

## ABSTRACT

Title of dissertation: INVESTIGATIONS INTO MECHANISMS  
UNDERLYING EXTREME WAVE  
FORMATIONS AND COMPUTATIONALLY  
INTENSIVE SIMULATIONS

Ayan Moitra, 2016

Dissertation directed by: Professor Balakumar Balachandran  
Department of Mechanical Engineering

Various mechanisms have been proposed to explain extreme waves or rogue waves in an oceanic environment including directional focusing, dispersive focusing, wave-current interaction, and nonlinear modulational instability. The Benjamin-Feir instability (nonlinear modulational instability), however, is considered to be one of the primary mechanisms for rogue-wave occurrence. The nonlinear Schrödinger equation is a well-established approximate model based on the same assumptions as required for the derivation of the Benjamin-Feir theory. Solutions of the nonlinear Schrödinger equation, including new rogue-wave type solutions are presented in the author's dissertation work. The solutions are obtained by using a predictive eigenvalue map based predictor-corrector procedure developed by the author. Features of the predictive map are explored and the influences of certain parameter variations are investigated. The solutions are rescaled to match the length scales of waves generated in a wave tank. Based on the information provided by the map and the details of physical scaling, a framework is developed that can serve as a basis for

experimental investigations into a variety of extreme waves as well localizations in wave fields.

To derive further fundamental insights into the complexity of extreme wave conditions, Smoothed Particle Hydrodynamics (SPH) simulations are carried out on an advanced Graphic Processing Unit (GPU) based parallel computational platform. Free surface gravity wave simulations have successfully characterized water-wave dispersion in the SPH model while demonstrating extreme energy focusing and wave growth in both linear and nonlinear regimes. A virtual wave tank is simulated wherein wave motions can be excited from either side. Focusing of several wave trains and isolated waves has been simulated. With properly chosen parameters, dispersion effects are observed causing a chirped wave train to focus and exhibit growth. By using the insights derived from the study of the nonlinear Schrödinger equation, modulational instability or self-focusing has been induced in a numerical wave tank and studied through several numerical simulations. Due to the inherent dissipative nature of SPH models, simulating persistent progressive waves can be problematic. This issue has been addressed and an observation-based solution has been provided. The efficacy of SPH in modeling wave focusing can be critical to further our understanding and predicting extreme wave phenomena through simulations.

A deeper understanding of the mechanisms underlying extreme energy localization phenomena can help facilitate energy harnessing and serve as a basis to predict and mitigate the impact of energy focusing.

INVESTIGATIONS INTO MECHANISMS UNDERLYING  
EXTREME WAVE FORMATIONS AND COMPUTATIONALLY  
INTENSIVE SIMULATIONS

by

Ayan Moitra

Dissertation submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
2016

Advisory Committee:

Professor Balakumar Balachandran, Chair and Advisor, Department of Mechanical  
Engineering

Professor Eugenia Kalnay, Department of Atmospheric and Oceanic Science (Dean's  
Representative)

Professor Amr Baz, Department of Mechanical Engineering

Associate Professor Peter Chung, Department of Mechanical Engineering

Assistant Professor Amir Riaz, Department of Mechanical Engineering

© Copyright by  
Ayan Moitra  
2016

## Acknowledgments

First and foremost I'd like to thank my advisor, Professor Balakumar Balachandran for giving me an invaluable opportunity to work on this challenging and extremely interesting project. He has been incredibly patient in guiding me in my development as a researcher. His key inputs and criticisms have been an important factor in the completion of this research in a timely manner. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank the members of my dissertation committee: Professor Eugenia Kalnay, Professor Amr Baz, Professor Peter Chung, and Professor Amir Riaz for their time and insightful feedback during my proposal and dissertation defenses. I would also like to thank Prof. Massimo Ricotti, a co-principal investigator on the NSF project that supported this work, for his valuable insights and support.

I would especially like to thank Dr. Christopher Chabalko. Without his cooperation, insights and computational expertise, I could not have successfully completed this thesis on time. I would like to thank him for helping me develop a strong foundation in the field of GPGPU-based parallel computing.

My appreciation goes to my past and present lab mates: Chris Chabalko, Arseniy Zakharov, Tim Fitzgerald, Marcelo Valdez, Vince Nguyen, Edmon Perkins, Hesham Ismail, Celeste Poley, Vipin Agarwal, Shane Hsu, Abdullah Zibdeh, and Saliou Telly for all these years together on the pursuits of our goals, and for their inputs during our uncountable group meetings.

My housemates at my place of residence have been a crucial factor in my

finishing smoothly. I'd like to express my gratitude to Inderjeet Singh Khurana, Gagandeep Singh Kohli, Shalabh Parmar and Suresh Kanagala for their friendship and support.

Finally, and most importantly, I would like to thank my parents and my companion-to-be, Payel Chatterjee, for their constant love, support, guidance, patience and many many other things that one takes for granted. This would not have been possible without them.

Support received for this research through NSF Grant No. CMMI-1125285 is gratefully acknowledged.

## Table of Contents

List of Figures	vii
List of Abbreviations	xii
1 Introduction	1
1.1 Problem of Interest . . . . .	1
1.2 Objectives . . . . .	3
1.3 Basic Water Wave Mechanics . . . . .	4
1.4 The Nonlinear Schrödinger Equation . . . . .	7
1.5 Benjamin-Feir Instability . . . . .	10
1.6 Periodic Spectral Theory . . . . .	13
1.7 Focusing as a Mechanism for Rogue-wave Formation . . . . .	16
1.7.1 Linear Focusing . . . . .	17
1.7.2 Nonlinear Dispersive Focusing . . . . .	17
1.7.3 Nonlinear Directional Focusing . . . . .	20
1.8 Outline . . . . .	22
2 The Nonlinear Schrödinger Equation and Rogue-wave Solutions	24
2.1 Literature Review . . . . .	24
2.2 Analytical Solutions to the NSE . . . . .	26
2.2.1 Lax's Generalization . . . . .	26
2.2.2 Nonlinear Fourier Structure of the NSE Solution Space . . . . .	29
2.2.3 Floquet Theory . . . . .	31
2.2.3.1 Floquet Theory Applied to the Spectral Problem of NSE . . . . .	31
2.2.3.2 Stability . . . . .	33
2.2.3.3 Numerical Algorithm . . . . .	36
2.2.3.4 Results . . . . .	38
2.2.4 Reconstruction of Potential . . . . .	46
2.2.4.1 Solutions using Hyperelliptic Functions . . . . .	47
2.2.4.2 Solutions using Riemann Theta Functions . . . . .	49
2.2.5 Solution Procedure to Explore $\lambda$ Plane . . . . .	50

2.2.5.1	Prediction . . . . .	52
2.2.5.2	Correction . . . . .	53
2.2.5.3	Verification . . . . .	56
2.2.6	New Rogue-wave Solutions . . . . .	57
2.2.6.1	Near Peregrine Solution . . . . .	60
2.2.6.2	Isolated Solution . . . . .	61
2.2.6.3	Transition from Rogue Wave to Amplified Wave . . .	62
2.2.7	Physical Scaling . . . . .	68
2.2.7.1	Predicted Evolution of a Dimensional Wave Field . .	70
3	Computational Studies of Extreme Energy Localization using Smoothed Particle Hydrodynamics . . . . .	75
3.1	Literature Review . . . . .	75
3.2	Smoothed Particle Hydrodynamics . . . . .	80
3.2.1	Momentum Equation . . . . .	86
3.2.2	Continuity Equation . . . . .	87
3.2.3	Equation of State . . . . .	88
3.2.4	Viscosity . . . . .	89
3.2.5	XSPH Correction . . . . .	90
3.2.6	Density Reinitialization . . . . .	91
3.2.7	Time Integration . . . . .	91
3.2.8	Parallel Implementation of the Algorithm . . . . .	92
3.3	Numerical Studies in Two-dimensional cases . . . . .	95
3.3.1	Smoothing Kernels . . . . .	95
3.3.2	Model Validation . . . . .	97
3.3.2.1	Dam Break . . . . .	97
3.3.2.2	Dispersion in the SPH Model . . . . .	100
3.3.3	Progressive Wave Generation and Dissipation in SPH Model .	102
3.3.4	Standing Waves in 1+1 Dimension . . . . .	106
3.3.5	Directional Focusing in 1+1 Dimension . . . . .	106
3.3.5.1	Case Study 1: $f = 1.2$ Hz; $S = 2$ cm . . . . .	109
3.3.5.2	Case Study 2: $f = 1.0$ Hz; $S = 3$ cm . . . . .	112
3.3.5.3	Case Study 3: $f = 0.8$ Hz; $S = 4$ cm . . . . .	112
3.3.5.4	Case Study 4: $f = 0.6$ Hz; $S = 5$ cm . . . . .	115
3.3.6	Dispersive Focusing in 1+1 Dimension . . . . .	120
3.3.7	Modulational Instability in 1+1 Dimension . . . . .	125
3.3.7.1	Case Study 1: $a_0 = 2.5$ cm; $L_0 = 54$ cm . . . . .	132
3.3.7.2	Case Study 2: $a_0 = 9.0$ cm; $L_0 = 162$ cm . . . . .	137
4	Summary and Concluding Remarks . . . . .	142
4.1	Summary of Contributions . . . . .	142
4.2	Recommendations for Future Work . . . . .	144
A	Mathematical Details of Floquet Theory . . . . .	147

B	Additional Physical Forms of New Rogue Wave Solutions to the NSE	154
C	Sample Codes	159
C.1	Smoothed Particle Hydrodynamics CUDA C++ Code Snippets . . .	159
C.1.1	CUDA Kernel to Initialize Density . . . . .	159
C.1.2	CUDA Kernel to Compute State Rates . . . . .	161
C.1.3	CUDA Kernel to Update Velocity and Position . . . . .	164
C.1.4	CUDA Kernel to Reinitialize Density . . . . .	167
C.2	Predictor-Corrector MATLAB Code Snippets . . . . .	170
	Bibliography	174

## List of Figures

1.1	Rogue wave off of Charleston, South Carolina . . . . .	3
1.2	Small-amplitude modulation of a carrier wave. . . . .	8
1.3	Instability diagram for small-amplitude modulations for the NSE. . .	12
1.4	Formation of the freak wave of Gaussian form in shallow water (Source: [22]). . . . .	18
1.5	Formation of the freak wave due to geometric focusing (Source: [23]).	18
1.6	The process of the freak wave formation from the nonlinear-dispersive wavetrain for different times (Source: [42]). . . . .	20
1.7	Snake wave maker used in directional focusing (Source: [18]). . . . .	21
1.8	Rogue waves simulated via directional focusing (Source: [18]). . . . .	21
2.1	$\lambda$ plane showing main spectrum eigenvalues with a detailed view of $\epsilon$ and $\theta$ for a particular choice of $(\lambda_R, \lambda_I)$ (Source: [40]). . . . .	31
2.2	Initial modulated wave train with $\epsilon = 10^{-5}$ and wavelength $L = 4.44$ .	39
2.3	$\lambda$ plane spectrum of a plane carrier wave which is modulated by unsta- ble, small-amplitude ( $\epsilon = 10^{-5}$ ) sine wave and has one (homoclinic) unstable mode. . . . .	40
2.4	Initial modulated wave train with $\epsilon = 0.05$ and wavelength $L = 2$ . . .	40
2.5	$\lambda$ plane spectrum of a plane carrier wave which is modulated by stable, small-amplitude ( $\epsilon = 0.05$ ) sine wave and has two stable modes. . . .	41
2.6	$\lambda$ plane spectrum of a plane carrier wave which is modulated by small- amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength $L = 4.44$ . . . . .	41
2.7	$\lambda$ plane spectrum of a plane carrier wave which is modulated by small- amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength $L = 10$ . . . . .	42
2.8	$\lambda$ plane spectrum of a plane carrier wave which is modulated by small- amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength $L = 100$ . . . . .	42
2.9	Initial modulated wave train with $\epsilon_1 = 0.1i$ , $\epsilon_2 = 0.1$ , $\theta_1 =$ $\pi/3$ , $\theta_2 = \pi/6$ , and wavelength $L = 10$ . . . . .	43
2.10	$\lambda$ plane spectrum of the initial wavetrain shown in Figure 2.9. . . . .	44
2.11	Initial modulated wavetrain with parameters shown in equations (2.34) and (2.35) and wavelength $L = 10$ . . . . .	45
2.12	Actual envelope profile (magnified) of the initial wavetrain shown in Figure 2.11. . . . .	45

2.13	$\lambda$ plane spectrum of the initial wavetrain shown in Figure 2.11. . . . .	46
2.14	Modulus of the space time evolution of the rogue-wave solution for $\lambda = ia/\sqrt{2}$ . . . . .	48
2.15	Modulus of the space time evolution of the rogue-wave solution for $\lambda = ia\sqrt{2}$ . . . . .	48
2.16	Flow chart illustrating the procedure to determine new Reimann theta function described rogue waves. . . . .	51
2.17	Map of periodic Reimann theta functions as defined in equation (2.41) for $(A = 1, \theta = 0, \epsilon = 0.01, t = 0)$ generated by GPGPU computations. Light colored locations indicate periodic functions, while dark colored locations indicate aperiodic functions over the interval $L$ . A point of interest is identified by an asterisk. . . . .	54
2.18	Main spectrum eigenvalues with a detailed view of $\epsilon$ and $\theta$ for a particular choice of $(\lambda_R, \lambda_I)$ . . . . .	55
2.19	Solution to the NSE for $(\lambda_R = 1.2415, \lambda_I = 1.61108, \epsilon = 0.006834, \theta = 1.11439)$ and $L = 4.44$ . This solution envelope has periodic temporal peaks, which reach a maximum amplitude of $\approx 4.2x$ the background. . . . .	58
2.20	GPU maps for $(L = 7, \epsilon = 0.005)$ and $(L = 12, \epsilon = 0.01)$ . . . . .	58
2.21	Solution to the NSE for $(\lambda_R = 0.6616, \lambda_I = 1.23660, \epsilon = 0.00276, \theta = 0.87385)$ and $L = 7$ . . . . .	59
2.22	Solution to the NSE for $(\lambda_R = 1.1367, \lambda_I = 1.2076, \epsilon = 0.004156, \theta = 0.372019)$ and $L = 12$ . . . . .	60
2.23	“Near Peregrine” solution with eigenvalues $(\lambda_R = 0.0098, \lambda_I = 1.0068)$ close to those associated with the Peregrine solution of the NSE. . . . .	61
2.24	Predictive map for isolated solution $(\lambda_R = 1.7806, \lambda_I = 0.99603)$ . . . . .	62
2.25	Solution to the NSE for $(\lambda_R = 1.7806, \lambda_I = 0.99603), \epsilon = 0.0027655086, \theta = 0.007827789$ , and $L = 10$ . This solution envelope reaches a maximum amplitude of $\approx 3x$ the background height. . . . .	63
2.26	Composite predictive map containing the highest solution band for multiple values of $L$ . Corrected pairs of $(\lambda_R, \lambda_I)$ for generated solutions are marked with an ‘x’. . . . .	64
2.27	Peaked rogue-wave: $L = 6; \lambda = 0.01178 + 2.5512i$ ; and $A_{max} = 6.1$ . . . . .	65
2.28	Peaked rogue-wave: $L = 8; \lambda = 0.0107 + 1.9373i$ ; and $A_{max} = 4.87$ . . . . .	66
2.29	Peaked rogue-wave: $L = 10; \lambda = 0.00609 + 1.5485i$ ; and $A_{max} = 4.2$ . . . . .	67
2.30	Peaked rogue-wave: $L = 12; \lambda = 0.0086 + 1.402i$ ; and $A_{max} = 3.8$ . . . . .	67
2.31	Peaked rogue-wave: $L = 14; \lambda = 0.008 + 1.276i$ ; and $A_{max} = 3.55$ . . . . .	68
2.32	Near Peregrine solution (solid line) appears identical to the Peregrine solution (marked with squares) over the interval examined. . . . .	71
2.33	Predicted temporal evolution for the near Peregrine solution $(\lambda_R = 0.0098, \lambda_I = 1.0068)$ . Surface heights are shown at various distances. . . . .	72
2.34	A wave field showing the predicted temporal evolution of a rogue-wave solution $(\lambda_R = 1.1367, \lambda_I = 1.2076)$ with two localization events highlighted. . . . .	72
2.35	Detailed view of dimensional rogue-wave solution with $\lambda = (1.1367, 1.2076i)$ . . . . .	74

