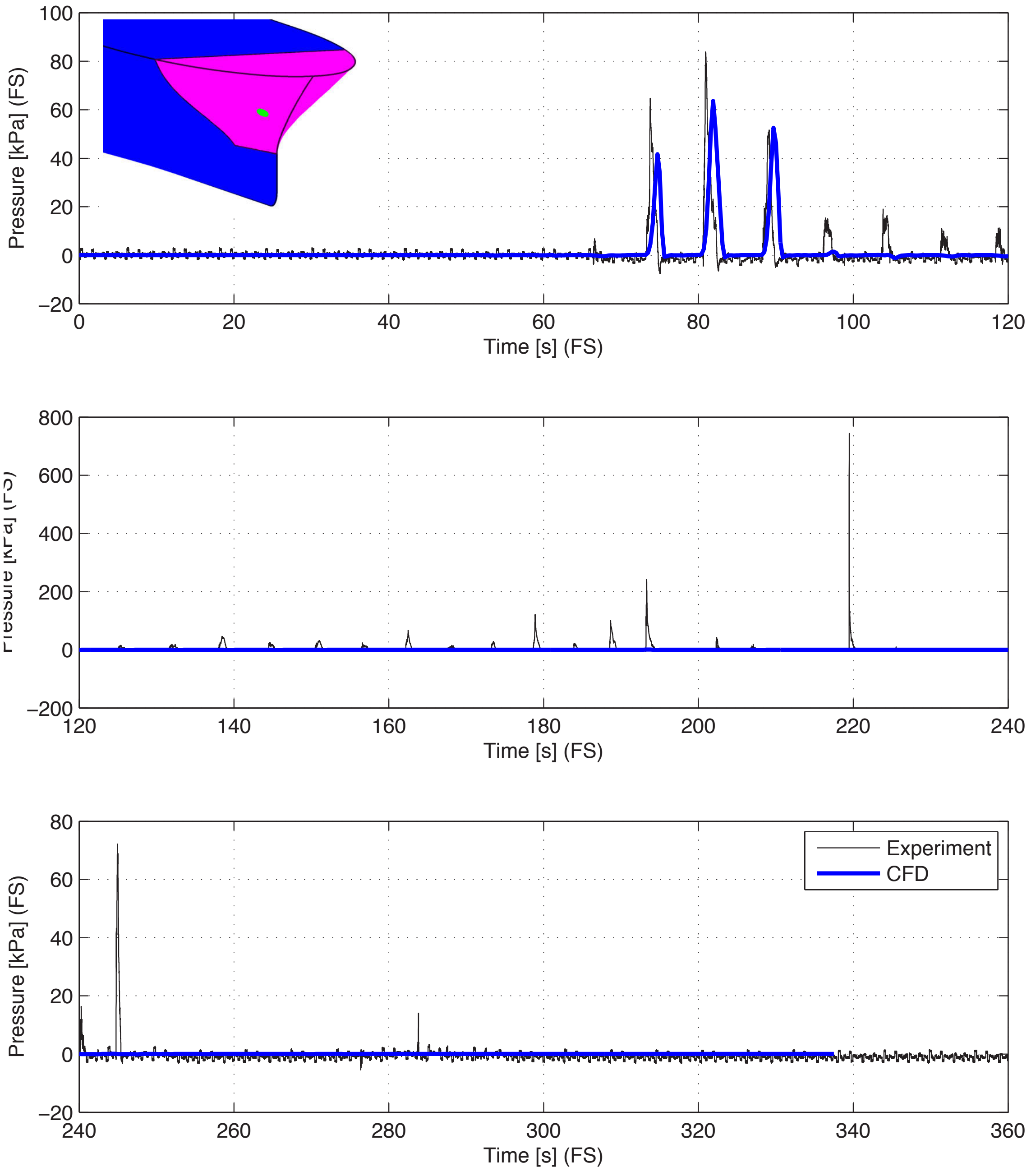

Figure H.12: *Pressure transducer #12 data*

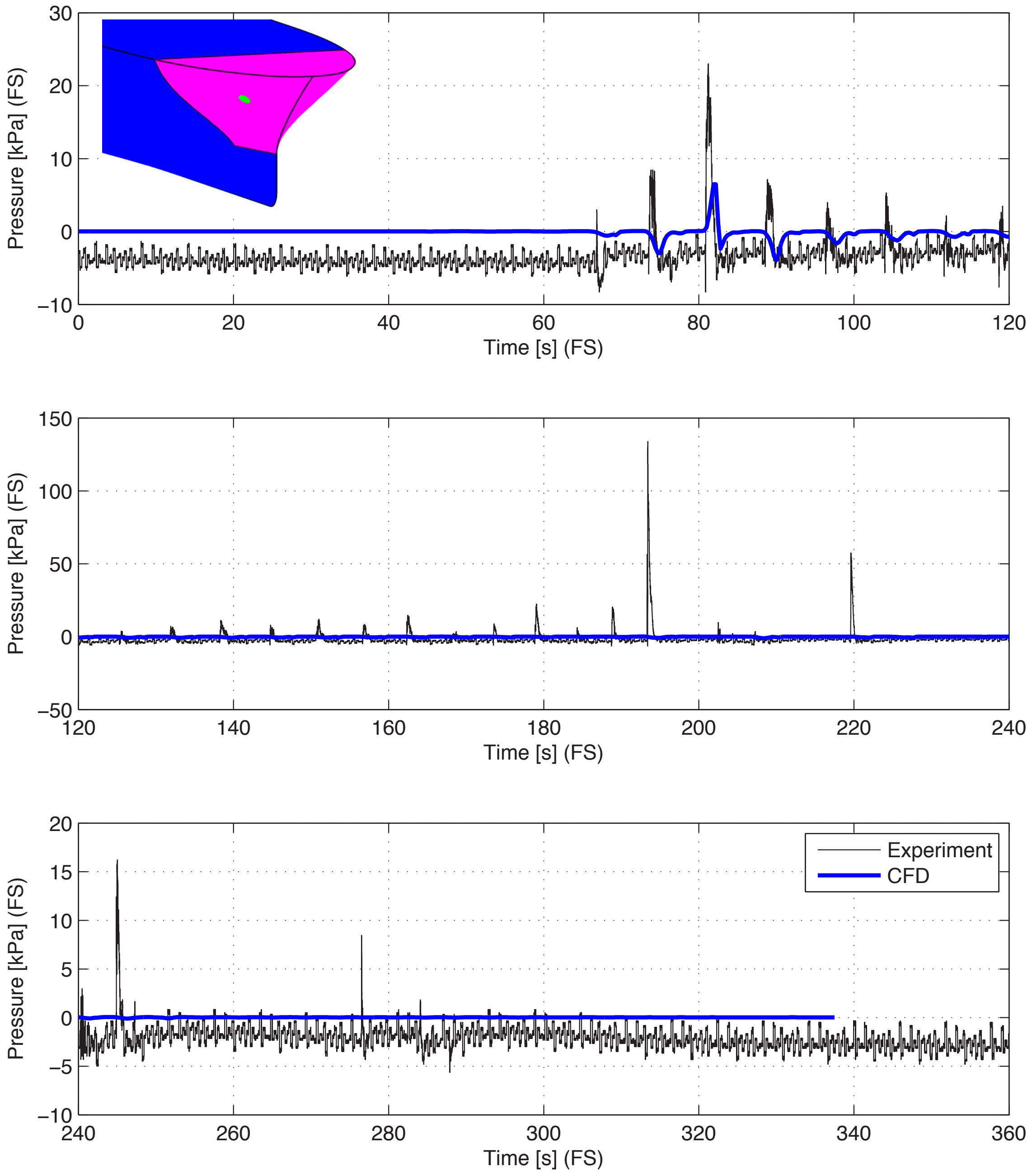

Figure H.13: *Pressure transducer #13 data*

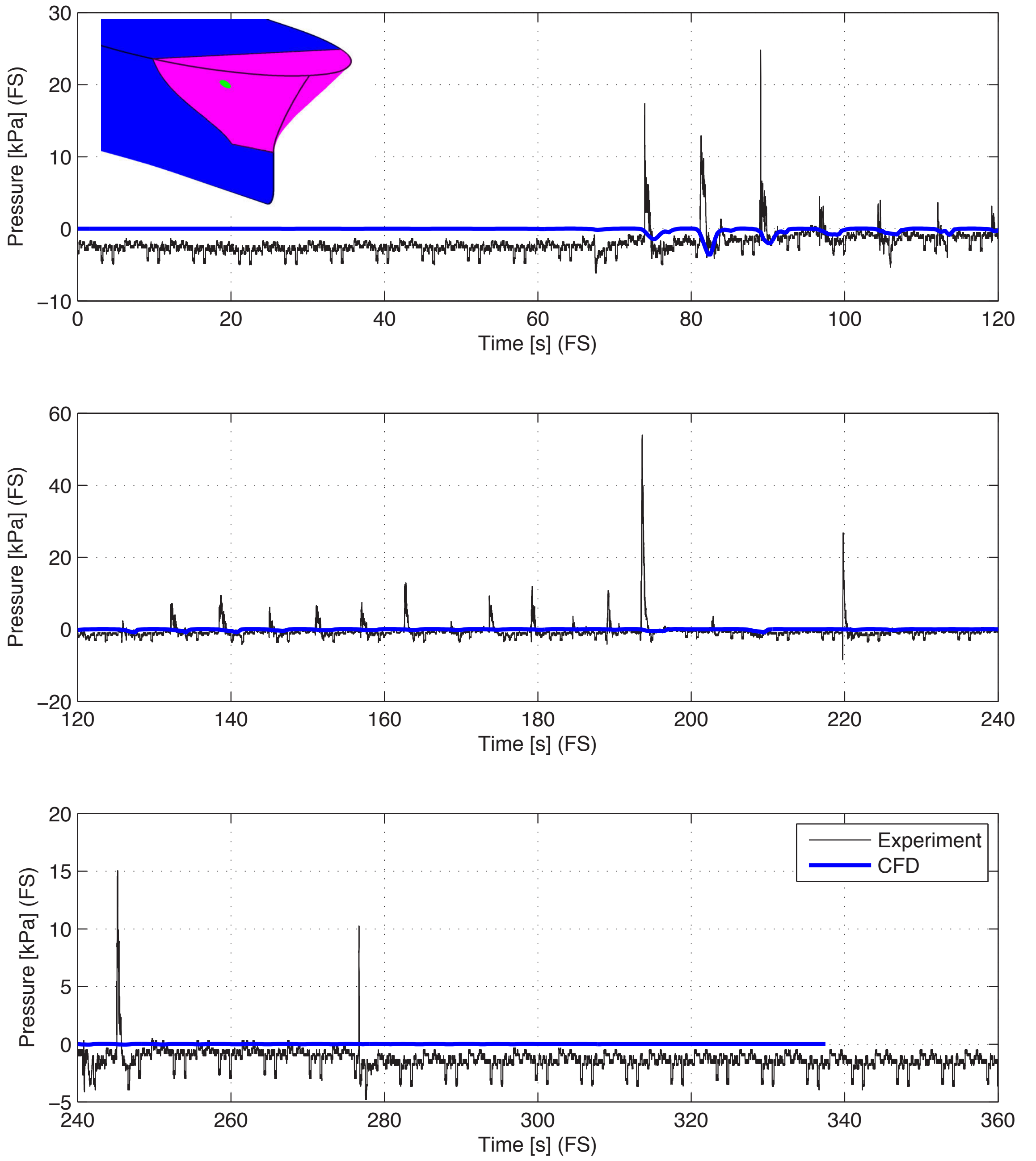

Figure H.14: *Pressure transducer #14 data*

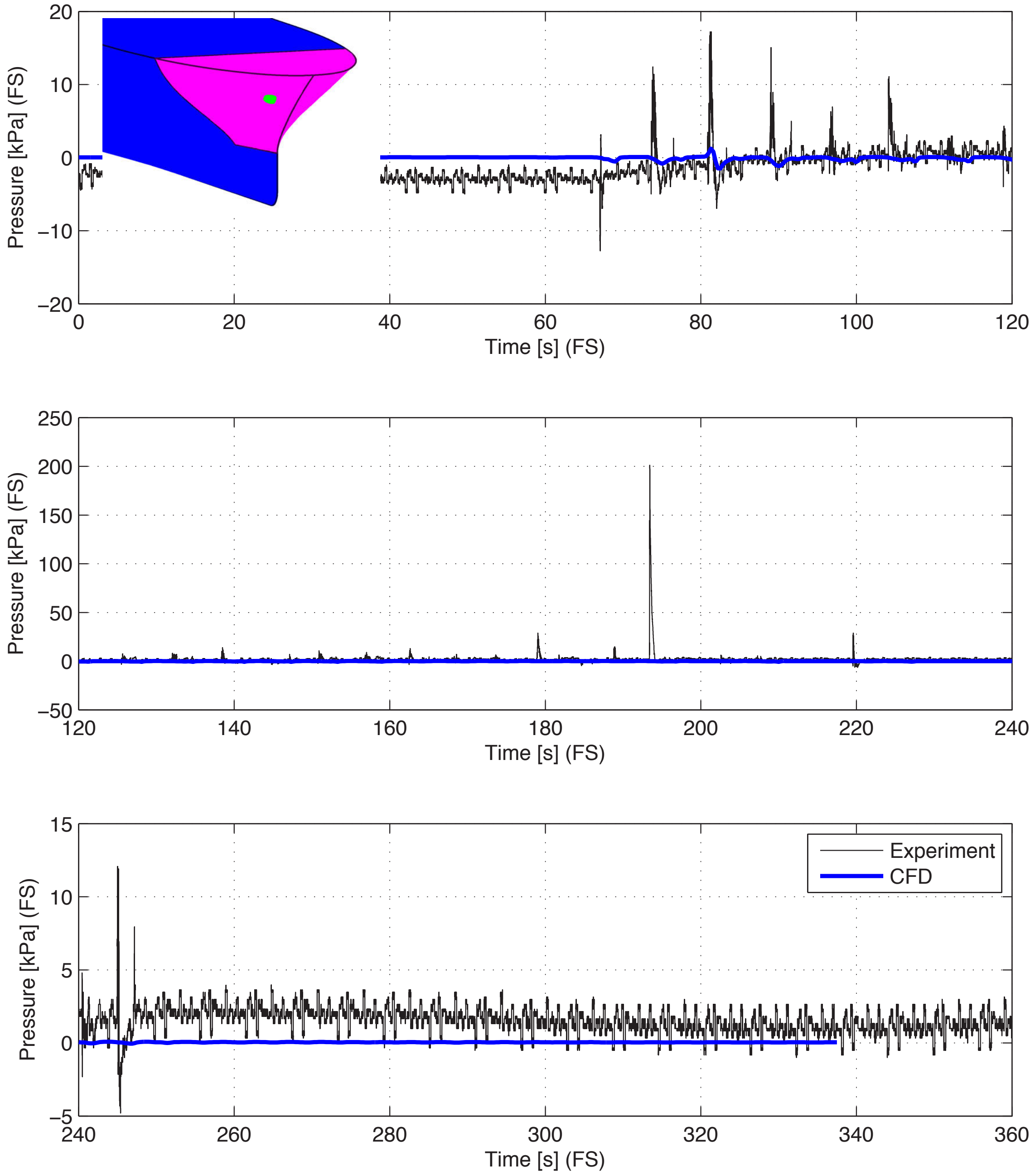

Figure H.15: *Pressure transducer #15 data*

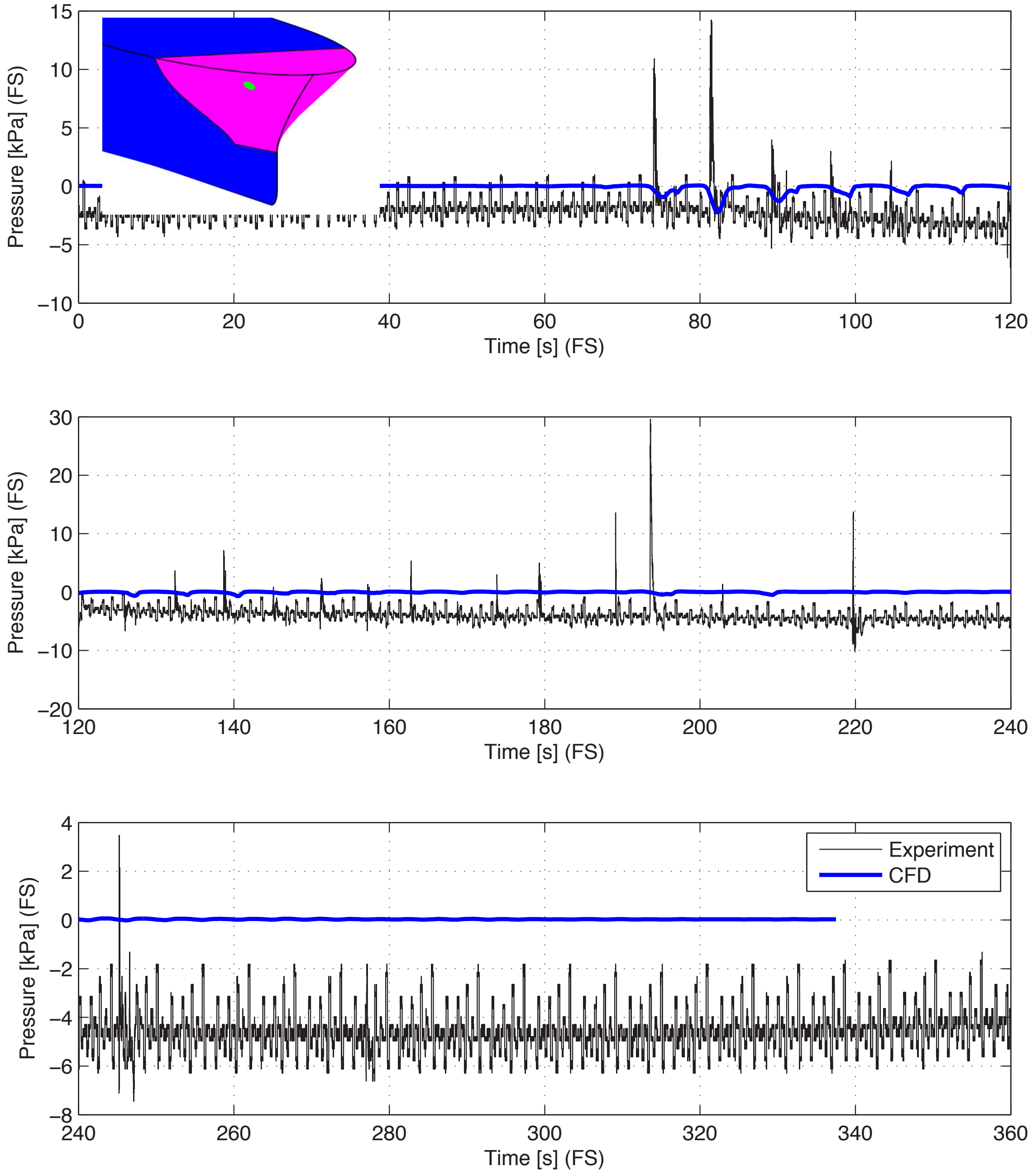

Figure H.16: *Pressure transducer #16 data*

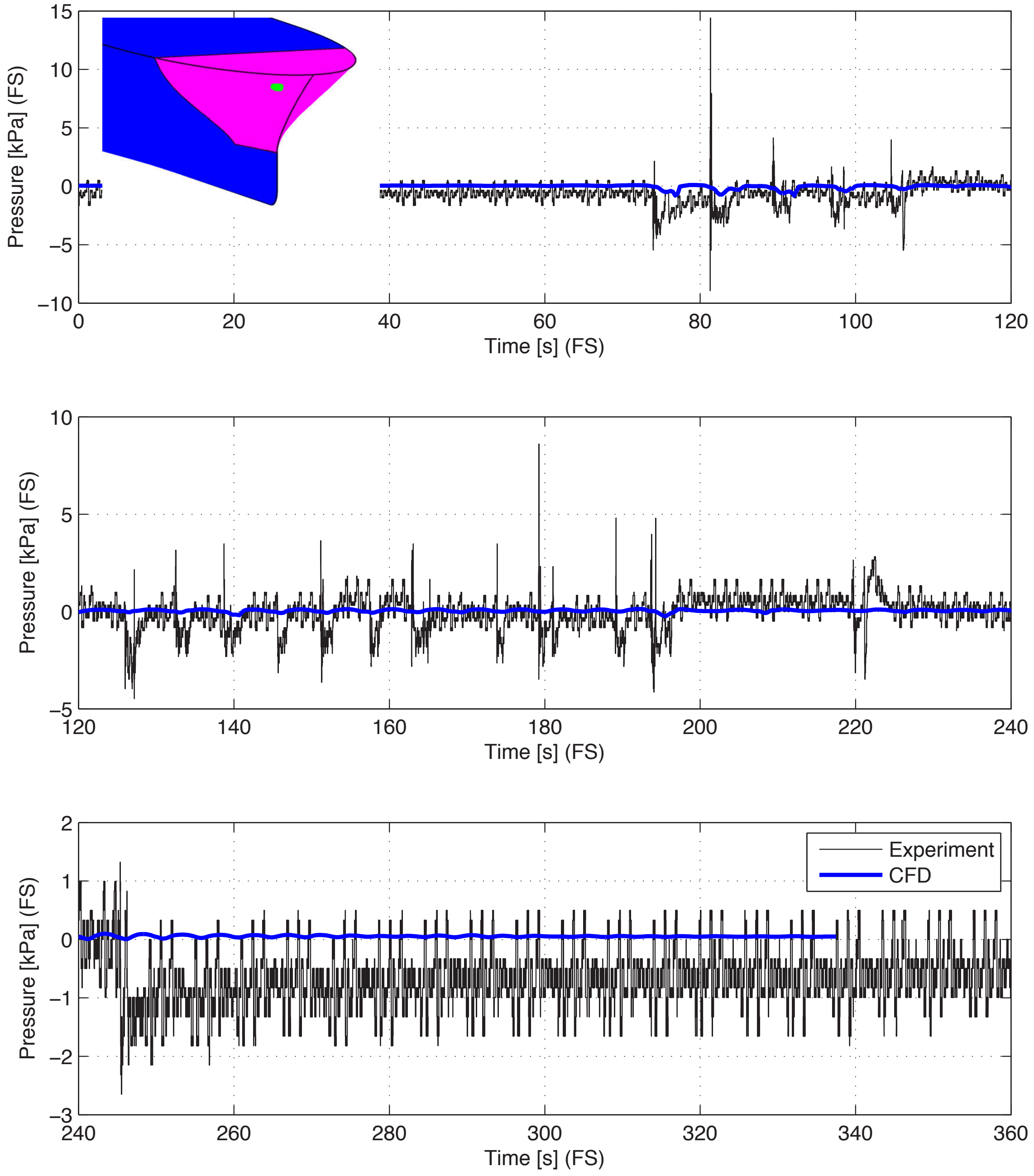