3.1	SPH simulated waves produced by a wavemaker (Source: [32]). . . . .	77
3.2	Solitary waves in a horizontal tank simulated using SPH (Source: [34]).	78
3.3	Wave packet evolution and focusing (Source: [13]). . . . .	79
3.4	Wave breaking and collapsing (Source: [13]). . . . .	79
3.5	Representation of a smoothing kernel in three dimensions. . . . .	82
3.6	Schematic representation of the CUDA-enabled parallel SPH Algorithm. . . . .	94
3.7	The $0^{th}$ derivative (upper) and $1^{st}$ derivative (lower) smoothing kernels used in this work. . . . .	96
3.8	The present SPH simulation results exhibit qualitatively similar behavior to previous SPH simulation results for the validation case of a dam break. . . . .	97
3.9	For a similar configuration, the results obtained from the present method are seen to be in quantitative agreement with previous experimental [29] and numerical results [54]. . . . .	98
3.10	Dam break experiment by Martin and Moyce [29]. . . . .	99
3.11	Actual $\omega^2$ versus Evaluated $\omega^2$ : Data consolidated from five numerical experiments to verify dispersion relation in the current SPH model . .	101
3.12	Plot showing the variation of SPH simulated wavelength versus the forcing frequency. . . . .	101
3.13	Numerical tank of length 4.5 m is simulated by using 88000 particles with water depth of 0.35 m. Left wall is forced with a sinusoidal function of frequency $f = 1.4$ Hz and stroke amplitude 2 cm. The case study with higher smoothing length ( $h$ ) exhibits considerably lower dissipation. . . . .	105
3.14	1+1 Dimension Standing Waves Case Study ( $t = 4.000$ secs to $t = 8.415$ secs): Numerical wave tank of length 15 m and water depth 1.0 m simulated using 90000 particles. . . . .	107
3.15	1+1 Dimension Standing Waves Case Study ( $t = 10.230$ secs to $t = 13.770$ secs). . . . .	108
3.16	1+1 Dimension Directional Focusing Case Study 1 ( $t = 3.174$ secs to $t = 3.900$ secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Both, left and right walls excited in a sinusoidal manner with a frequency of 1.2 Hz and stroke amplitude of 2 cm. The wave-fronts interfere in the middle of the tank. . . . .	110
3.17	1+1 Dimension Directional Focusing Case Study 1 ( $t = 4.314$ secs to $t = 8.160$ secs). . . . .	111
3.18	1+1 Dimension Directional Focusing Case Study 2 ( $t = 1.536$ secs to $t = 2.730$ secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Both, left and right walls excited in a sinusoidal manner with a frequency of 1.0 Hz and stroke amplitude of 3 cm. The wave-fronts interfere in the middle of the tank. . . . .	113

3.19	1+1 Dimension Directional Focusing Case Study 2 ( $t = 3.162$ secs to $t = 4.122$ secs).	114
3.20	1+1 Dimension Directional Focusing Case Study 3: Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Left and right walls excited in a sinusoidal manner with a frequency of 0.8 Hz and stroke amplitude of 4 cm.	116
3.21	1+1 Dimension Directional Focusing Case Study 3: Particles colored based on velocity magnitudes.	117
3.22	1+1 Dimension Directional Focusing Case Study 4 ( $t = 2.242$ secs to $t = 3.918$ secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Left and right walls excited in a sinusoidal manner with a frequency of 0.6 Hz and stroke amplitude of 5 cm.	118
3.23	1+1 Dimension Directional Focusing Case Study 4 ( $t = 4.746$ secs to $t = 6.246$ secs).	119
3.24	Sinusoidal Motion described by the left wall to generate waves. Constant frequency for the first 9 cycles. Frequency variation commences at $t = 8.65$ secs.	122
3.25	1+1 Dimension Dispersive Focusing Case Study ( $t = 10.700$ secs to $t = 15.250$ secs): Numerical wave tank of length 15 m and water depth 1.20 m simulated using 240000 particles.	123
3.26	1+1 Dimension Dispersive Focusing Case Study ( $t = 16.150$ secs to $t = 16.700$ secs): Numerical wave tank of length 15 m and water depth 1.20 m simulated using 240000 particles.	124
3.27	Parametric study showing generated wave amplitude versus wave maker amplitude for different water depths.	126
3.28	Parametric study showing generated wave length versus wave maker amplitude for different forcing frequencies and water depths. These results follow the dispersion relation. See Figure 3.11.	127
3.29	Evolution of the dimensional form of Peregrine breather as described by equation (3.44) ( $a_0 = 0.01$ m, $L_0 = 0.54$ m).	129
3.30	Predicted surface profile according to the analytic formulation of the Peregrine breather as described by equation (3.44) ( $t = 6.0$ secs to $t = 9.5$ secs). The perturbation is introduced after 10 lead cycles. ( $a_0 = 0.025$ m, $L_0 = 0.54$ m).	130
3.31	Predicted surface profile according to the analytic formulation of the Peregrine breather as described by equation (3.44) ( $t = 10.0$ secs to $t = 11.0$ secs).The perturbation is introduced after 10 lead cycles. ( $a_0 = 0.025$ m, $L_0 = 0.54$ m).	131
3.32	Time history of the wave maker to theoretically induce the surface evolution shown in Figures 3.30 and 3.31. The wave maker amplitude is 0.01 m produces a carrier wave of amplitude 0.025 m.	132
3.33	1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$ m, $L_0 = 0.54$ m): The perturbation is introduced after 10 lead cycles ( $x = 0.5$ m to $x = 1.8$ m).	134

3.34	1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$ m, $L_0 = 0.54$ m): ( $x = 2.0$ m to $x = 2.4$ m). . . . .	135
3.35	1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$ m, $L_0 = 0.54$ m): ( $x = 2.6$ m to $x = 3.5$ m). . . . .	136
3.36	1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$ m, $L_0 = 1.62$ m): The perturbation is introduced after 10 lead cycles ( $x = 1.0$ m to $x = 4.0$ m). . . . .	139
3.37	1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$ m, $L_0 = 1.62$ m): ( $x = 5.0$ m to $x = 8.0$ m). . . . .	140
3.38	1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$ m, $L_0 = 1.62$ m): ( $x = 9.0$ m to $x = 11.0$ m). . . . .	141
B.1	Solution to the NSE for ( $\lambda_R = 0.87528$ , $\lambda_I = 1.3008$ ), $\epsilon = 0.008014$ , $\theta = 0.629216906$ , and $L = 5.44$ . This solution envelope reaches a maximum amplitude of $\approx 3.6x$ the background height. . . . .	154
B.2	Solution to the NSE for ( $\lambda_R = 1.22$ , $\lambda_I = 1.852$ ), $\epsilon = 0.0026265$ , $\theta = 1.047105344$ , and $L = 4.44$ . This solution envelope reaches a maximum amplitude of $\approx 4.7x$ the background height. . . . .	155
B.3	Solution to the NSE for ( $\lambda_R = 0.7356$ , $\lambda_I = 0.8237$ ), $\epsilon = 2.53E - 03$ , $\theta = -1.051080567$ , and $L = 20$ . This solution envelope reaches a maximum amplitude of $\approx 2.65x$ the background height. . . . .	155
B.4	Solution to the NSE for ( $\lambda_R = -0.0167$ , $\lambda_I = 0.9805$ ), $\epsilon = 0.019545839$ , $\theta = 1.178877343$ , and $L = 10$ . This solution envelope reaches a maximum amplitude of $\approx 3x$ the background height. . . . .	156
B.5	Solution to the NSE for ( $\lambda_R = 0.14615$ , $\lambda_I = 0.90249$ ), $\epsilon = 0.024303445$ , $\theta = -0.835951856$ , and $L = 10$ . This solution envelope reaches a maximum amplitude of $\approx 2.8x$ the background height. . . . .	156
B.6	Solution to the NSE for ( $\lambda_R = 0.54774$ , $\lambda_I = 0.654$ ), $\epsilon = 0.007805263$ , $\theta = -0.676122251$ , and $L = 15$ . This solution envelope reaches a maximum amplitude of $\approx 2.3x$ the background height. . . . .	157
B.7	Solution to the NSE for ( $\lambda_R = 0.60503$ , $\lambda_I = 2.2946$ ), $\epsilon = 0.000495576$ , $\theta = 1.236466627$ , and $L = 4.44$ . This solution envelope reaches a maximum amplitude of $\approx 5.6x$ the background height. . . . .	157
B.8	Solution to the NSE for ( $\lambda_R = 1.5948$ , $\lambda_I = 2.5409$ ), $\epsilon = 0.000160675$ , $\theta = -0.475103634$ , and $L = 4.44$ . This solution envelope reaches a maximum amplitude of $\approx 6x$ the background height. . . . .	158
B.9	Solution to the NSE for ( $\lambda_R = 0.0099$ , $\lambda_I = 1.7498$ ), $\epsilon = 0.002485633$ , $\theta = 0.14954754$ , and $L = 9$ . This solution envelope reaches a maximum amplitude of $\approx 4.5x$ the background height. . . . .	158

## List of Abbreviations

CUDA	Compute Unified Device Architecture
GPGPU	General Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
IST	Inverse Scattering Transform
KdV	Korteweg-de Vries
NSE	Nonlinear Schrödinger Equation
PDE	Partial Differential Equation
SPH	Smoothed Particles Hydrodynamics
WSPH	Weakly Compressible SPH

## Chapter 1: Introduction

### 1.1 Problem of Interest

Freak, rogue, or giant waves correspond to large-amplitude waves surprisingly appearing on the sea surface (“wave from nowhere”). Such waves can be accompanied by deep troughs (holes), which occur before and /or after the largest crest. In one definition, the amplitude of rogue waves should exceed the significant wave height in 2-2.2 times. Considering the devastating effects of such extreme waves in an oceanic environment, the need to develop a deeper understanding of its underlying mechanisms become imperative. The ability to develop predictive tools and model extreme waves can afford a wide range of benefit to the offshore and marine field. While energy focusing in systems such as fiber optics is well established, the conditions leading to oceanic rogue waves are not well understood. Moreover, since extreme wave formation is essentially energy localization, understanding this phenomenon can facilitate energy harnessing as well.

Various mechanisms have been proposed to explain the rogue-wave phenomenon. Amongst the most popular theories are Modulational Instability and Nonlinear Focusing. The Benjamin-Feir instability (nonlinear modulational instability) [8] is considered to be the one of the reasons for rogue-wave occurrence, in which a uni-

form train of weak amplitude modulated wave loses energy to a small perturbation of other waves with nearly the same frequency and direction. The nonlinear Schrödinger equation (NSE) is a well-established approximate model based on the same assumptions as required for the derivation of the Benjamin-Feir theory: a narrow-banded spectrum of waves of moderate amplitude, propagating primarily in one direction in a dispersive medium with little or no dissipation. The NSE solution space can be viewed to have a nonlinear Fourier structure which is comprised of stable modes, unstable modes, and nonlinear interactions between them based on associated eigenvalues [3]. These unstable modes are potential ‘rogue-wave’ solutions. The eigenvalue space for unstable modes has not been completely investigated.

In shallow waters, however, modulational instability is absent. In this case, focusing of nonlinear wave packets with phase modulation is the main reason for freak wave formation. In deep waters, a combined effect of modulational instability and nonlinear focusing can lead to larger amplification of freak waves than suggested by amplitude modulation only [24], [48]. Other mechanisms include directional focusing [18], wave amplification due to blocking of water waves on the current (wave-current interaction), and atmospheric forcing [22].

Extreme waves have been reported in the context of many systems, ranging from ocean [10, 11] to optical fibers [50] to plasmas [36]. While infrequent, oceanic rogue waves are gaining attention due to their destructive nature. Limited descriptive quantitative data have been recorded [17]. Extreme waves occur infrequently enough that they are difficult to analyze, but yet, they cannot be ignored [6].

---

<sup>2</sup>Source: <http://www.opc.ncep.noaa.gov/perfectstorm/>



Figure 1.1: Rogue wave estimated at 60 feet moving away from ship after crashing into it a short time earlier. In the Gulf Stream off of Charleston, South Carolina, with light winds of 15 knots. <sup>2</sup>

## 1.2 Objectives

The principal objective of this work is to develop a better understanding of the complexities of extreme wave phenomena through analytical methods, mathematical models, and computational simulations. It is the expectation that the insights derived from this effort can be utilized towards the development of a predictive tool for forecasting purposes or the generation of a localized wave-field in a controlled environment for energy harnessing. Specific objectives broadly include the following:

- Study NSE as a model to describe modulational instability leading to rogue-wave behaviour. Explore the NSE main spectrum eigenvalue space ( $\lambda$  plane) to find new forms of rogue-wave solutions to the NSE with the aid of parallel GPGPU computations. Subsequently, predict initial conditions that lead to

energy localizations using the insight obtained from the NSE solution space.

- Perform Lagrangian based N-particle computational simulations in two dimensions to provide further insights into the mechanisms underlying rogue-wave formation. Smoothed Particle Hydrodynamics (SPH) is chosen as a numerical tool to study the hydrodynamics processes related to rogue-wave formation. The simulations are carried out using an in-house developed GPU-based massively parallel SPH code.

### 1.3 Basic Water Wave Mechanics

The differential form of the basic equation of fluid mechanics are as follows [12], [21]

$$\begin{aligned}
 \rho_t + \nabla \cdot (\rho \mathbf{v}) &= 0 && : \text{conservation of mass} \\
 \mathbf{v}_t + (\mathbf{v} \cdot \nabla) \mathbf{v} &= -\frac{1}{\rho} \nabla p + \mathbf{f} + \frac{\nu}{\rho} \nabla^2 \mathbf{v} && : \text{conseravtion of momentum}
 \end{aligned}
 \tag{1.1}$$

where  $\rho$  is the density,  $\mathbf{v}$  is the fluid velocity,  $p$  is the pressure,  $\mathbf{f}$  is the sum of body forces acting on the fluid, and  $\nu$  is the coefficient of viscosity called kinematic viscosity.

It is assumed that the depth of the fluid is  $h$  and it is bounded from below by a hard horizontal bed. The upper fluid surface is assumed to be free. The unperturbed free surface is at  $z = 0$ . When the upper surface is perturbed, there is vertical displacement  $\eta(x, y, t)$  of each point of the surface. Then, the free surface boundary condition is at  $z = \eta(x, y, t)$ . On the other lower solid surface, the normal

component of velocity should be zero. i.e, no flux is permitted at the bottom. That is,  $v_z = 0$  at  $z = -h$ .

The quantity  $\boldsymbol{\omega} = \nabla \times \mathbf{v}$  is called vorticity of the flow, and when  $\boldsymbol{\omega} = 0$  the flow is called irrotational. For an irrotational flow, the velocity is a potential field:  $\mathbf{v} = \nabla\phi$ , where  $\phi$  is the velocity potential. In addition, for gravity waves  $\mathbf{f} = -g\hat{k}$ , where  $g$  is the acceleration due to gravity. For small amplitude waves and irrotational flow approximation, equations (1.1) and boundary conditions(B.C.'s) can be reduced as follows:

$$\begin{aligned}
\nabla^2\phi &= 0; \quad -h < z < \eta(x, y, t) \rightarrow \text{from mass conservation} \\
\phi_t &= -\left[\frac{1}{2}(\phi_x^2 + \phi_y^2 + \phi_z^2) + \eta g\right]; \quad z = \eta(x, y, t) \rightarrow \text{from momentum conservation} \\
\phi_z &= \eta_x\phi_x + \eta_y\phi_y + \eta_z\phi_z; \quad z = \eta(x, y, t) \rightarrow \text{from B.C. on the free surface} \\
\phi_z &= 0; \quad z = -h \rightarrow \text{from bottom B.C.}
\end{aligned}
\tag{1.2}$$

Thus, the model equations becomes linear (Laplace equation) but the boundary conditions are nonlinear.

For small amplitudes (but long wave lengths) water waves, the nonlinear relationships from the equations (1.2) can be linearized. If the mean surface displacement and the mean velocity potential are small with respect to the wavelength and to wave period scales, then, the nonlinear terms in the boundary conditions can be neglected. After a Taylor series expansion of the small quantity  $\eta$  and keeping only the leading order terms, the free surface boundary condition can be written as condition on  $z = 0$ . Thus, one obtains the following simplified (and linear) equations

$$\begin{aligned}
\nabla^2\phi &= 0; \quad -h < z < 0, \\
\phi_t &= -g\phi_z; \quad \text{at } z = 0, \\
\phi_z &= 0; \quad \text{at } z = -h.
\end{aligned} \tag{1.3}$$

Next, it is assumed that the waves propagate in  $x$ -direction and are uniform in  $y$ -direction. Thus, this becomes a one-dimensional problem. Let the traveling wave solution to equation (1.3) have a frequency  $\omega$  and wavenumber  $k$ :

$$\phi(x, t) = \bar{A}(x, z) \sin(kx - \omega t) \tag{1.4}$$

The substitution of equation (1.4) in equation (1.3) leads to the following solutions for velocity potential  $\phi$  and for the surface displacement  $\eta$

$$\begin{aligned}
\eta &= A \cos(kx - \omega t), \\
\phi &= \omega A \frac{\cosh(k(z+h))}{k \sinh(kh)} \sin(kx - \omega t), \\
A &= 2 \frac{ak}{\omega} \exp(-kh) \sinh(kh)
\end{aligned} \tag{1.5}$$

where  $a$  is a constant of integration,  $A$ ,  $k$ , and  $\omega$  denote the wave amplitude, the wavenumber and wave frequency, respectively. The dispersion relationship for the small amplitude surface water waves is given by

$$\omega^2 = gk \tanh(kh) \tag{1.6}$$

Their phase velocity  $c$  and group velocity  $c_g$  are as follows

$$\begin{aligned}
c &= \omega/k = \sqrt{g/k \tanh(kh)}, \\
c_g &= d\omega/dk = \frac{c}{2} \left[ 1 + \frac{2kh}{2 \sinh(2kh)} \right]
\end{aligned} \tag{1.7}$$

The relationship  $R = \frac{\text{depth}}{\text{wavelength}} = h/\lambda = kh$  has two limiting cases:  $R \ll 1$  corresponding to shallow water and  $R \gg 1$  corresponding to deep water. For shallow-water waves, the dispersion relation (1.6) can be approximated by

$$\omega = k\sqrt{gh} \left[ 1 - \frac{k^2 h^2}{6} + \dots \right] = k\sqrt{gh} \quad (1.8)$$

Thus, for shallow water waves,  $c = c_g = \sqrt{gh}$ . Similarly, for deep water waves, the dispersion is given by

$$\omega = \sqrt{gk} \quad (1.9)$$

The phase velocity and group velocity are  $c = \sqrt{g/k}$  and  $c_g = \sqrt{g/4k}$ , respectively. That is, the phase velocity is twice the group velocity.

## 1.4 The Nonlinear Schrödinger Equation

A weakly nonlinear approximation to the nonlinear deep-water problem is the Stokes waves. However, these waves are unstable to modulation perturbations. This is called the *Benjamin-Feir Instability*, which will be discussed in the next section. The dispersion of the Stokes waves to second-order steepness is

$$\omega = \sqrt{gk \left( 1 + \frac{k^2 a^2}{2} \right)} \quad (1.10)$$

where  $a$  denotes the wave amplitude. Consider a slowly modulated Stokes wavetrain

$$\eta = \text{Re}[A(X, T) \exp(i(\omega_0 t - k_0 x))] \quad (1.11)$$

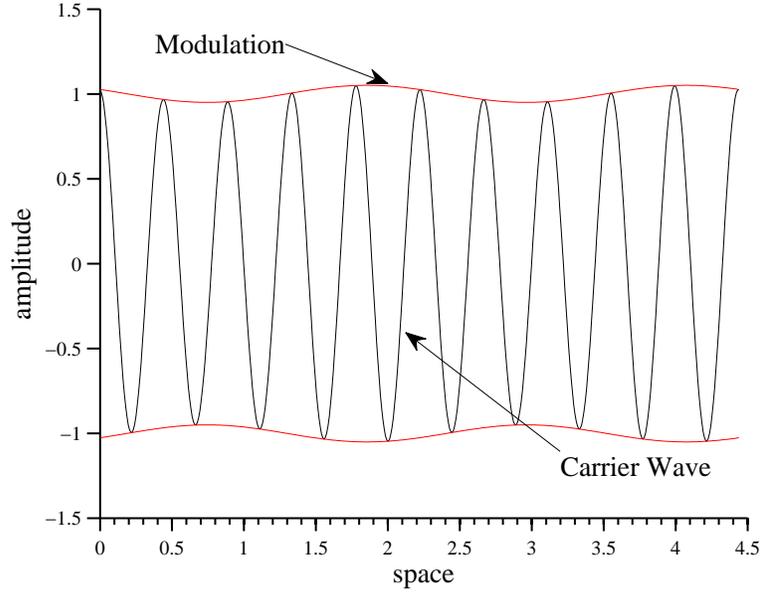


Figure 1.2: Small-amplitude modulation of a carrier wave.

where  $\omega_0$  and  $k_0$  are the frequency and wave number of Stokes carrier wave and  $A(X, T)$  is the modulation amplitude of the wavetrain as shown in Figure 1.2. In addition,  $X = \epsilon x$  and  $T = \epsilon t$  ( $\epsilon \ll 1$ ) are the slowly varying space and time variables, respectively. Physically,  $\epsilon := Ak_0$  is the steepness of the wave and is assumed to be small. Now consider a Taylor series expansion around the wavenumber  $k_0$  and the amplitude  $A_0 = A(0, 0) \dots$ . The dispersion relation of the carrier Stokes wave is

$$\omega = \sqrt{gk(1 + k^2|A|^2)} \quad (1.12)$$

where  $|A|$  is the amplitude of the Stokes wave (and the amplitude of the envelope). The Taylor series expansion about the wavenumber  $k_0$  of the carrier wave and about the envelope  $A = A_0 = 0$  [43].

$$\omega = \omega_0 + \frac{\partial \omega}{\partial k}(k - k_0) + \frac{1}{2} \frac{\partial^2 \omega}{\partial k^2}(k - k_0)^2 + \frac{\partial \omega}{\partial |A|^2}(|A|^2 - |A_0|^2) \quad (1.13)$$

Let  $\Omega = \omega - \omega_0$  and  $K = k - k_0$ . In addition, from equation (1.12),

$$\begin{aligned}\left.\frac{\partial\omega}{\partial k}\right|_{k=k_0} &= c_g = \frac{\omega_0}{2k_0}, \\ \left.\frac{\partial^2\omega}{\partial k^2}\right|_{k=k_0} &= -\frac{\omega_0}{8k_0^2}, \\ \left.\frac{\partial\omega}{\partial A^2}\right|_{A_0=0} &= \frac{1}{2}\omega_0 k_0^2\end{aligned}$$

Then, from equation (1.13)

$$\Omega = c_g K - \frac{\omega_0}{16k_0^2} K^2 + \frac{1}{2}\omega_0 k_0^2 |A|^2 \quad (1.14)$$

The Fourier and inverse Fourier transforms of the envelope function are

$$\begin{aligned}A(K, \Omega) &= \mathcal{F}[A(X, T)] = \int_{-\infty}^{\infty} dX dT A(X, T) \exp[i(\Omega T - KX)], \\ A(X, T) &= \mathcal{F}^{-1}[A(K, \Omega)] = \left(\frac{1}{2\pi}\right)^2 \int_{-\infty}^{\infty} dX dT A(K, \Omega) \exp[-i(\Omega T - KX)]\end{aligned} \quad (1.15)$$

From equations (1.15),

$$\begin{aligned}\frac{\partial A}{\partial X} &= iK \mathcal{F}^{-1}[A(K, \Omega)], \\ \frac{\partial A}{\partial t} &= i\Omega \mathcal{F}^{-1}[A(K, \Omega)]\end{aligned} \quad (1.16)$$

$\Omega$  and  $K$  are of order  $\epsilon$ . Then from equation (1.16),

$$\begin{aligned}K &= -i\epsilon \frac{\partial}{\partial X}, \\ \Omega &= i\epsilon \frac{\partial}{\partial T}\end{aligned} \quad (1.17)$$

The substitution of the relationships from equation (1.17) into equation (1.14) and application of the resulting operator equation to the envelope amplitude  $A$  leads to the nonlinear Schrödinger equation for the evolution of the amplitude of the envelope of the wavetrain ( $\epsilon$  is incorporated in  $T$  and  $X$  by appropriate rescaling). In order

to maintain uniformity throughout this document,  $A, X, T$  have been replaced by  $\psi, x, t$ . Thus, the NSE can be written as

$$i\left(\frac{\partial\psi}{\partial t} + \frac{\omega_0}{2k_0}\frac{\partial\psi}{\partial x}\right) - \frac{\omega_0}{8k_0^2}\frac{\partial^2\psi}{\partial x^2} - \frac{1}{2}\omega_0k_0^2|\psi|^2\psi = 0 \quad (1.18)$$

## 1.5 Benjamin-Feir Instability

The normalized form of equation (1.18) in a frame of reference moving with linear group velocity  $c_g$  is

$$i\psi_t - \left(\frac{\omega_0}{8k_0^2}\right)\psi_{xx} - \frac{1}{2}\omega_0k_0^2|\psi|^2\psi = 0 \quad (1.19)$$

One of the simplest solutions of equation (1.19) is given by

$$A(t) = a_0\exp\left(-\frac{1}{2}i\omega_0k_0^2a_0^2t\right) \quad (1.20)$$

where  $a_0$  is a constant, the amplitude of the carrier wave. This essentially represents the fundamental component of the Stokes wave. Next, consider a perturbation of equation (1.20) in the form

$$a(x, t) = A(t)[1 + B(x, t)] \quad (1.21)$$

where  $B(x, t)$  is the perturbation function. On substituting this result in equation (1.20), one obtains

$$\begin{aligned} i(1 + B)A_t + iAB_t - \left(\frac{\omega_0}{8k_0^2}\right)AB_{xx} \\ = \frac{1}{2}\omega_0k_0^2a_0^2t[(1 + B) + BB^*(1 + B) + (B + B^*)B \\ + (B + B^*)]A \end{aligned} \quad (1.22)$$

where  $B^*(x, t)$  is the complex conjugate of the perturbed quantity  $B(x, t)$ . Neglecting squares of  $B$ , equation (1.22) reduces to

$$iB_t - \left( \frac{\omega_0}{8k_0^2} \right) B_{xx} = \frac{1}{2} \omega_0 k_0^2 a_0^2 (B + B^*). \quad (1.23)$$

The perturbed quantity  $B(x, t)$  can be expressed in the form

$$B(x, t) = B_1 \exp(\Omega t + iKx) + B_2 \exp(\Omega^* t - iKx) \quad (1.24)$$

where  $B_1$  and  $B_2$  are complex constants,  $K$  is a real wavenumber and  $\Omega$  is a growth rate to be determined.  $\Omega$  is also the modulational frequency. On substituting the solution for  $B$  in equation (1.23), one obtains a pair of coupled equations:

$$\left( i\Omega + \frac{\omega_0 K^2}{8k_0^2} \right) B_1 - \frac{1}{2} \omega_0 k_0^2 a_0^2 (B_1 + B_2^*) = 0, \quad (1.25)$$

$$\left( i\Omega^* + \frac{\omega_0 K^2}{8k_0^2} \right) B_2 - \frac{1}{2} \omega_0 k_0^2 a_0^2 (B_1^* + B_2) = 0 \quad (1.26)$$

The complex conjugate of equation (1.26) can be transformed into

$$\left( -i\Omega + \frac{\omega_0 K^2}{8k_0^2} \right) B_2^* - \frac{1}{2} \omega_0 k_0^2 a_0^2 (B_1 + B_2^*) = 0 \quad (1.27)$$

The pair of linear homogeneous equations (1.25) and (1.27) for  $B_1$  and  $B_2^*$  admits a nontrivial eigenvalue for  $\Omega$  provided

$$\begin{vmatrix} i\Omega + \frac{\omega_0 K^2}{8k_0^2} - \frac{1}{2} \omega_0 k_0^2 a_0^2 & -\frac{1}{2} \omega_0 k_0^2 a_0^2 \\ -\frac{1}{2} \omega_0 k_0^2 a_0^2 & -i\Omega + \frac{\omega_0 K^2}{8k_0^2} - \frac{1}{2} \omega_0 k_0^2 a_0^2 \end{vmatrix} = 0 \quad (1.28)$$

which is equivalent to

$$\Omega^2 = \frac{1}{2} \left( \frac{\omega_0 K}{2k_0} \right)^2 \left( k_0^2 a_0^2 - \frac{K^2}{8k_0^2} \right) \quad (1.29)$$

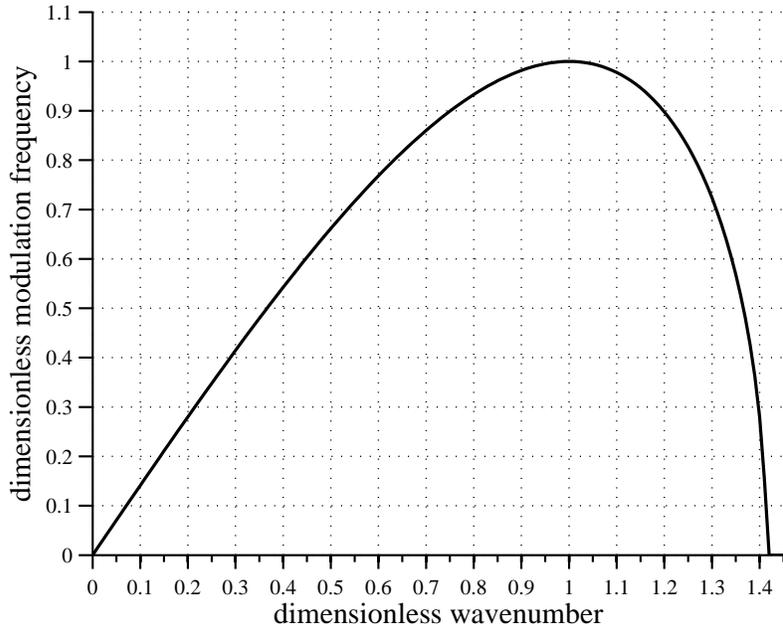


Figure 1.3: Instability diagram for small-amplitude modulations for the NSE.

The growth rate  $\Omega$  is purely imaginary or real (and positive) depending whether  $K^2 > 8k_0^4 a_0^2$  or  $K^2 < 8k_0^4 a_0^2$ . The former case represents a wave solution for  $B$  and the latter corresponds to the *Benjamin-Feir* [8] or *modulational* instability with criterion given as

$$0 < K < 2\sqrt{2}k_0^2 a_0 \quad (1.30)$$

Equation (1.29) can be written as

$$\Omega = \omega_0 k_0^2 a_0^2 \left( \frac{K}{2\sqrt{2}k_0^2 a_0} \right) \sqrt{1 - \left( \frac{K}{2\sqrt{2}k_0^2 a_0} \right)^2} \quad (1.31)$$

Let  $2\Omega/\omega_0 k_0^2 a_0^2$  be the dimensionless modulation frequency  $\tilde{\Omega}$  and  $K/2k_0^2 a_0$  be the dimensionless wavenumber  $\tilde{K}$ . Then equation (1.31) can be written as

$$\tilde{\Omega} = \sqrt{2}\tilde{K} \sqrt{1 - \frac{\tilde{K}^2}{2}} \quad (1.32)$$

As shown in Figure 1.3, the maximum instability occurs at  $K = 2k_0^2 a_0$  that cor-

responds to the maximum growth rate of  $\Omega_{max} = \frac{1}{2}\omega_0 k_0^2 a_0^2$ . Thus, it can be concluded that Stokes waves are definitely unstable to modulation perturbations in the Benjamin-Feir range.

## 1.6 Periodic Spectral Theory

Equation (1.18) can be rescaled to a non-dimensional form. The details of this rescaling have been presented in Section 2.1.7. The scaled NSE is given by

$$iu_t - u_{xx} + 2\sigma|u|^2u = 0 \quad (1.33)$$

The deepwater (known as "focusing") case corresponds to  $\sigma = -1$ . The shallow water (known as "defocusing") case corresponds to  $\sigma = 1$ . Equation (1.33) can be divided into a spatial scattering problem and a time dependence problem. The solution space of focusing NSE with periodic boundary conditions can be viewed to have a nonlinear Fourier structure which is comprised of stable, unstable modes and nonlinear interactions between them based on the eigenvalues of the spatial problem. The unstable modes are unstable in a Benjamin-Feir sense and correspond to what one calls 'rogue waves'. Details of nonlinear Fourier structure of the NSE solution space are presented in Section 2.1.2. Periodic boundary conditions are assumed so that  $u(x, t) = u(x + L, t)$  for  $0 \leq x \leq L$ . The NSE can be solved by using a method called the *Inverse Scattering Transform Method* (IST). This method was first devised for the infinite line case by Zakharov and Shabat [46] and then later extended to periodic boundary conditions [52]. A brief overview of the IST method for the periodic boundary condition (or periodic spectral theory) is presented in this

section.