Figure H.17: *Pressure transducer #17 data*

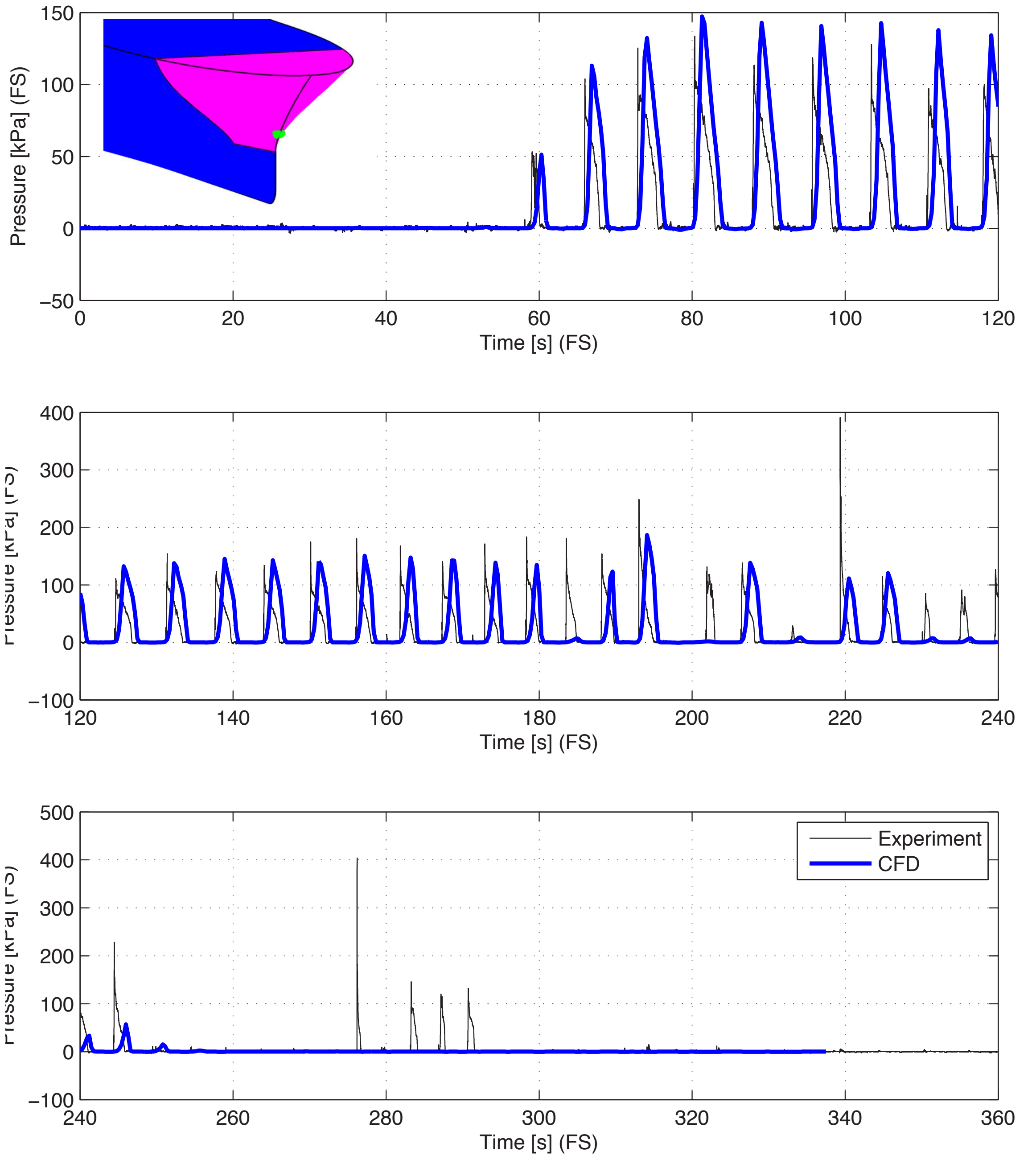

Figure H.18: *Pressure transducer #18 data*

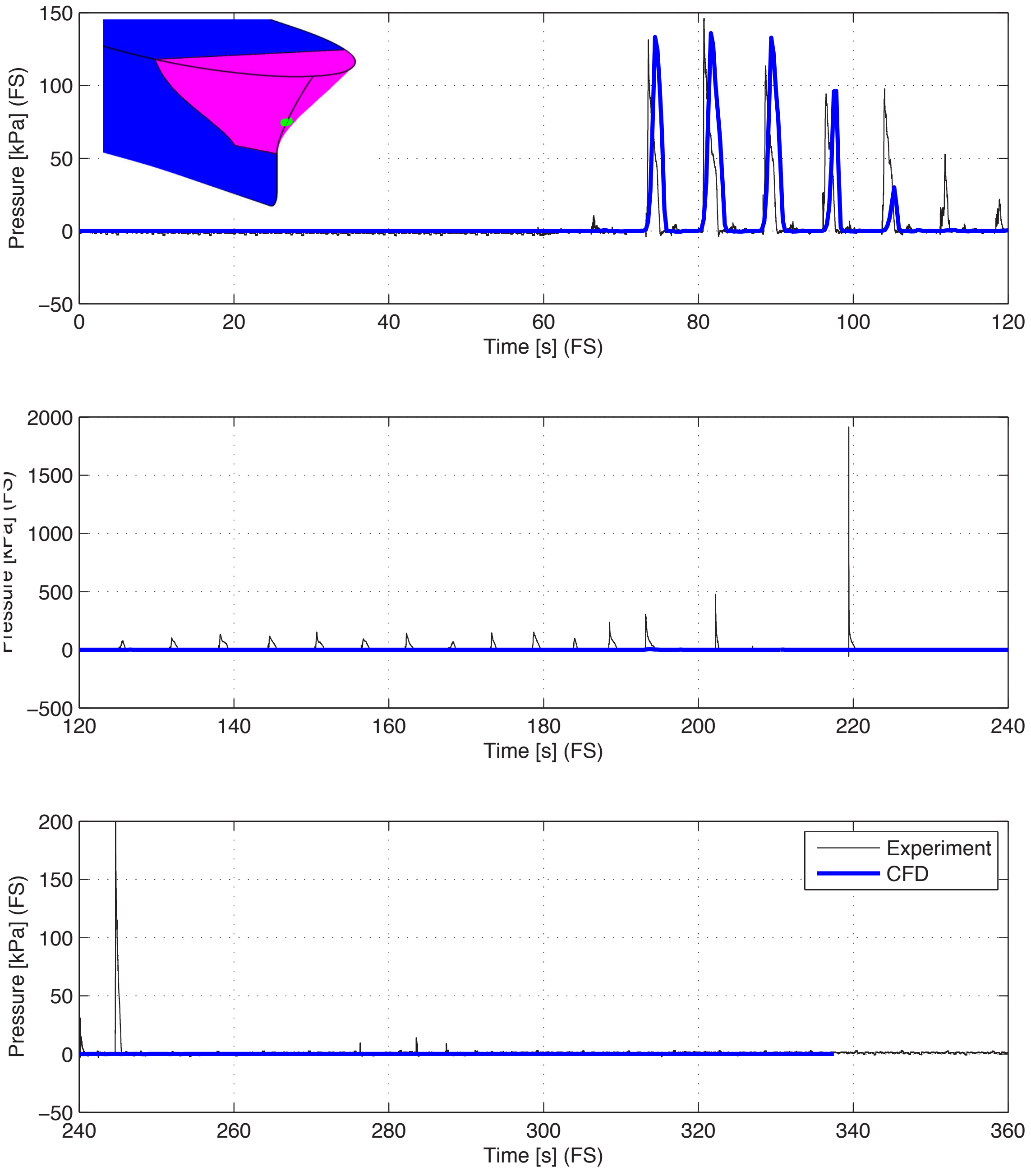

Figure H.19: *Pressure transducer #19 data*

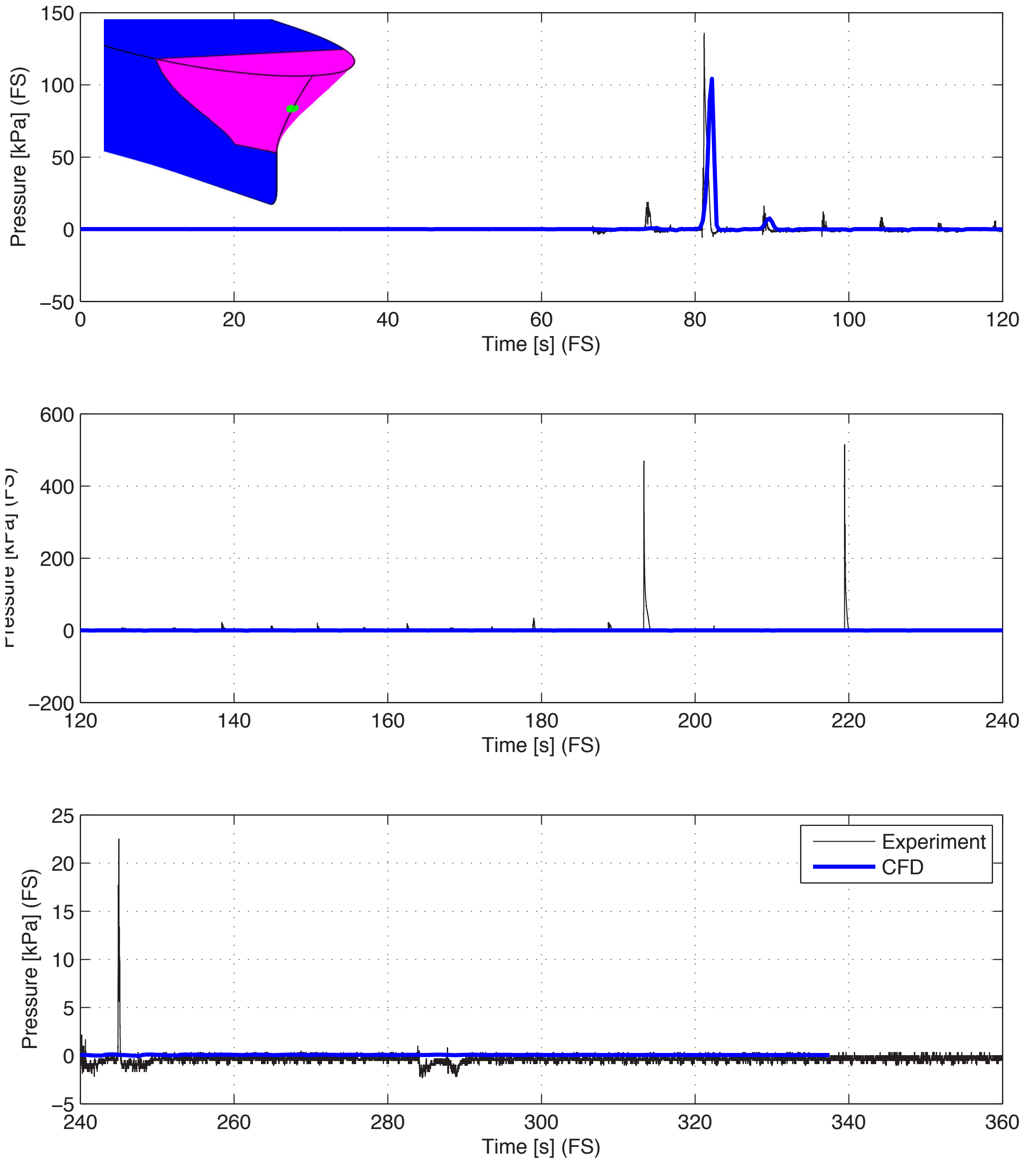

Figure H.20: *Pressure transducer #20 data*

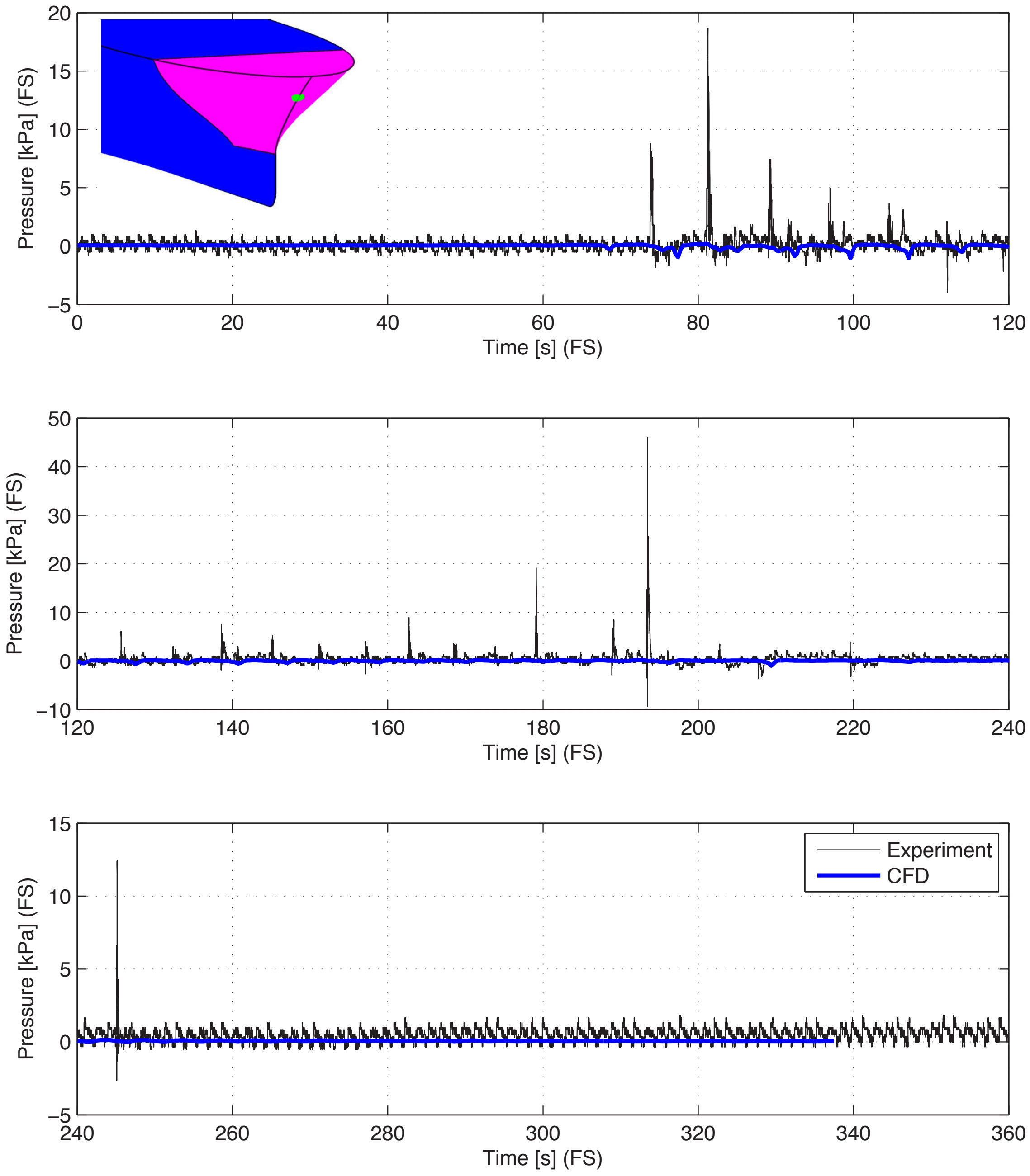

Figure H.21: *Pressure transducer #21 data*

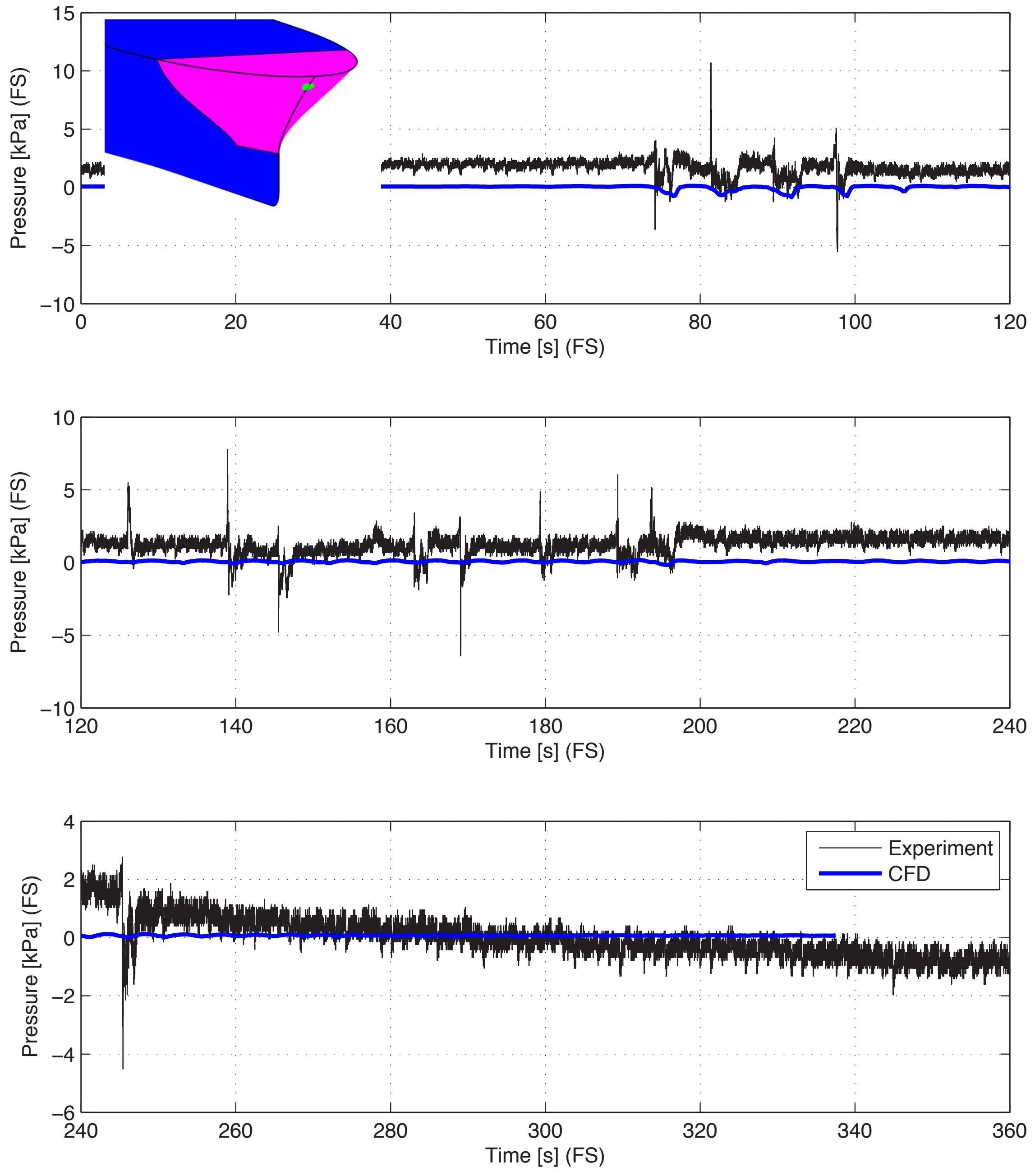

Figure H.22: *Pressure transducer #22 data*