The direct problem of periodic spectral theory is that of constructing the spectral data of certain linear operators with periodic coefficients; that is, the determination of the spectrum of this operator and of the associated eigenfunctions. The inverse problem of periodic spectral theory is the problem of the reconstruction of such an operator (and thus its coefficients) from given spectral data. The history of periodic spectral theory starts with the investigations of Sturm and Liouville on the eigenvalues of certain differential equations of second order with given boundary conditions, now referred to as *Sturm-Liouville* theory [19]. Sturm and Liouville examined independently different aspects of this problem, such as the asymptotics of eigenvalues, different comparison theorems on the solutions of similar equations with different coefficients, and theorems on the zeros of eigenfunctions. For the class of equations Sturm and Liouville considered, these results imply the existence of an infinite sequence of real, increasing eigenvalues, and orthogonality of eigenfunctions corresponding to different eigenvalues. Although their investigations did not as such deal with periodic spectral theory, many of their results carry over to this case. Consider an ordinary differential operator of order  $n$

$$L = q_n(x) \frac{d^n}{dx^n} + q_{n-1}(x) \frac{d^{n-1}}{dx^{n-1}} + \dots + q_1(x) \frac{d}{dx} + q_0(x)$$

where the coefficients  $q_j(x)$ ,  $j = 0, \dots, n$  are periodic functions of  $x$ , sharing a common period:  $q_j(x + L) = q_j(x)$ ,  $j = 0, \dots, n$  and  $q_{n-1}(x) = 0$ . They are referred to as potentials. Using this operator  $L$ , the following differential equation is defined

$$L\psi = \lambda\psi, \tag{1.34}$$

This is similar to the spectral problem of the NSE shown in Section 2.1 equation (2.1). Thus, the direct spectral problem is the problem of

1. determining the set of all  $\lambda \in C$  for which this differential equation has at least one bounded solution, and
2. for each such  $\lambda$ , the determination of all bounded solutions

Most of the times, one is interested in the periodic solutions of  $\psi : \psi(x+L) = \psi(x)$  or anti-periodic solutions  $\psi(x+L) = -\psi(x)$ . These and other choices lead to spectra that are subsets of the spectrum as obtained without making these choices.

One approach to solve the direct spectral problem is *Floquet Theory*. Rewriting equation (1.34) as a first-order linear system ( as in equation (2.6)), one gets:

$$\psi' = X(x, \lambda)\psi, \quad X(x+L, \lambda) = X(x, \lambda) \quad (1.35)$$

It follows from  $q_{n-1} = 0$  that  $\text{tr}X(x, \lambda) = 0$ . Define the monodromy matrix of this system as  $M(x_0, \lambda) = (x_0 + L, x_0, \lambda)$ , where  $\psi(x, x_0, \lambda)$  is a fundamental matrix of system (1.35) such that  $\psi(x_0, x_0, \lambda)$  is the identity matrix. Thus,  $M(x_0, \lambda)$  is the operator of translating  $x$  by  $L$ :  $M(x_0, \lambda)\psi(x) = \psi(x+L)$ . This operation commutes with  $d/dx$ , since  $X(x, \lambda)$  is periodic in  $x$  with period  $L$ . Thus, the system (1.35) has a set of solutions  $\phi(x)$  which are also eigenvectors of  $M(x_0, \lambda)$ . These solutions (as mentioned in section 2.1) are known as Bloch functions or Floquet functions. If the eigenvalue of  $M(x_0, \lambda)$  for any Bloch function has magnitude greater than one, than this Bloch function is unbounded as  $x \rightarrow +\infty$  or  $x \rightarrow -\infty$ . Thus, the spectrum of the system (1.35) is the set of all  $\lambda$  such that at least one eigenvalue

of  $M(x_0, \lambda)$  has magnitude one. This will be discussed in detail in section 3.2. The periodicity of  $X(x, \lambda) = X(x + L, \lambda)$  and the requirement  $\text{tr}X(x, \lambda) = 0$  guarantee that the spectrum is independent of the choice of  $x_0$ .

$$-\psi'' + q(x)\psi = \lambda\psi, \quad q(x + T) = q(x) \quad (1.36)$$

The above equation is known as the Hill's equation or the time-dependent Schrödinger equation. Its spectrum is bounded from below. It is a collection of intervals such that the length of the separating gaps between intervals  $\rightarrow 0$  as  $\lambda \rightarrow \infty$ . After using Floquet theory, the condition for  $\lambda$  to be in the spectrum is found to be  $|\text{tr}M(x_0, \lambda)| \leq 2$ . The endpoints of the intervals are given by  $|\text{tr}M(x_0, \lambda)| = 2$ . Since equation (1.36) is a second-order equation, there are two linearly independent Bloch functions. A similar result will be derived for the NSE spectral eigenvalue problem in the upcoming sections. In this case,  $q(x)$  is the potential of the system, and  $\lambda$  plays the role of energy. In equation (2.9),  $u$  is the potential and  $\lambda$  are the eigenvalues. It may be noted that the intervals constituting the spectrum are known as allowed (energy)bands and the gaps between them as forbidden (energy) bands. The inverse periodic spectral problem is the reconstruction of the potential given by the spectral data discussed in Section 2.1.4.

## 1.7 Focusing as a Mechanism for Rogue-wave Formation

Amongst the most popular theories proposed to explain rogue-wave formation, Modulational Instability and Focusing are the most popular ones. In deep waters, a combined effect of both these mechanisms might lead to an energy localization.

However, in shallow waters modulational instability is absent. In this case, focusing of nonlinear wave packets with phase modulation is the main reason for freak or extreme wave formation. Other mechanisms include directional focusing [18], wave-current interaction and atmospheric forcing [22].

### 1.7.1 Linear Focusing

From the dispersion relation, it is evident that the phase velocities and the group velocities depend on a frequencies in a way that in a wave group, the long waves (low frequency) lead and short waves (high frequency) lag. If during the initial moment the short waves with small group velocities are located in front of the long waves having large group velocities, then in the phase development, a significant focusing of the wave energy can occur only if all the quasi-monochromatic groups merge at a fixed location. This is spatio-temporal or dispersive focusing as shown in Figure 1.4. Geometric focusing is when multiple wave fronts arrive (superimpose) at a point from different directions. An example of geometric focusing is shown in Figure 1.5. Locations of focal points for various curvatures of focusing cylindrical waves are shown.

### 1.7.2 Nonlinear Dispersive Focusing

Modulation instability does not occur with shallow water waves. Hence, nonlinear focusing phenomenon is the predominant cause of freak wave formation in shallow water. Focusing occurs in a significantly phase modulated wave packet. In

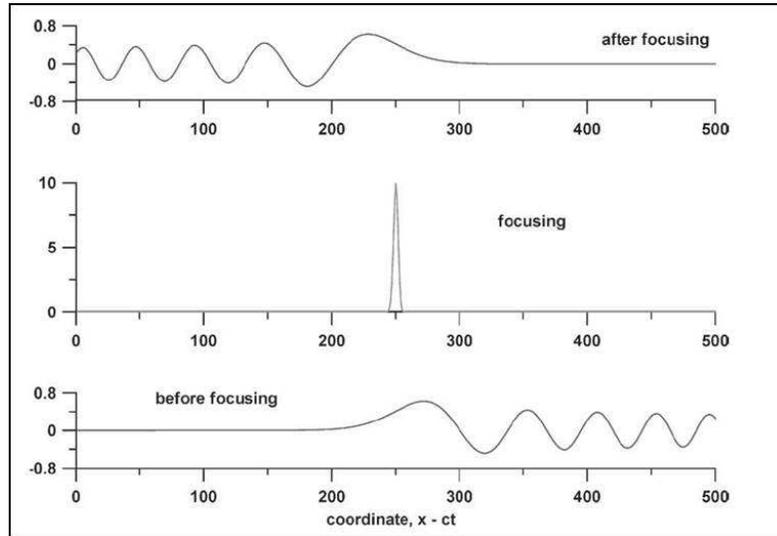


Figure 1.4: Formation of the freak wave of Gaussian form in shallow water (Source: [22]).

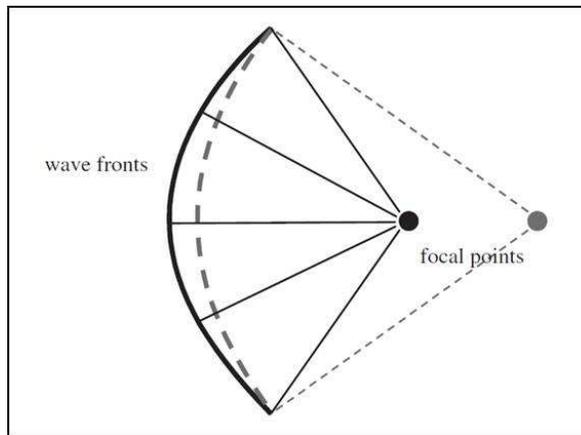


Figure 1.5: Formation of the freak wave due to geometric focusing (Source: [23]).

shallow water, it is studied based on Korteweg-de Vries (KdV) equation:

$$\frac{\partial \eta}{\partial t} + c \left( 1 + \frac{3\eta}{2h} \right) \frac{\partial \eta}{\partial x} + \frac{ch^2}{6} \frac{\partial^3 \eta}{\partial x^3} = 0 \quad (1.37)$$

where  $c$  is the wave celerity and  $h$  is the water depth. The KdV equation is invariant with respect to the reversal of time and abscissa which implies that we can choose the expected form of the freak wave as the initial condition for the KdV equation and calculate the surface elevation for anytime  $t$ . Subsequently, one can reverse its evolution that should lead to the freak wave as shown in Figure 1.6. E. Pelinovsky *et al.* [42] proved this by using the dimensionless KdV equation:

$$\begin{aligned} \frac{\partial \zeta}{\partial \tau} + \frac{3}{2} \zeta \frac{\partial \zeta}{\partial y} + \frac{1}{6} \frac{\partial^3 \zeta}{\partial y^3} &= 0 \\ \zeta = \frac{\eta}{h}, \quad y = \frac{x - ct}{h}, \quad \tau = \frac{ct}{h} \end{aligned} \quad (1.38)$$

The initial condition representing the freak wave is given by [42]

$$\zeta(y, 0) = A_0 \exp \left[ - \left( \frac{x - 5L/6}{d} \right)^2 \right] \quad (1.39)$$

where the amplitude  $A_0$  and characteristic width  $d$  are parameters varied in numerical experiments.

Nonlinear focusing phenomenon in deep water is similar to that in shallow water within the framework of NSE due to its invariance with respect to space and time. In deep water, a rogue-wave formation can be due to a combination of modulational instability and focusing.

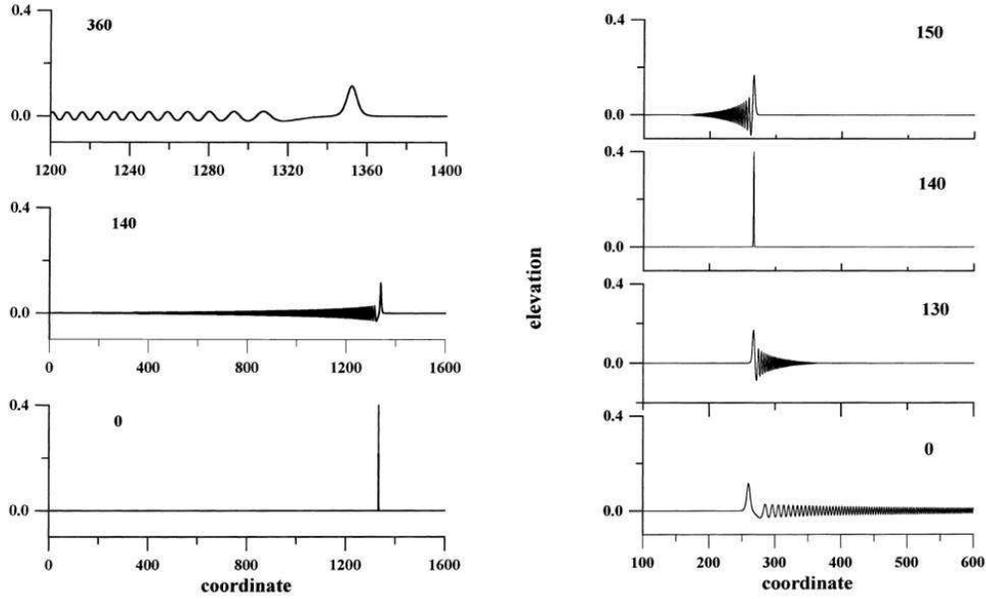


Figure 1.6: The process of the freak wave formation from the nonlinear-dispersive wavetrain for different times (Source: [42]).

### 1.7.3 Nonlinear Directional Focusing

Fochesato *et al.* [18] simulated and analyzed three-dimensional (3D) directional wave focusing phenomenon leading to the generation of rogue waves. These authors generated extreme waves via realistic fully nonlinear 3D simulations in a numerical wave tank, by specifying the motion of a snake wave maker shown in Figure 1.7. A two-dimensional (2D) longitudinal cross-section through extreme wave crest looked similar to characteristic rogue-wave shape (crests followed by holes) as shown in Figure 1.8.

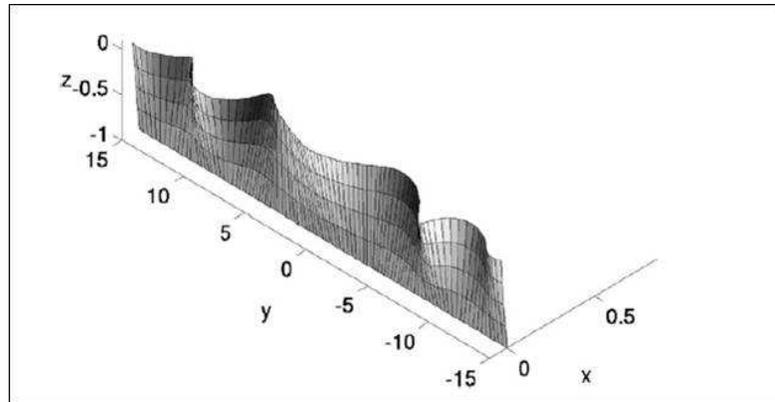


Figure 1.7: Snake wave maker used in directional focusing (Source: [18]).

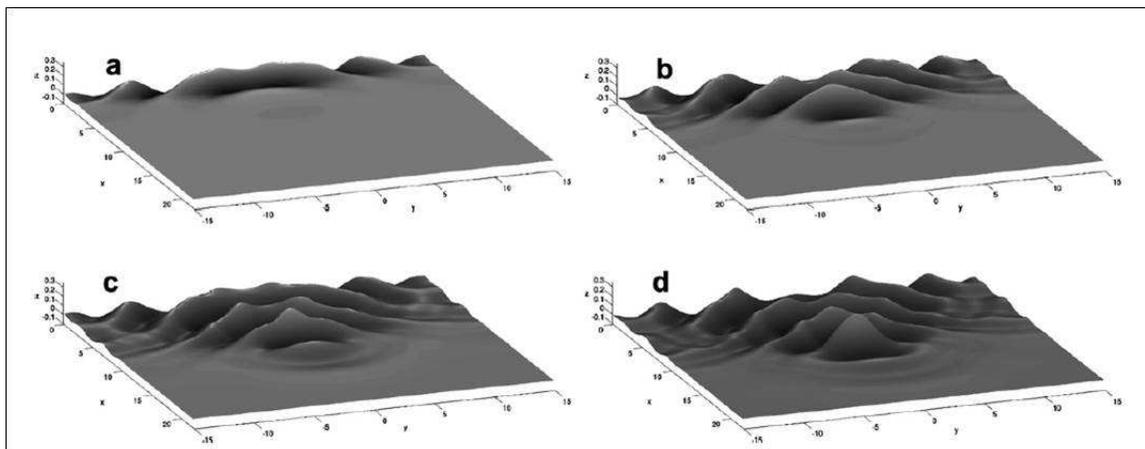


Figure 1.8: Rogue waves simulated via directional focusing (Source: [18]).

## 1.8 Outline

The rest of the dissertation is organized in the following manner. In Chapter 2, the author deals with the analytical and computational study of the nonlinear Schrödinger equation and its associated “rogue-wave” solutions. To begin with, a brief literature review has been presented. In the following section, existing analytical solution methodology to periodic NSE has been described. The existing mathematical formulations have been used to develop a new predictor-corrector based algorithm to determine parameters governing periodic Riemann theta function rogue-wave solutions to the nonlinear Schrödinger equation. A detailed explanation of this novel algorithm has been presented in the subsequent sections. By using the predictor-corrector algorithm, new physical forms of rogue-wave solutions to the NSE have been generated. Several interesting features of the predictive map and the generated rogue-wave solutions have been presented. The solutions then are rescaled to match the length scales of waves generated in a wave tank. Based on these new rogue-wave like solutions and the details of physical scaling, it is believed that the presented framework could serve as a basis for experimental investigations into a variety of rogue waves as well localizations in wave fields.

In Chapter 3, the author deals with computational simulation study of extreme wave localizations. First, a Lagrangian based N-particle method, namely, Smoothed Particle Hydrodynamics (SPH) is described. Its application to free surface flows is discussed. The equations governing the SPH simulation and details of the parallel CUDA implementation is then described. The author’s formulation

closely follows the formulation presented in an earlier work [54]. In the following section, an improved smoothing kernel applied to the pressure forces is described. This smoothing kernel prevents particle overlap more successfully than the kernel described previously in reference [54]. Subsequently, results from the current SPH formulation are validated through a comparison to a classic “dam break” simulation from prior studies. The robustness of the dispersion relation is then evaluated over a quiescent surface through numerical experiments. The issue of wave attenuation due to algorithmic dissipation in SPH is then addressed and a possible solution has been suggested. This is followed up with results obtained from various case studies on standing waves and directional focusing in 1+1 dimension. Results of free surface gravity wave simulations carried out to demonstrate dispersive focusing are presented in the next section. Finally, simulation case studies have been presented to realize modulational instability or self focusing a numerical wave tank.

Appendices that provide some additional technical details and references are included at the end.

## Chapter 2: The Nonlinear Schrödinger Equation and Rogue-wave Solutions

### 2.1 Literature Review

In this work, mainly modulational instability has been investigated as a mechanism for formation of rogue waves. The periodic nonlinear Schrödinger equation (NSE) has been used to model extreme waves in many domains. Several families of analytical solutions have been determined for the NSE. Shabat and Zakharov [46] were the first to use the Inverse Scattering Transform (IST) to develop analytic solutions of the NSE with infinite line boundary conditions. Analytical solutions to the periodic NSE were presented by Tracy [52]. The NSE solution space can be viewed as having a nonlinear Fourier structure, which is comprised of stable and unstable modes. Nonlinear interactions can occur between these modes based on associated eigenvalues [39]. The unstable modes are potential “rogue-wave” solutions. Several solutions to the NSE are already known, and motivate the search for more solutions. The Peregrine breather is a well-known extreme wave solution [15]. Akhmediev *et al.* [2,3] have also determined a family of rational solutions to the NSE. The rational solutions are determined by taking a modified Darboux transform of a specially

chosen seed solution. They have successfully tested for the presence of rational solutions in a randomly perturbed wave field. A system governed by the NSE was excited with a plane wave with random perturbations. Regions of large amplitudes were identified and found to match almost identically to the envelope predicted by the rational solutions. It may be noted that the Peregrine solution is a first order rational solution to the NSE. Ma and Ablowitz [28] have provided a solution methodology for obtaining spectral solutions for periodic boundary conditions for both the focusing and defocusing cases. Given the current state of understanding of solutions of the NSE, as of yet unknown rogue-wave solutions may be critical to further the understanding of instabilities and extreme behaviors of many systems.

Other contributions to determining analytical solutions of the NSE include those due to Akhmediev and Korneev [4], who determined a family of single parameter solutions. Based on finite gap integration, Smirnov [49] constructed a family of two-gap solutions and derived conditions under which they behave as rogue waves.

A review of nonlinear optical waves, including exact solutions to the nonlinear Schrödinger equation, nonlinear interference, and soliton behavior in dispersive media is available in the book by Akhmediev and Ankiewicz [1]. Different groups have determined other families of rogue-wave type solutions to the standard NSE. Notably, Akhmediev, Soto-Crespo, and Ankiewicz [5] identify the interference of Akhmediev breathers (ABs) as leading to a type of rogue-wave solution. They show that properly phased AB collisions can result in rogue waves and suggest it as a method to explain and possibly provoke rogue waves in optical fibers [50].

Several groups have verified the analytically predicted solutions with experi-

mental results. The Peregrine breather, which is a limiting form of several families of analytical rogue-wave solutions, has been studied in a fiber optic cable [25]. In turn, several analytically predicted extreme waves have been demonstrated experimentally in optical fibers [50] and water wave tanks [11], [10]. In each case, the observed rogue waves have been modeled after solutions to the NSE. Finally, Dysthe [16] has introduced a higher order approximation to the wave equation, and this equation is called the Dysthe equation. This equation is considered to provide a more accurate model of extreme wave behavior under certain conditions. A comprehensive review of past contributions and the state of the art related to rogue waves can be found in several review papers (e.g., [22], [56], and [44]).

## 2.2 Analytical Solutions to the NSE

### 2.2.1 Lax's Generalization

Peter Lax, in 1968, paved the way to generalize the IST technique as a method for solving other partial differential equations by dividing the PDE into a spectral and a temporal problem. Consider operators  $L$  and  $A$ , where  $L$  is the operator of the spectral problem and  $A$  is the operator governing the the associated time evolution of the eigenfunctions.

$$Lv = \lambda v, \tag{2.1}$$

$$v_t = Av, \tag{2.2}$$

Now taking  $\partial/\partial t$  of equation (2.1), leads to

$$L_t v + L v_t = \lambda_t v + \lambda v_t$$

Hence, using equation (2.2),

$$\begin{aligned} L_t v + L A v &= \lambda_t v + \lambda A v \\ &= \lambda_t v + A \lambda v \\ &= \lambda_t v + A L v \end{aligned}$$

Thus, it is obtained that

$$[L_t + (L A - A L)] v = \lambda_t v$$

and hence in order to solve for nontrivial eigenfunctions  $v(x, t)$

$$L_t + [L, A] = 0 \tag{2.3}$$

where

$$[L, A] := L A - A L$$

if and only if  $\lambda_t = 0$ . equation (2.3) is called the *Lax's Equation* and the operators are called the *Lax pair* and are said to be *compatible* if they satisfy the Lax's equation.

The Lax pair for the NLS equation is given by

$$L := \begin{bmatrix} i\partial_x & -iu \\ i\sigma u^* & -i\partial_x \end{bmatrix} \tag{2.4}$$

$$A := \begin{bmatrix} 2i\lambda^2 + i\sigma u u^* & -2\lambda u - iu_x \\ -2\lambda\sigma u^* + i\sigma u_x^* & -2i\lambda^2 - i\sigma u u^* \end{bmatrix} \tag{2.5}$$

According to equation (2.1), the spectral operator  $L$  follows

$$L\phi = \lambda\phi$$

or

$$\phi_x = Q\phi \tag{2.6}$$

where

$$Q = \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix} \tag{2.7}$$

According to equation (2.2), the temporal operator follows

$$\phi_t = A\phi$$

where  $\phi$  is a two component eigenfunction also known as the **Bloch eigenfunction**

$$\phi = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}$$

It may be noted that equation (2.3) can also be written as

$$Q_t - A_x + [Q, A] = 0 \tag{2.8}$$

Substituting  $Q$  from equation (2.7) and  $A$  from equation (2.5) into equation (2.8), one gets back the scaled NSE which is equation (1.33). The eigenvalue problem

$$\phi_x = \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix} \phi \tag{2.9}$$

can be solved for the *main spectrum eigenvalues*,  $\lambda$ , using Floquet Analysis.

## 2.2.2 Nonlinear Fourier Structure of the NSE Solution Space

According to Osborne [41], all solutions of the NLS can be decomposed into two fundamentally distinct kinds of wave modes, stable and unstable. The stable modes are modulationally stable to perturbations of their envelope functions. The unstable modes are instead modulationally unstable in the Benjamin-Feir sense. Formally, the following nonlinear Fourier decomposition holds for all solutions of the NLS equation:

$$\psi(x, t) = \psi_{unstable}(x, t) + \psi_{stable}(x, t) + \psi_{nonlinearinteractions}(x, t)$$

In linear fourier analysis, the sine wave components are characterized by the parameters consisting of amplitude, wavenumber, frequency and phase and therefore, with all the varieties of these parameters possible there are effectively an infinite number of kinds of linear fourier components, all being sine waves. In the nonlinear theory for the periodic NSE the same is true, except that the shape of the spectral components is different depending upon the parameters. Not only is the shape of the nonlinear fourier components different, but there are also two types: stable components (or modes) that are classical Stokes waves and unstable components that are ‘breather’ or ‘rogue-wave’ solutions of the NSE. There can be infinite number of nonlinear fourier components of the periodic IST each characterized by the five parameter family  $(a, \lambda_R, \lambda_I, |\epsilon|, \theta)$  as shown in Figure 2.1. On the  $\lambda$ -plane,  $\lambda = \lambda_R + i\lambda_I$ , correspond to the main spectrum eigenvalues evaluated from the eigenvalue problem shown in equation (2.9). It is the centroid of a non-degenerate

pair of eigenvalues. The two points of this non-degenerate pair are given by  $\lambda \pm \epsilon$ , where  $\epsilon$  is a complex number,  $\epsilon = |\epsilon|\exp(i\theta)$ . To have a degree of freedom these eigenvalues must be connected by a spine, a curve that connects the two simple eigenvalues. These spines, shown in Figure 2.1, are curves in the complex plane with values of  $\lambda$  which ensure that the Bloch eigenfunctions  $\phi$  in equation (2.9) are stable. Further details about spines can be found in reference [40]. When two points of main spectrum are connected by a spine, the combination of the spectral information is called a ‘nonlinear mode’. There are two kinds of nonlinear modes:

1. When two points of the spectrum are connected by spine that crosses the real axis, one has a ‘stable mode’ or ‘stable Stokes mode’.
2. When two points of the spectrum are connected by a spine that does not cross the real axis, one has an ‘unstable’ or ‘rogue’ mode in the spectrum.

$\lambda_I$  axis can be physically characterized as a spectral amplitude while  $\lambda_R$  axis corresponds to spatial or temporal frequency for the NSE. When  $0 < \lambda < ia$  on the imaginary axis,  $|\epsilon|$  is the actual (small - amplitude) modulation amplitude. When  $\lambda_I > a$ , no small initial modulation generates the motion of the wave; only large-amplitude modulations occur. Also, when  $\theta = 0$  the unstable mode is denoted by a ‘cross state’. When  $\theta = \pi/2$  the unstable mode is ‘slot state’. Any other value of  $\theta$  corresponds to a ‘slant state’.

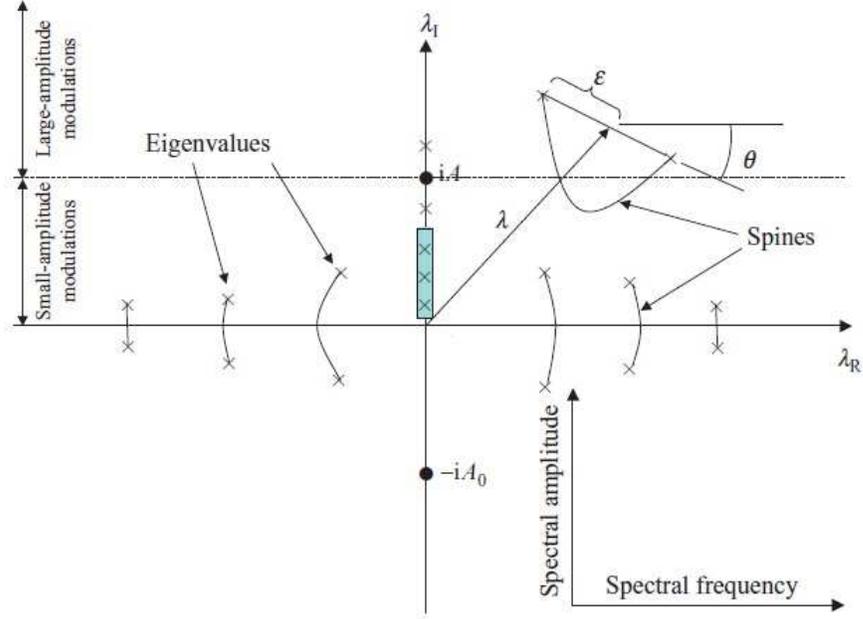


Figure 2.1:  $\lambda$  plane showing main spectrum eigenvalues with a detailed view of  $\epsilon$  and  $\theta$  for a particular choice of  $(\lambda_R, \lambda_I)$  (Source: [40]).

## 2.2.3 Floquet Theory

### 2.2.3.1 Floquet Theory Applied to the Spectral Problem of NSE

The mathematical proof and details of Floquet theory are discussed in Appendix A. Here, Floquet analysis is applied to the spectral eigenvalue problem of the NSE (2.9). Rewriting the equation, one has

$$\Phi_x = \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix} \Phi \quad (2.10)$$

The matrix  $Q(\lambda)$ , equivalent to  $A(t)$  in equation (A.8) is a  $2 \times 2$  with  $Q(x, \lambda) = Q(x + L, \lambda)$  (since  $u$  in equation (1.33) has periodic boundary conditions)

$$Q(\lambda) = \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix}$$

It can be noted that here  $\text{tr}Q = 0$ . Next, the fundamental matrix  $\Phi(t)$  for equation (2.10) is formed such that  $\Phi(0) = I$ . Hence

$$\Phi = \begin{bmatrix} \phi_1^1 & \phi_1^2 \\ \phi_2^1 & \phi_2^2 \end{bmatrix} \quad (2.11)$$

where  $\phi^1$  and  $\phi^2$  are linearly independent solutions of equation (2.10) such that

$$\begin{aligned} \phi_1^1 &= 1, & \phi_1^2 &= 0 \\ \phi_2^1 &= 0, & \phi_2^2 &= 1 \end{aligned} \quad (2.12)$$

The **monodromy** matrix  $M$  given by equation (A.12), where here  $\Phi(0) = I$ , so that

$$M = \begin{bmatrix} \phi_1^1(L) & \phi_1^2(L) \\ \phi_2^1(L) & \phi_2^2(L) \end{bmatrix} \quad (2.13)$$

It can be noted that the  $\text{tr}Q = 0$ , and as a result of equation (A.11),

$$\det M = 1 \quad (2.14)$$

The floquet multipliers  $\rho$  are the eigenvalues of  $M$  and, hence, are given by

$$\rho^2 - 2\left[\frac{1}{2}\text{tr}M\right]\rho + \det M = 0 \quad (2.15)$$

Let  $\frac{1}{2}\text{tr}M = \beta$ . Thus, the eigenvalues  $\rho_{1,2}$  are functions of a single parameter  $\beta$  and these eigenvalues are given by

$$\rho_{1,2} = \beta \pm \sqrt{\beta^2 - 1} \quad (2.16)$$

From equation (2.15),

$$\rho_1\rho_2 = 1, \quad \rho_1 + \rho_2 = 2\beta \quad (2.17)$$

The floquet exponents are  $\mu_{1,2}$  where  $\rho_{1,2} = e^{\mu_{1,2}L}$ , and consequent to equation (2.17),

$$\mu_1 + \mu_2 = 0, \quad \cosh \mu_1 L = \beta \quad (2.18)$$

### 2.2.3.2 Stability

(i)  $\beta > 1$ : Here, in equation (2.16),  $\rho_{1,2}$  are both real and positive and  $\rho_1 > 1 > \rho_2 > 0$ . Consequently  $\mu_1$  in equation (2.18) is real and positive, while  $\mu_2 (= -\mu_1)$  is real negative. From the general theory of section 3.2 (for instance equation (A.18)) it can be deduced that the general solution of equation (2.10) is

$$\Phi = c_1 e^{\mu_1 x} p_1(x) + c_2 e^{-\mu_1 x} p_2(x) \quad (2.19)$$

where for all  $x$ ,

$$p_{1,2}(x + L) = p_{1,2}(x)$$

There are no periodic solutions and, in general,  $|\Phi| \rightarrow \infty$  as  $t \rightarrow \infty$ . Thus equation (2.10) describes unstable behavior.