Annex I: ELAST Ferry CFD Results

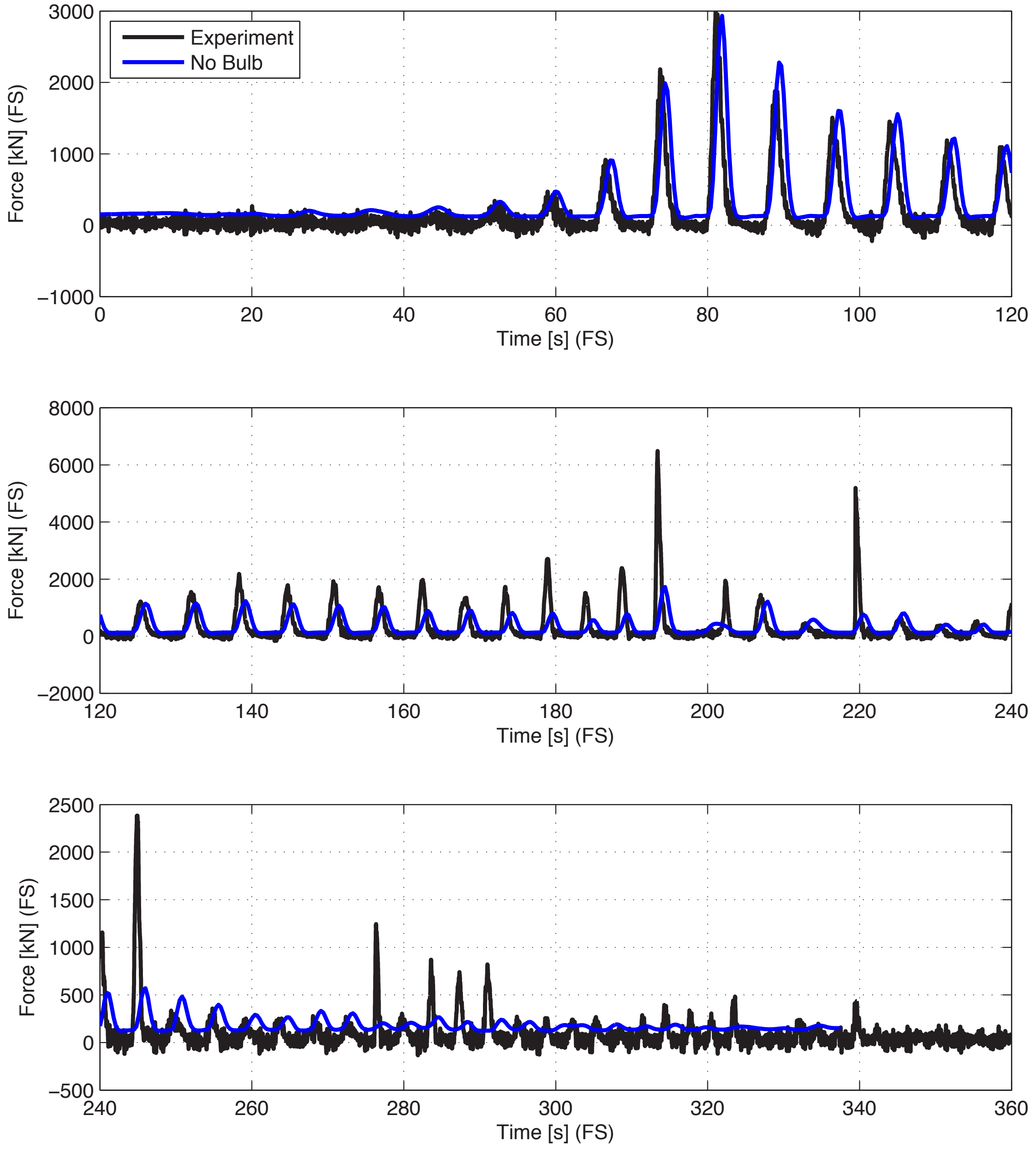

Figure I.1: *X-force on bow segment*

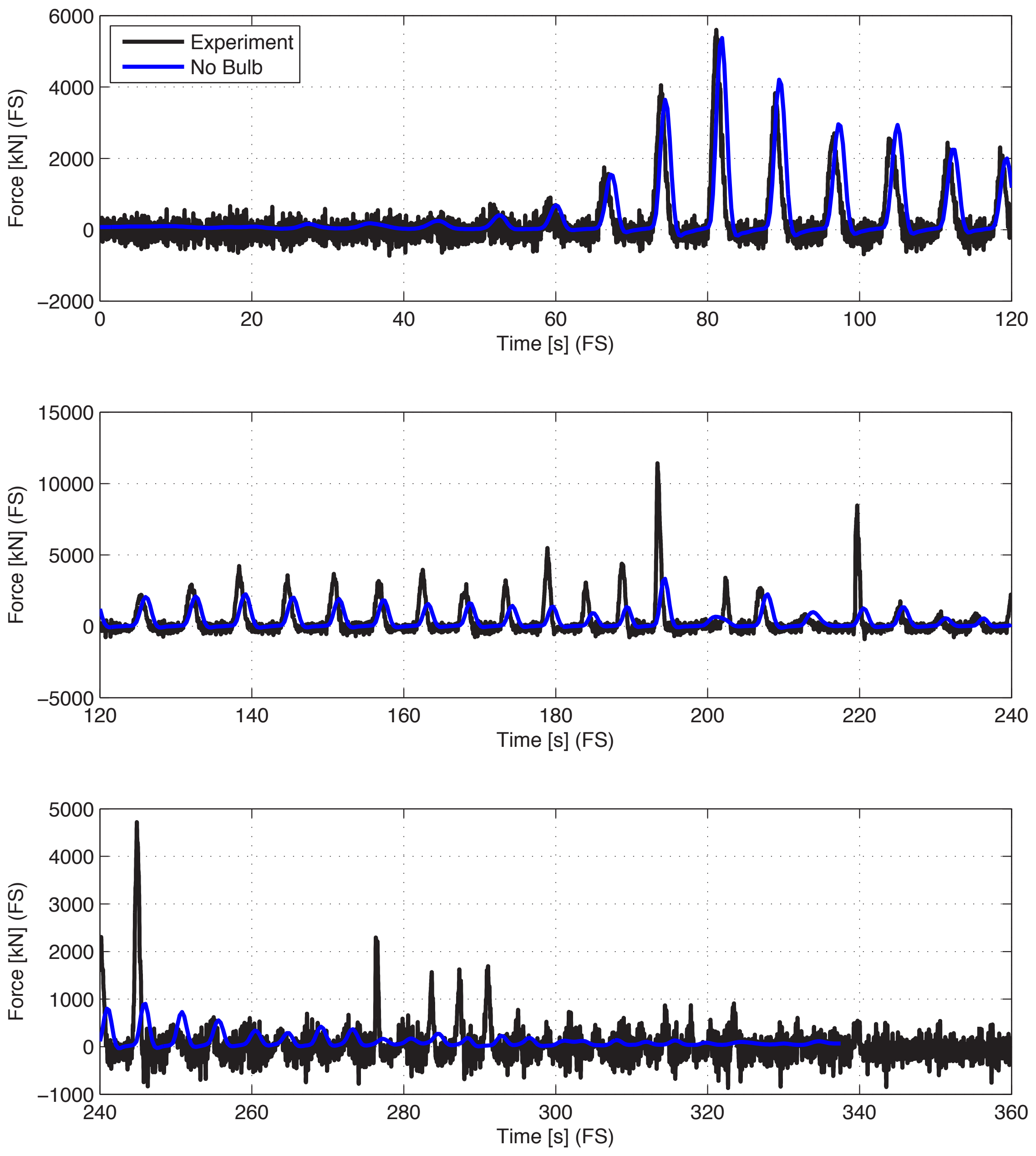

Figure I.2: *Z-force on bow segment*

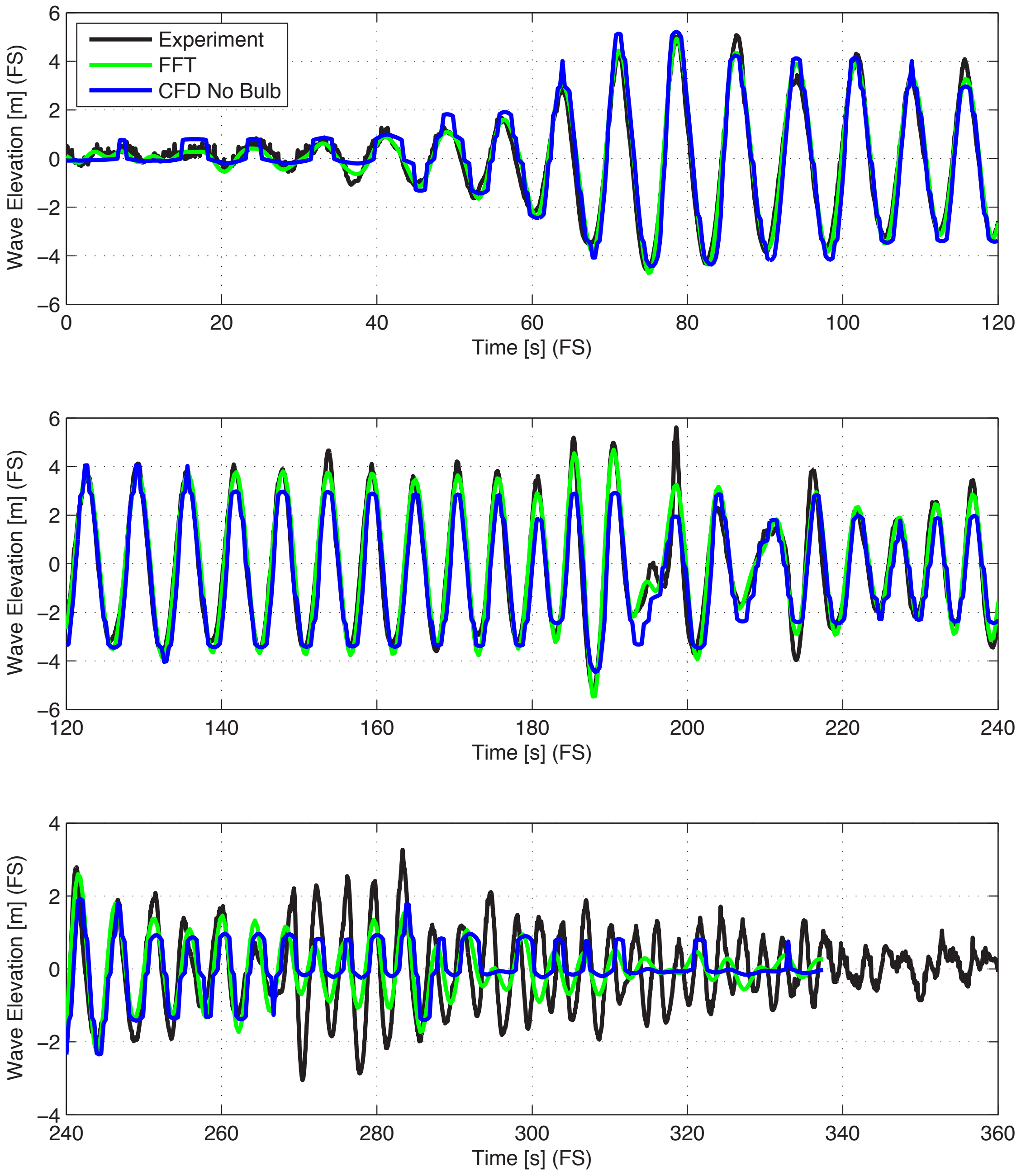

Figure I.3: *Wave probe #1 data*

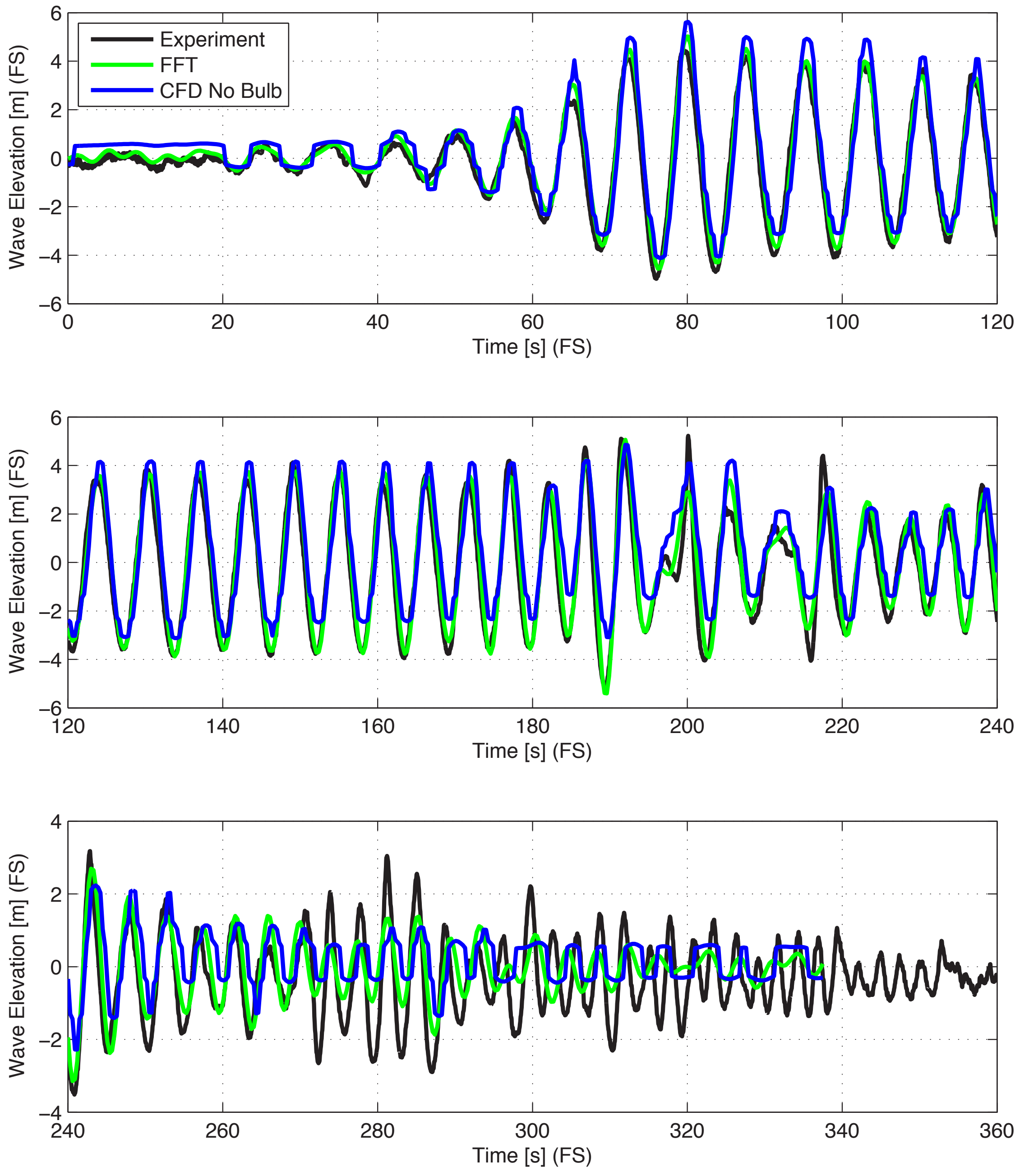

Figure I.4: *Wave probe #2 data*

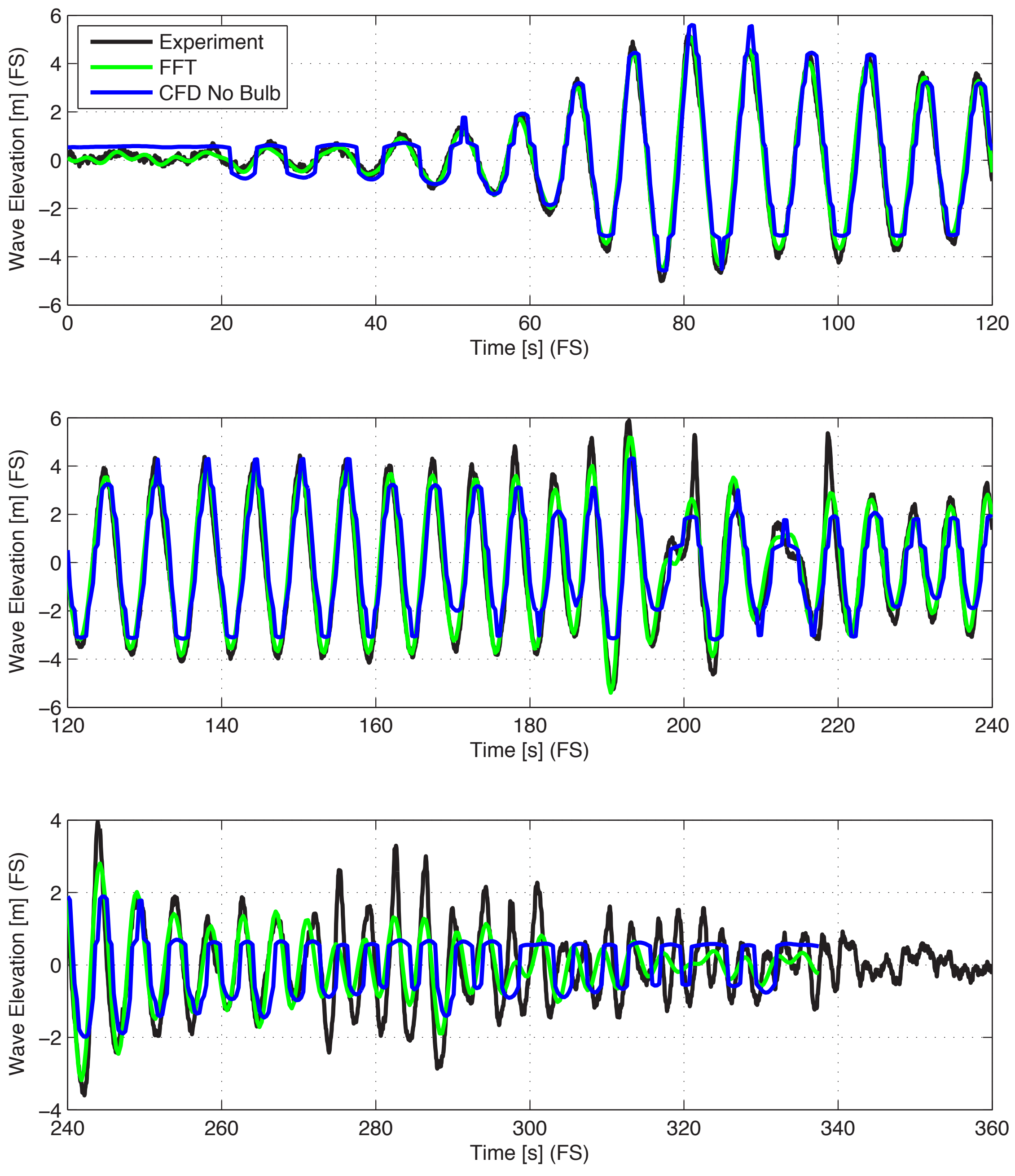

Figure I.5: *Wave probe #10 data*

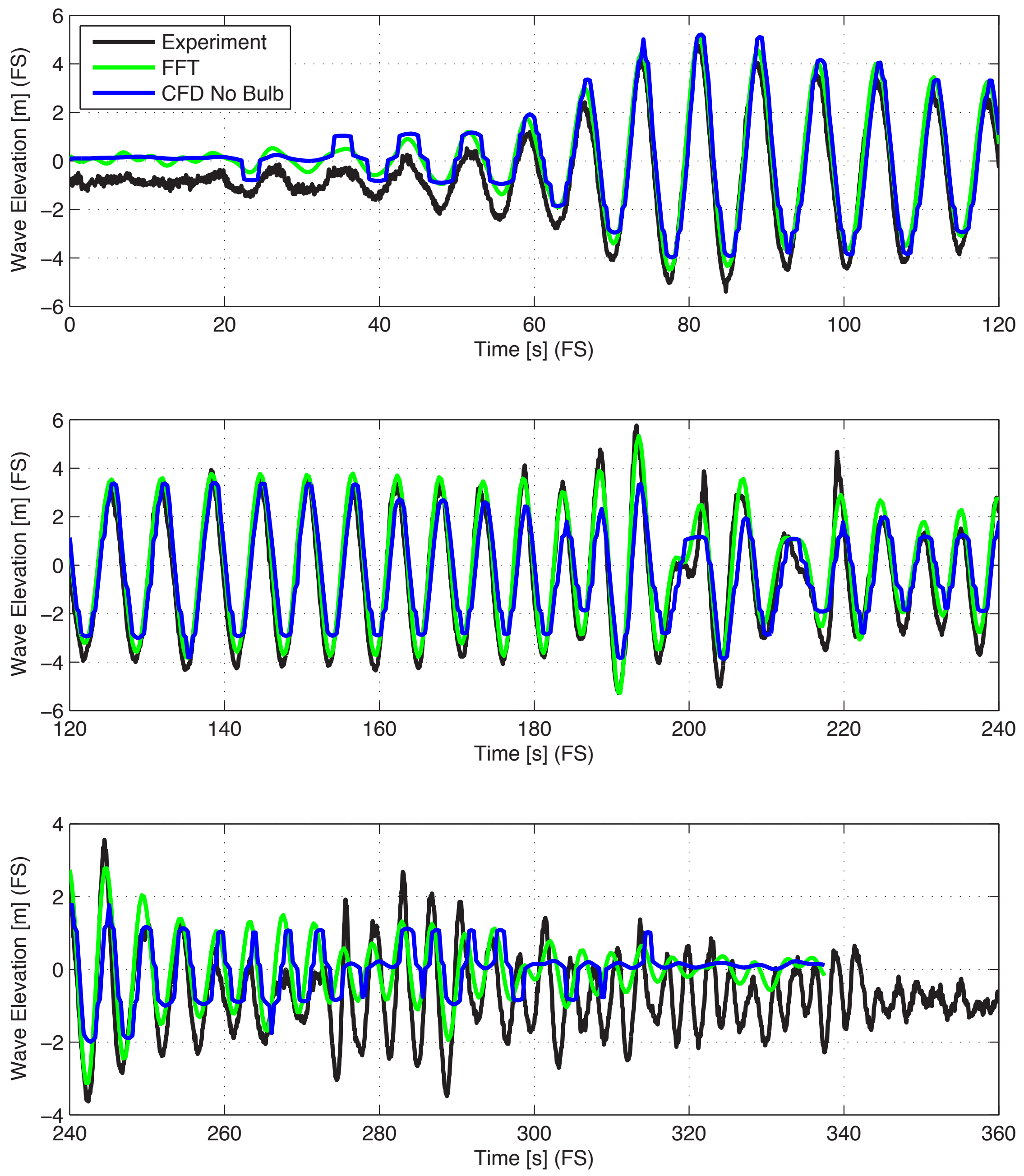

Figure I.6: *Wave probe #16 data*