(ii)  $\beta < -1$ : Here, equation (2.16),  $\rho_{1,2}$  are both real and negative and  $\rho_2 < -1 < \rho_1 < 0$ . Substituting

$$\mu_1 = \frac{i\pi}{L} - \gamma, \quad \cosh \mu_1 \gamma L = -\beta \quad (2.20)$$

The general solution of equation (2.10) is now

$$\Phi = c_1 e^{-\gamma_1 x} q_1(x) + c_2 e^{\gamma_1 x} q_2(x) \quad (2.21)$$

where

$$q_{1,2}(x + 2L) = q_{1,2}(x)$$

In contrast to case(i), the underlying period is  $2L$ . Again there are no periodic solutions and in general  $|\Phi| \rightarrow \infty$  as  $t \rightarrow \infty$ , so that equation (2.10) describes unstable behavior.

Thus both cases (i) and (ii) have  $\rho$  real for which  $|\rho_{1,2}| \neq 1$ . This implies that the Bloch eigenfunctions  $\Phi$  are unstable to spatial translations along x axis.

(iii)  $-1 < \beta < 1$ : Here  $\rho_{1,2}$  are both complex-valued, with unit magnitude (i.e.  $|\rho_{1,2}| = 1$ ). Indeed,

$$\rho_{1,2} = \exp(\pm i\sigma L), \quad \mu_1 = i\sigma \tag{2.22}$$

where

$$\cos \sigma L = \beta, \quad (0 < \sigma L < \pi)$$

The general solution of equation (2.10) can be given as

$$\Phi = c_1 \text{Re}[e^{i\sigma x} p(x)] + c_2 \text{Im}[e^{i\sigma x} p(x)] \tag{2.23}$$

where for all x

$$p(x + L) = p(x)$$

Here, of course,  $p(x)$  is a complex-valued periodic function. The solutions are bounded and oscillatory, and thus equation (2.10) describes stable behavior. There are no periodic solutions of period  $L$ , or  $2L$ , there are exceptionally, periodic solutions of period  $mL$  whenever  $\sigma L = 2\pi/m$  for  $m = 3, 4, \dots$

Case (iii) implies that the Bloch eigenfunctions are stable under spatial translation. In this context, the entire real  $\lambda$  axis is a band of stability. All other stable bands in the complex  $\lambda$  plane are called *spines*.

(iv)  $\beta = 1$ : This case is boundary between cases (i) and (iii). There is just a single characteristic multiplier,  $\rho_1 = 1$ , and a single characteristic exponent  $\mu_1 = 0$ . It can be regarded as the limit  $\mu_1 \rightarrow 0$  in case (i), or  $\sigma \rightarrow 0$  in case (iii). The general solution is

$$\Phi = c_1 p_1(x) + c_2 [kx p_1(x) + p_2(x)] \quad (2.24)$$

where for all  $x$

$$p_{1,2}(x + L) = p_{1,2}(x)$$

Here  $k$  is a constant, which may equal zero. This case is one of marginal stability and significantly, choosing  $c_2 = 0$ , there exists a solution of period  $L$ .

(v)  $\beta = -1$ : This case is boundary between cases (ii) and (iii). There is again just a single characteristic multiplier,  $\rho_1 = -1$ , and a single characteristic exponent  $\mu_1 = i\pi/L$ . It can be regarded as the limit  $\gamma \rightarrow 0$  in case (ii), or  $\sigma \rightarrow \pi/L$  in case (iii). The general solution is

$$\Phi = c_1 q_1(x) + c_2 [kx q_1(x) + q_2(x)] \quad (2.25)$$

where for all  $x$

$$q_{1,2}(x + 2L) = q_{1,2}(x)$$

Again  $k$  is a constant, which may equal zero. This case is also one of marginal stability and significantly, choosing  $c_2 = 0$ , there exists a solution of period  $2L$ .

Case (iv) and (v) correspond to discrete points in the  $\lambda$  plane where  $\beta = \pm 1$  and the Bloch eigenfunctions are either periodic or antiperiodic (see section 2.2). This set of eigenvalues in the  $\lambda$  domain is called the *main spectrum of the periodic NLS*

equation which can be obtained by

$$\frac{1}{2}\text{Tr}M = \pm 1 \quad (2.26)$$

### 2.2.3.3 Numerical Algorithm

It has been shown that the monodromy matrix for the NSE floquet problem

$$\Phi_x = Q(\lambda)\Phi, \quad Q = \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix}$$

is given by  $M(\lambda, u)$ , where

$$\Phi(x + L) = M(\lambda, u)\Phi(x) \quad (2.27)$$

Thus,  $M$  is effectively a state transition matrix operator which translate  $x$  by  $L$ .

Let us consider a narrow-banded, periodic nonlinear wave train  $\eta(x, 0)$  with complex envelope function  $u(x, 0) = A(x, 0)\exp[i\phi(x, 0)]$  (assumed to be a solution of the NSE). The wave train is assumed to have periodic boundary conditions, so that  $\eta(x, 0) = \eta(x + L, 0)$ , and  $u(x, 0) = u(x + L, 0)$  on the interval  $(0 \leq x \leq L)$

A piecewise-constant discretization is then assumed for the wave envelope function  $u(x, 0)$ , divided into  $N$  constant values  $u_n = u_n(x_n, 0)$  ( $1 \leq n \leq N$ ) inside spatial intervals  $\Delta x = L/N$ ,  $x_n = x_0, x_1, x_2, \dots, x_{N-1}, x_N$  as described in reference [38]. Here  $x_N = x(x_0 + L)$ .

Thus, one can arrive at

$$\begin{aligned}
\Phi(x_0 + \Delta x) &= \Phi(x_1) = e^{\Delta x Q(\lambda, u_1)} \Phi(x_0) = U(u_1) \Phi(x_0) \\
\Phi(x_1 + \Delta x) &= \Phi(x_2) = e^{\Delta x Q(\lambda, u_2)} \Phi(x_1) = U(u_2) \Phi(x_1) \\
&\vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \vdots \\
\Phi(x_{N-1} + \Delta x) &= \Phi(x_0 + L) = e^{\Delta x Q(\lambda, u_N)} \Phi(x_{N-1}) = U(u_N) \Phi(x_{N-1})
\end{aligned}$$

Combining the above, one arrives at

$$\begin{aligned}
\Phi(x_0 + L) &= U(u_1) U(u_2) \dots U(u_N) \Phi(x_0) \\
&= \prod_{j=1}^N U(u_j, \lambda) \Phi(x_0)
\end{aligned} \tag{2.28}$$

Thus, the monodromy matrix is given by

$$M(x_0, \lambda) = \prod_{j=1}^N U(u_j, \lambda) \tag{2.29}$$

where  $U(u_n, \Delta x)$  is the exponential of the trace vanishing matrix  $Q(\lambda)$ :

$$\begin{aligned}
U(u) &= e^{\Delta x Q(\lambda)} = \exp \left[ \Delta x \begin{pmatrix} -i\lambda & u \\ \sigma u^* & i\lambda \end{pmatrix} \right] \\
&= \begin{pmatrix} \cosh(k\Delta x) - \frac{i\lambda}{k} \sinh(k\Delta x) & \frac{u}{k} \sinh(k\Delta x) \\ \frac{\sigma u^*}{k} \sinh(k\Delta x) & \cosh(k\Delta x) + \frac{i\lambda}{k} \sinh(k\Delta x) \end{pmatrix}
\end{aligned} \tag{2.30}$$

Here  $k^2 = \sigma|u|^2 - \lambda^2$  is constant inside an interval  $\Delta x$ .

The trace of the monodromy matrix is used to determine the main spectrum eigenvalues,  $\lambda_k$  for  $k = 1, 2, \dots, 2N$ , by equation (2.26) as

$$\frac{1}{2} Tr M = \frac{1}{2} (M_{11} + M_{22}) = a_R(x_0, \lambda) = \pm 1 \tag{2.31}$$

The main spectrum eigenvalues can be used to reconstruct the potential  $u$  or the solution to the NSE using Riemann theta functions. However, this is beyond the scope of the current work. The main spectrum eigenvalues having non-zero imaginary parts correspond to the 'unstable modes' that give rise to rogue-wave solutions.

### 2.2.3.4 Results

The numerical algorithm described above has been implemented using a MATLAB code. The results shown in this section are for the focusing NSE i.e.  $\sigma = -1$ . The initial condition chosen is a cosine perturbed wave given by

$$u(x, 0) = a \left[ 1 + \sum_{n=1}^N \epsilon_n \cos(nKx - \theta_n) \right] \quad (2.32)$$

The complex wave train is obtained by taking the Hilbert transform. The surface elevation is given by

$$\eta(x, 0) = a \left[ 1 + \sum_{n=1}^N \epsilon_n \cos(nKx - \theta_n) \right] \cos(k_0x) \quad (2.33)$$

where  $k_0$  is the carrier wave number corresponding to a wavelength of  $L/10$  where  $L$  is the wavelength (one period) of the complex envelope. Without loss of generality, the unmodulated carrier wave amplitude  $a$  has been taken to be 1. Note that the carrier wavelength is chosen in an adhoc manner since it doesn't affect the solution. Periodic inverse scattering theory tells us that unstable modes occur only when  $aL \geq \sqrt{2}\pi$  [52]. Hence,  $L$  has been taken to be greater or equal to 4.44.

The spectral eigenvalue problem has been solved for various initial conditions (2.32) by varying parameter values for  $\epsilon_n$ ,  $\theta_n$ , and  $L$ .

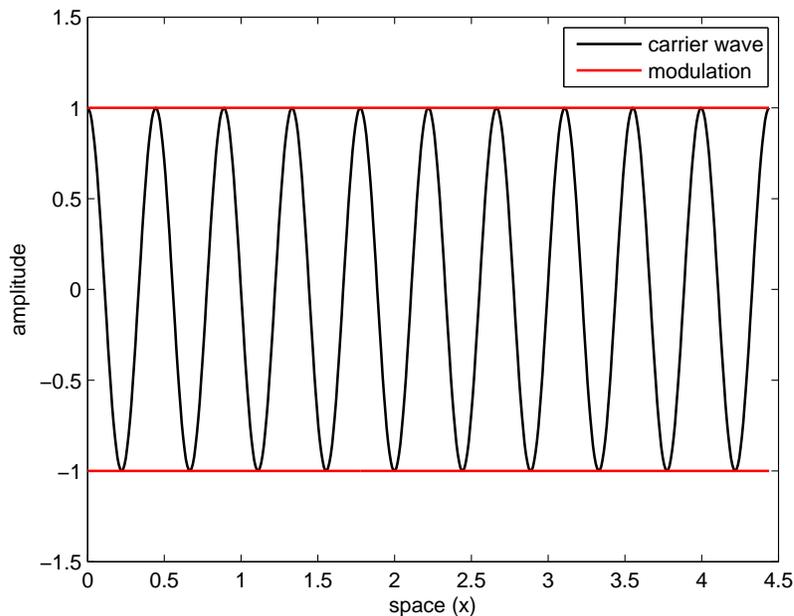


Figure 2.2: Initial modulated wave train with  $\epsilon = 10^{-5}$  and wavelength  $L = 4.44$  .

To begin with, a small-amplitude modulation has been considered with  $\epsilon = 10^{-5}$ ,  $L = 4.44$  and  $\theta = 0$ . Thus, the initial envelope has the form  $1 + \epsilon \cos Kx$ . Shown in Figure 2.2 is the initial modulated wave train for this case.

The  $\lambda$  plane for this wave train has been shown in Figure 2.3. The eigenvalues on the real axis correspond to *stable Stokes waves*. The unstable mode below the carrier wave is at  $\lambda = ia/\sqrt{2}$ . In reality there are two  $\times$ 's spaced slightly apart ( $\epsilon = 10^{-5}$ ), but the distance is so small that we cannot see it.

Now, shown in Figure 2.4 with  $\epsilon = 0.05$ ,  $a = 1$  and  $L = 2$  is considered. Since in this case  $aL < 4.44$ , shown in Figure 2.5, there are no unstable modes on the imaginary axis. Only two stable Stokes modes that appear on the real axis.

However, when the wavelength  $L$  is increased to 4.44 for the same  $\epsilon$  (i.e  $\epsilon = 0.05$ ) and  $a$  , the unstable modes appear as can be seen in Figure 2.6

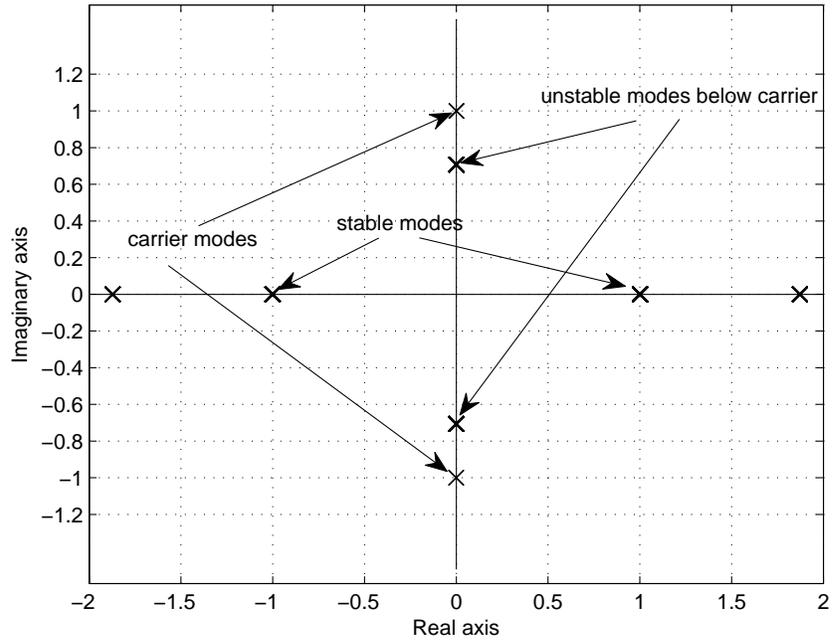


Figure 2.3:  $\lambda$  plane spectrum of a plane carrier wave which is modulated by unstable, small-amplitude ( $\epsilon = 10^{-5}$ ) sine wave and has one (homoclinic) unstable mode.

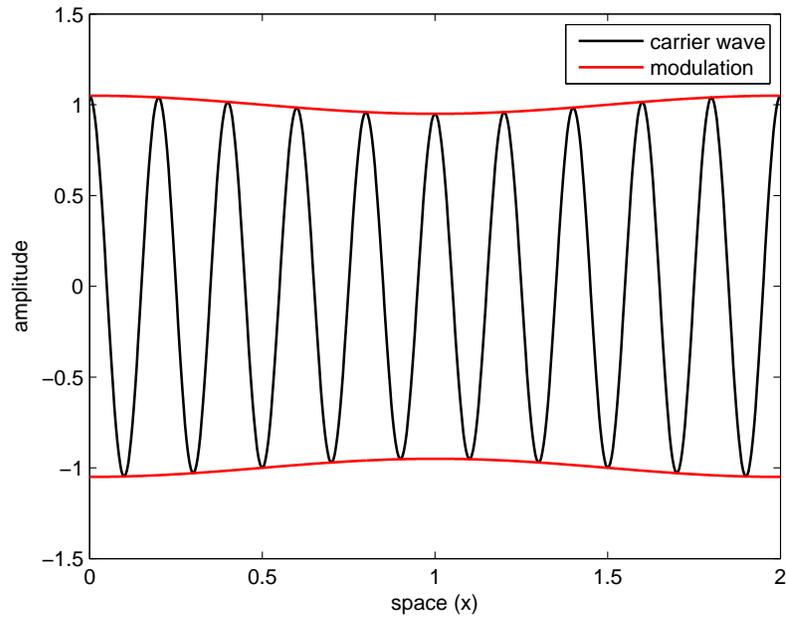


Figure 2.4: Initial modulated wave train with  $\epsilon = 0.05$  and wavelength  $L = 2$ .

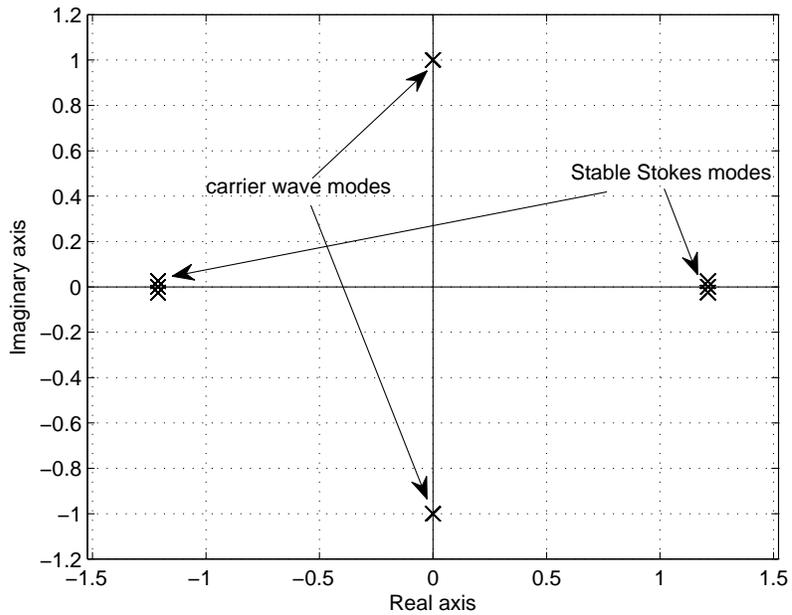


Figure 2.5:  $\lambda$  plane spectrum of a plane carrier wave which is modulated by stable, small-amplitude ( $\epsilon = 0.05$ ) sine wave and has two stable modes.

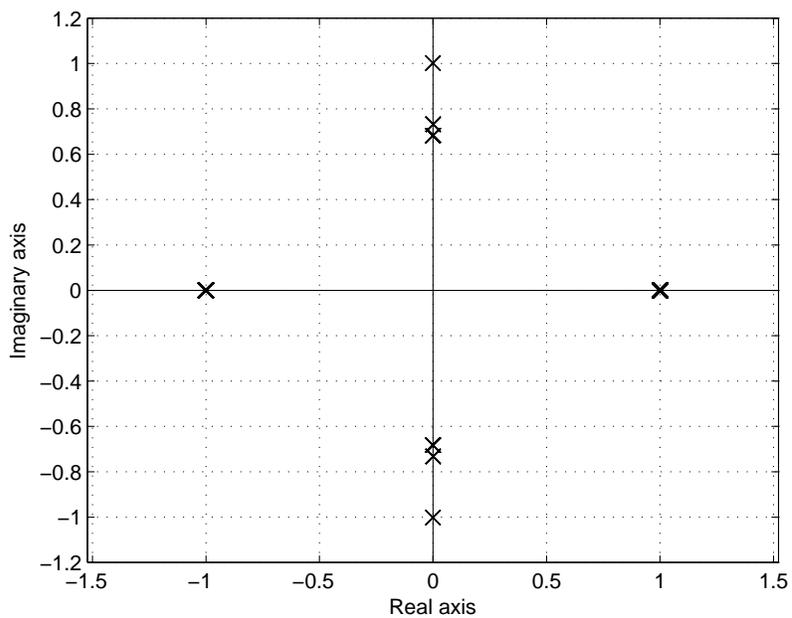


Figure 2.6:  $\lambda$  plane spectrum of a plane carrier wave which is modulated by small-amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength  $L = 4.44$ .

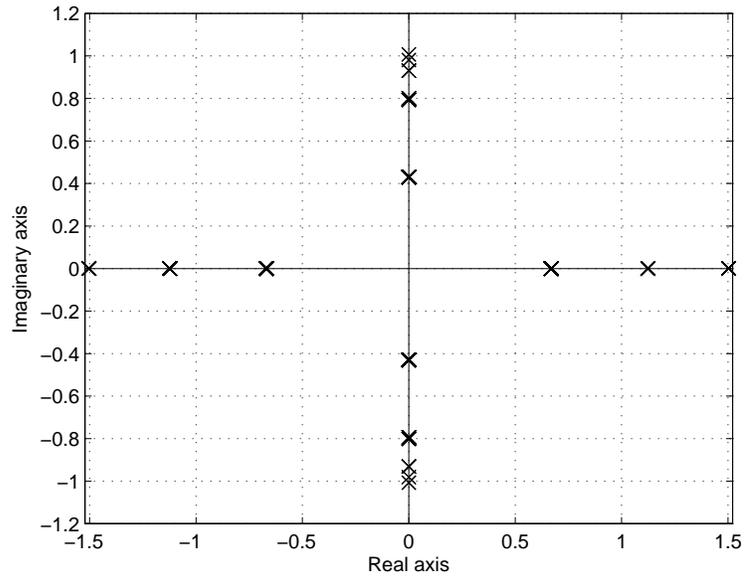


Figure 2.7:  $\lambda$  plane spectrum of a plane carrier wave which is modulated by small-amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength  $L = 10$ .

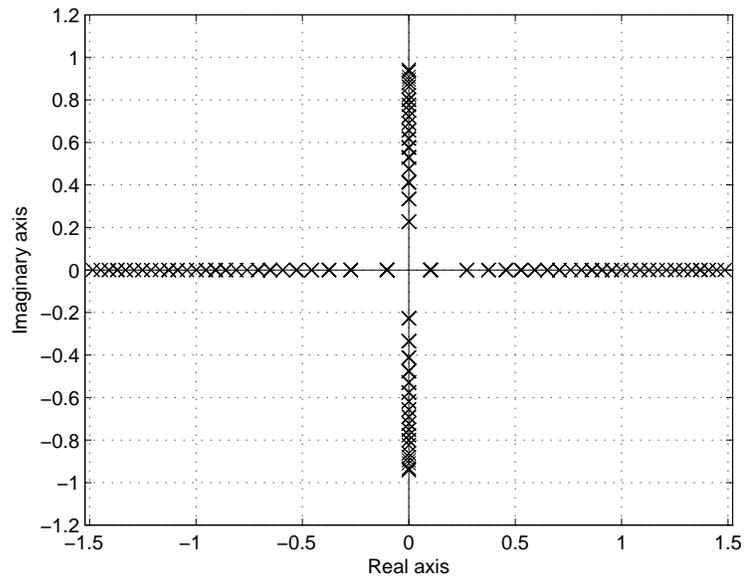


Figure 2.8:  $\lambda$  plane spectrum of a plane carrier wave which is modulated by small-amplitude ( $\epsilon = 0.05$ ) sine wave having a wavelength  $L = 100$ .

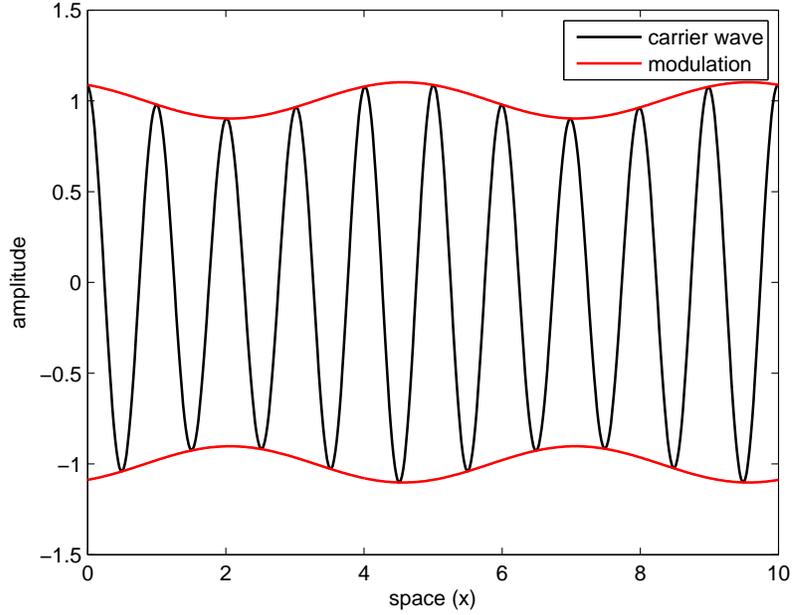


Figure 2.9: Initial modulated wave train with  $\epsilon_1 = 0.1i$ ,  $\epsilon_2 = 0.1$ ,  $\theta_1 = \pi/3$ ,  $\theta_2 = \pi/6$ , and wavelength  $L = 10$ .

As shown in Figure 2.7 and Figure 2.8, the number of main spectrum eigenvalues on the  $\lambda$  plane increases with the increase in wavelength  $L$  (from  $L = 10$  to  $L = 100$ ).

Next, a case with non-zero phases and complex  $\epsilon$  is considered. Let

$$\epsilon_1 = 0.1i, \quad \epsilon_2 = 0.1 \quad \theta_1 = \pi/3, \quad \theta_2 = \pi/6$$

Thus, as shown in Figure 2.9

$$u(x, 0) = 1 + 0.1i \cos(Kx - \pi/3) + 0.1 \cos(2Kx - \pi/6)$$

where  $L = 10$  and  $K = 2\pi/L = \pi/5$ .

In Figure 2.10, the main eigenvalue spectrum plot for this wave train is shown.

Finally, another similar case with following parameter values is considered:

$$\epsilon_1 = 0.05, \quad \epsilon_2 = 0.1i, \quad \epsilon_3 = 0.05i \quad (2.34)$$

$$\theta_1 = \pi/3, \quad \theta_2 = \pi/3, \quad \theta_3 = \pi/6 \quad (2.35)$$

Shown in Figure 2.11 is the initial modulated wavetrain. The actual envelope profile is shown in Figure 2.12. The eigenvalue plot has been shown in Figure 2.13.

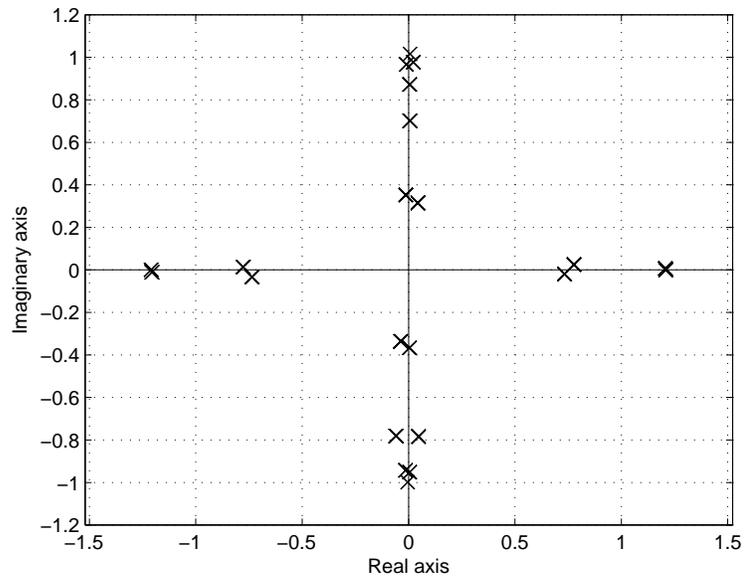


Figure 2.10:  $\lambda$  plane spectrum of the initial wavetrain shown in Figure 2.9.

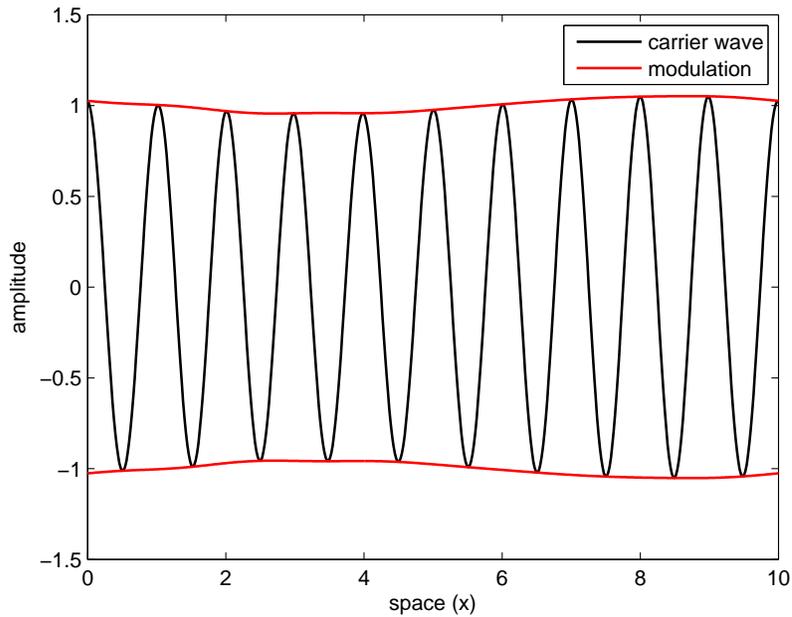


Figure 2.11: Initial modulated wavetrain with parameters shown in equations (2.34) and (2.35) and wavelength  $L = 10$ .

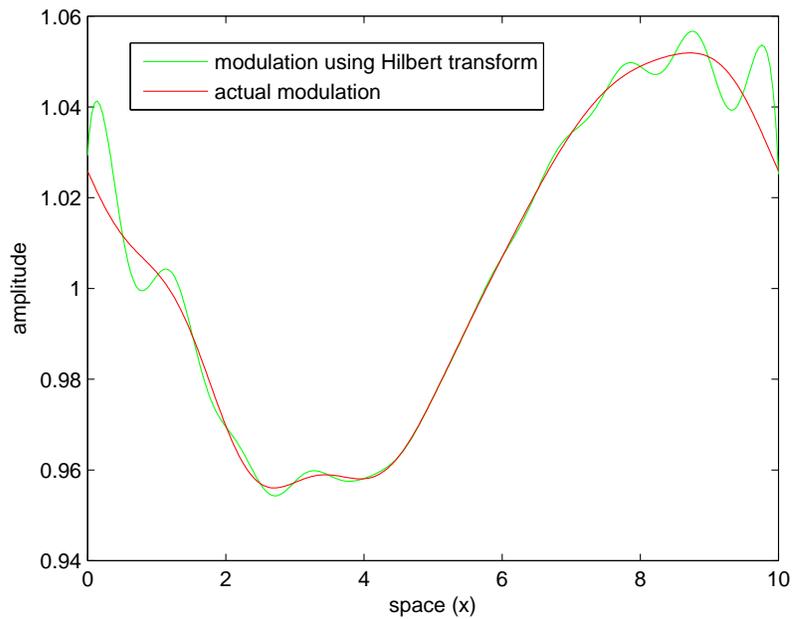


Figure 2.12: Actual envelope profile (magnified) of the initial wavetrain shown in Figure 2.11.

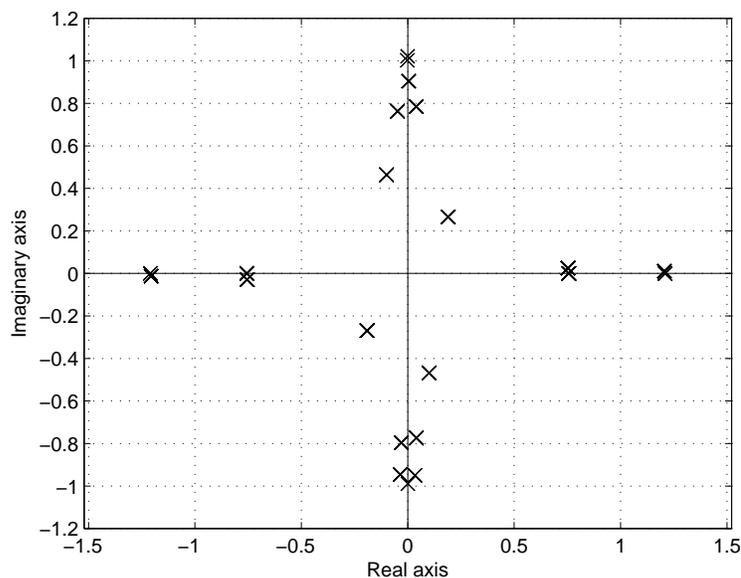


Figure 2.13:  $\lambda$  plane spectrum of the initial wavetrain shown in Figure 2.11.

## 2.2.4 Reconstruction of Potential

The eigenvalue solutions to equation (2.9) constitute the *main spectrum*. They are complex parameters which determine the global nature of the wave potential  $u$  such as the periods in space and time and their amplitudes. Algorithm for solving for these main spectrum eigenvalues algebraically is discussed in references [20] and [51].

Another set of spectral data called the ‘auxiliary spectra’ and their ‘sheet indices’, denoted by  $(\mu_j; \sigma_j)$   $j = 1, 2, \dots, N - 1$  are required to completely reconstruct the potential. The complex variables  $\mu_j(x, t)$  obey a set of ordinary differential equations in space and time. These variables generate the wave’s dynamic behavior in space-time. The sheet indices  $\sigma_j$  are required because the auxiliary variables reside on two-sheeted Riemann surface. They take values  $\pm 1$ .

### 2.2.4.1 Solutions using Hyperelliptic Functions

The auxiliary variables  $\mu_j$  are *hyperelliptic functions*, the equations of motion of which are given by

$$\begin{aligned}\mu_{jx} &= \frac{-2i\sigma_j\sqrt{\prod_{k=1}^{2N}(\mu_j - \lambda_k)}}{\prod_{m \neq j}(\mu_j - \mu_m)} \\ \mu_{jt} &= -2\left(\sum_{m \neq j} \mu_m - \frac{1}{2}\sum_{k=1}^{2N} \lambda_k\right)\mu_{jx}\end{aligned}\tag{2.36}$$

where  $\lambda_k$  are the main spectrum eigenvalues. It can be shown by calculating the crossed derivatives in  $x$  and  $t$  that these equations are self consistent. That is to say that a solution to these equations always exists. These equations can be solved analytically by using Abel transform [51].