Acronyms

BEM	boundary element method
CCL	CFX Command Language
CFD	Computational Fluid Dynamics
CRS	Cooperative Research Ships
DNV	Det Norske Veritas
DRDC	Defence Research and Development Canada
FFT	fast Fourier transform
FS	full scale
GUI	Graphical User Interface
IGES	Initial Graphics Exchange Specification
LBP	Length between perpendiculars
RANS	Reynolds-averaged Navier-Stokes
SGI	Silicon Graphics Incorporated
SST	Shear Stress Transport
VOF	volume-of-fluid

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated.)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research and Development Canada – Atlantic PO Box 1012, Dartmouth NS B2Y 3Z7, Canada	2a. SECURITY MARKING (Overall security marking of the document, including supplemental markings if applicable.) UNCLASSIFIED 2b. CONTROLLED GOODS (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC APRIL 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

CFD Simulations of a Ferry in Head Seas

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Thornhill, E.

5. DATE OF PUBLICATION (Month and year of publication of document.) March 2011	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.) 104	6b. NO. OF REFS (Total cited in document.) 3

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Atlantic
PO Box 1012, Dartmouth NS B2Y 3Z7, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) 11ge07	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC Atlantic TM 2011-030	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

A ship travelling in a seaway can often experience slamming characterized by large sudden loads from impacts with waves. These loads can be large enough to cause local damage, but also induce a whipping response which stresses the primary hull structure and adversely affects the overall fatigue life of the vessel. Most common seakeeping prediction programs use potential flow theory that cannot directly simulate the physics of slamming. This report contains a study using more advanced computational fluid dynamics methods which can reproduce wave impacts. Simulations were conducted for a ferry hull travelling in head seas and compared to existing experimental data. It was found that slamming loads are well predicted, but there can be difficulties in generating and propagating large steep waves inside the simulation.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

CFD, CRS, Elast Ferry, Slamming, Head Seas, Bow Flare