From these variables we can construct the potential  $u$  by solving the following set of differential equations:

$$\begin{aligned}\partial_t \ln u(x, t) &= 2i\left(\sum_{j>k} \lambda_j \lambda_k - \frac{3}{4}\left(\sum_{k=1}^{2N} \lambda_k\right)^2\right) \\ &\quad - 4i\left(-\frac{1}{2}\sum_{k=1}^{2N} \lambda_k\right)\left(\sum_{j=1}^{N-1} \mu_j\right) + \sum_{j>k} \mu_j \mu_k\end{aligned}\tag{2.37}$$

$$\partial_x \ln u(x, t) = 2i\left(\sum_{m \neq j} \mu_j(x, t) - \frac{1}{2}\sum_{k=1}^{2N} \lambda_k\right)\tag{2.38}$$

An analytic form of the solution can be constructed using equations (2.36), (2.37), and (2.38). For example, the analytic expression for a single unstable ‘rogue’ mode corresponding to main spectrum eigenvalue  $\lambda = ia/\sqrt{2}$  [39] is given by

$$u(x, t) = a \left[ \frac{\cos(\sqrt{2}ax) \operatorname{sech}(2a^2t) + i\sqrt{2} \tanh(2a^2t)}{\sqrt{2} - \cos(\sqrt{2}ax) \operatorname{sech}(2a^2t)} \right] e^{2ia^2t}\tag{2.39}$$

The space time dynamics of this solution for  $a = 1$  is shown in Figure 2.14.

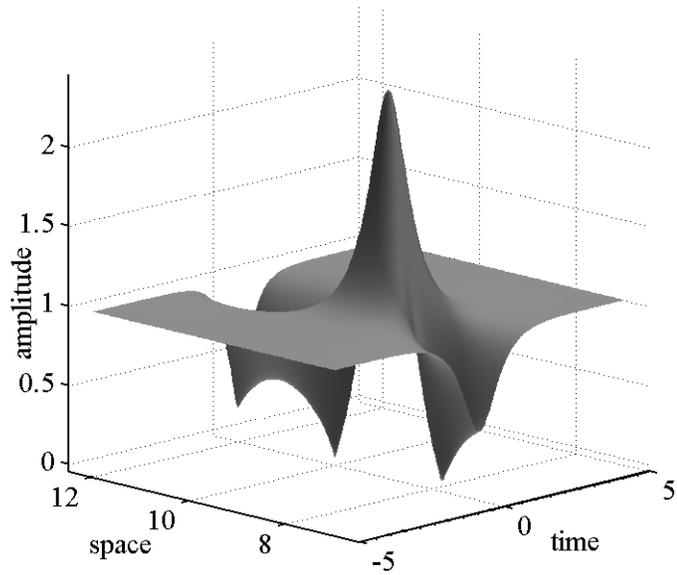


Figure 2.14: Modulus of the space time evolution of the rogue-wave solution for  $\lambda = ia/\sqrt{2}$ .

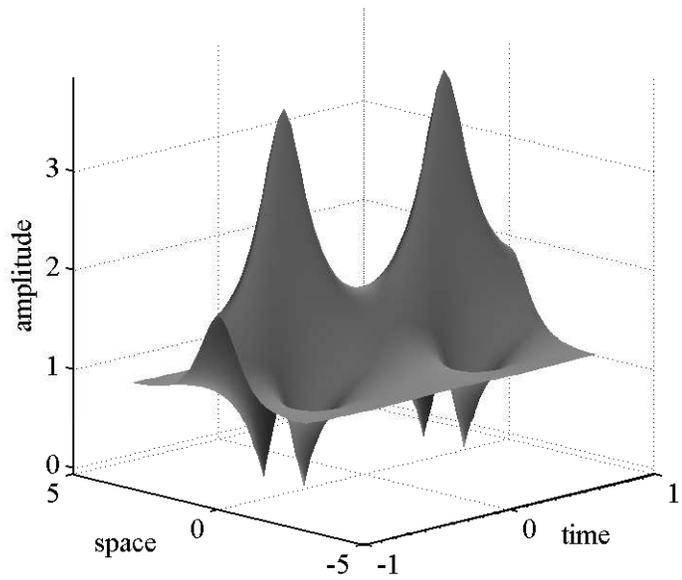


Figure 2.15: Modulus of the space time evolution of the rogue-wave solution for  $\lambda = ia\sqrt{2}$ .

Another analytic expression for a single unstable ‘rogue’ mode corresponding to main spectrum eigenvalue  $\lambda = ia\sqrt{2}$  [39] is given by

$$u(x, t) = a \left[ 1 + \frac{2(\cos(4\sqrt{2}a^2t) + i\sqrt{2}\sin(4\sqrt{2}a^2t))}{\cos(4\sqrt{2}a^2t) + \sqrt{2}\cosh(2ax)} \right] e^{2ia^2t} \quad (2.40)$$

The space time dynamics of this solution for  $a = 1$  is shown in Figure 2.40

#### 2.2.4.2 Solutions using Riemann Theta Functions

Space periodic spectral solutions to the NSE can be described by

$$u(x, t) = a \frac{\Theta(x, t|\tau, \delta^-)}{\Theta(x, t|\tau, \delta^+)} e^{2ia^2T}. \quad (2.41)$$

where  $\Theta(x, t|\tau, \delta^\pm)$  is a Riemann theta function [52], [40]. The  $\Theta(x, t|\tau, \delta^\pm)$  are generalized Fourier series known as N-dimensional Riemann theta functions:

$$\Theta(x, t|\tau, \delta^\pm) = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} \exp i \left[ \sum_{n=1}^N m_n K_n x + \sum_{n=1}^N m_n \Omega_n T + \sum_{n=1}^N m_n \delta_n^\pm + \sum_{j=1}^N \sum_{k=1}^N m_j m_k \tau_{jk} \right] \quad (2.42)$$

A single unstable mode can be considered by taking  $\Theta(x, t|\tau, \delta^\pm)$  as a two-dimensional theta function defined as

$$\Theta(x, t|\tau, \delta^\pm) = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} \exp i \left[ \sum_{n=1}^2 m_n K_n x + \sum_{n=1}^2 m_n \Omega_n T + \sum_{n=1}^2 m_n \delta_n^\pm + \sum_{j=1}^2 \sum_{k=1}^2 m_j m_k \tau_{jk} \right] \quad (2.43)$$

The parameters governing the theta function ( $K_n$ ,  $\Omega_n$ , and  $\delta^\pm$ ) are defined in terms of five spectral parameters  $a$ ,  $\lambda_R$ ,  $\lambda_I$ ,  $\epsilon_0$ , and  $\theta$  where  $a$  is the carrier wave amplitude,  $\lambda$  is the main spectrum eigenvalue and  $\epsilon_0$ , and  $\theta$  are as shown in Figure 2.18. Following the notation used in earlier work [40], the spectral parameters are defined as

$$\epsilon_1 = \epsilon_0 e^{i\theta}, \quad \epsilon_2 = \epsilon_1^*, \quad \sigma_1 = 1, \quad \sigma_2 = -1 \quad (2.44)$$

$$\lambda_1 = \lambda_R + i\lambda_I, \quad \lambda_2 = \lambda_1^* \quad (2.45)$$

$$K_n = -2\sqrt{a^2 + \lambda_n^2}, \quad \Omega_n = 2\lambda_n K_n \quad (2.46)$$

$$\delta_n^\pm = \pi + i \ln(\lambda_n \mp \frac{1}{2}K_n) + i \ln(\sigma_n \lambda_n - (-1)^n \frac{1}{2}K_n) \quad (2.47)$$

$$\begin{aligned} \tau_{11} &= \frac{1}{2} + \frac{i}{\pi} \ln \left( \frac{K_1^2}{\epsilon_1} \right), & \tau_{22} &= \frac{1}{2} + \frac{i}{\pi} \ln \left( \frac{K_2^2}{\epsilon_2} \right) \\ \tau_{12} = \tau_{21} &= \frac{i}{2\pi} \ln \left( \frac{1 + \lambda_1 \lambda_2 + \frac{1}{4}K_1 K_2}{1 + \lambda_1 \lambda_2 - \frac{1}{4}K_1 K_2} \right) \end{aligned} \quad (2.48)$$

### 2.2.5 Solution Procedure to Explore $\lambda$ Plane

A procedure which allows the discovery of a certain form of rogue-wave solution to the NSE is presented. It is based on a predictor-corrector style framework. The steps involved in are summarized in the flowchart given in Figure 2.16 and explained with the aid of equations included in the previous sections. Through GPU

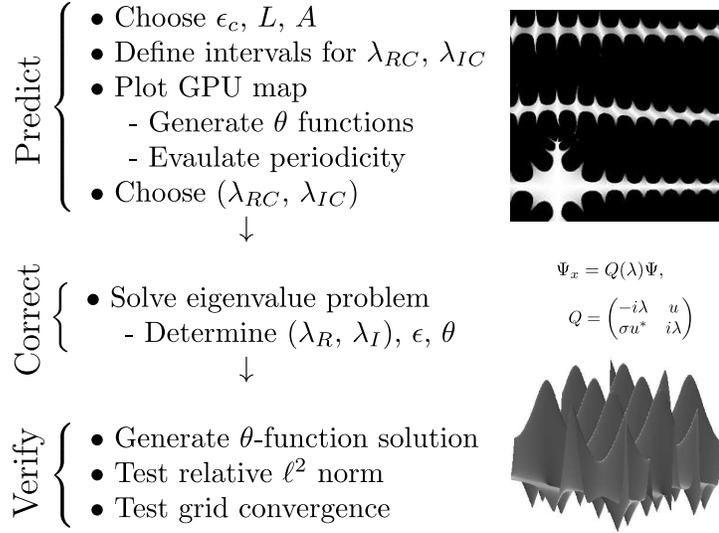


Figure 2.16: Flow chart illustrating the procedure to determine new Reimann theta function described rogue waves.

computing, this procedure allows an investigator to explore the parameter space which governs possible solutions of the form of equation (2.41). In the predictor step, a map of parameters which result in periodic functions,  $u(x, 0)$ , is generated. A particular combination of parameters that has a high likelihood of resulting in a rogue-wave can then be determined. In the corrector step, the parameters are conclusively refined by solving the spectral eigenvalues problem, shown in equation (2.9), based on the initial guess. In the verification step, a candidate solution is formed by substituting the corrected parameters into equation (2.41). The candidate solution is verified by numerical evaluating the NSE to see if a zero residual is obtained. Upon passing two numerical tests, the candidate solution is accepted as a solution. These steps are described further below.

### 2.2.5.1 Prediction

In this step, the solution space has been reduced by choosing  $a = 1$ ,  $\theta = 0$ , and various reasonable values for  $\epsilon_0 < 0.05$ . The choice of  $\theta$  and  $\epsilon_0$  are refined in the corrector step. These choices leave  $\lambda_R$  and  $\lambda_I$  as the only free parameters governing the initial function selection. While the mathematical relationships refer exclusively to  $(\lambda_R, \lambda_I)$ , these values are chosen in the predictor step, and later explicitly computed in the corrector step. Based on these two distinct classifications, in the text, these values which are chosen by an investigator are referred to as  $(\lambda_{RC}, \lambda_{IC})$  and those that are determined as solutions to the eigenvalue problem as  $(\lambda_R, \lambda_I)$ . The two-dimensional space can be evaluated for periodic functions by direct numerical evaluation of  $u(x, 0)$  using equation (2.41) over an interval  $nL$  (where  $n = 1, 2, 3$ ). Each function evaluation of  $u(x, 0)$  is computationally expensive; however, each evaluation is independent of another one. This allows a large domain to be evaluated rapidly through a GPGPU implementation. For this reason, equation (2.41) with  $(t = 0)$  is implemented in a CUDA kernel. A single thread is assigned to compute the function at a given coordinate  $(\lambda_{RC}, \lambda_{IC})$ . The periodicity of  $u(x, 0)$  is also determined in the GPGPU kernel, according to the metric presented in equation (2.51).

$$C_0 = \left| \frac{U(L, 0) - U(0, 0)}{U(0, 0)} \right| \quad (2.49)$$

$$C_1 = U_x(x, 0)|_{x=0} - U_x(x, 0)|_{x=L} \quad (2.50)$$

$$C_f = \begin{cases} C_1 & C_0 \leq 0.01 \\ N/A & otherwise \end{cases} \quad (2.51)$$

The resulting map of  $\lambda$  pairs that form periodic  $u(x, 0)$  functions is shown in Figure 2.17. Parameter combinations resulting in periodic functions are displayed in white. Combinations, which do not meet the periodicity criteria, are displayed in progressively darker colors. The thresholds are admittedly ad hoc. The solutions are constructed and verified in the corrector procedure. In Figure 2.17, the particular value of  $(\lambda_{RC} = 1.2495, \lambda_{IC} = 1.6125)$  is identified as a solution of interest (shown as an example).

The accelerations experienced by the GPU implementation increase with increasing domain size. For typical  $(\lambda_{RC}, \lambda_{IC})$  domain sizes of  $256 \times 256$ , the basic Matlab implementation required 133.0 seconds (baseline), the codegen implementation required 25.8 seconds ( $5.15\times$ ), and the CUDA kernel implementation required 0.0573 seconds ( $2,321\times$ ).

### 2.2.5.2 Correction

The parameters governing the periodic  $u(x, 0)$  function chosen above need to be refined to yield a solution to the NSE. The spectral eigenvalues are determined by solving the spectral eigenvalue problem given by equation (2.9). The eigenvalue problem of equation (2.9) is recast as a Floquet problem by appropriately discretizing the complex wavetrain as described in section 2.2.3.1. The choice of  $\lambda_{RC}$  and  $\lambda_{IC}$  from the GPU map implies periodic boundary conditions. The solution of the

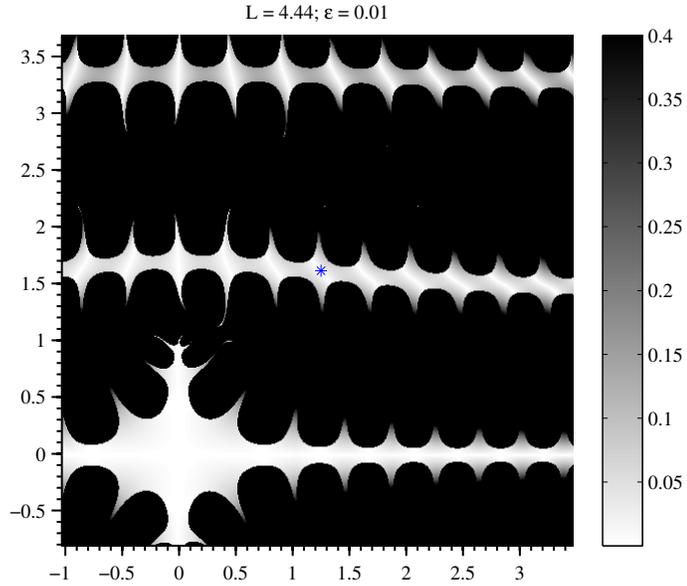


Figure 2.17: Map of periodic Reimann theta functions as defined in equation (2.41) for  $(A = 1, \theta = 0, \epsilon = 0.01, t = 0)$  generated by GPGPU computations. Light colored locations indicate periodic functions, while dark colored locations indicate aperiodic functions over the interval  $L$ . A point of interest is identified by an asterisk.

spectral eigenfunction  $\phi$  in each interval  $\Delta x$  is then obtained by integrating the eigenvalue problem for a constant potential as described in section 2.2.3.3. Finally, the main spectrum eigenvalues are obtained by solving for  $\lambda$  in equation (2.31).

Although many eigenvalues are determined, a pair of eigenvalues will be close to the single complex eigenvalue which is used to construct the original  $u(x, 0)$ , as shown in Figure 2.18. From this pair of complex eigenvalues, the parameter  $\epsilon$  is determined to be half the distance between the pair, and  $\theta$  is determined as the angle between the pair and the horizontal, beginning from the line adjoining the pair, as shown in the detailed portion of Figure 2.18. Based on experience,  $\epsilon$  is recognized to be of the same order as  $\epsilon_0$ .

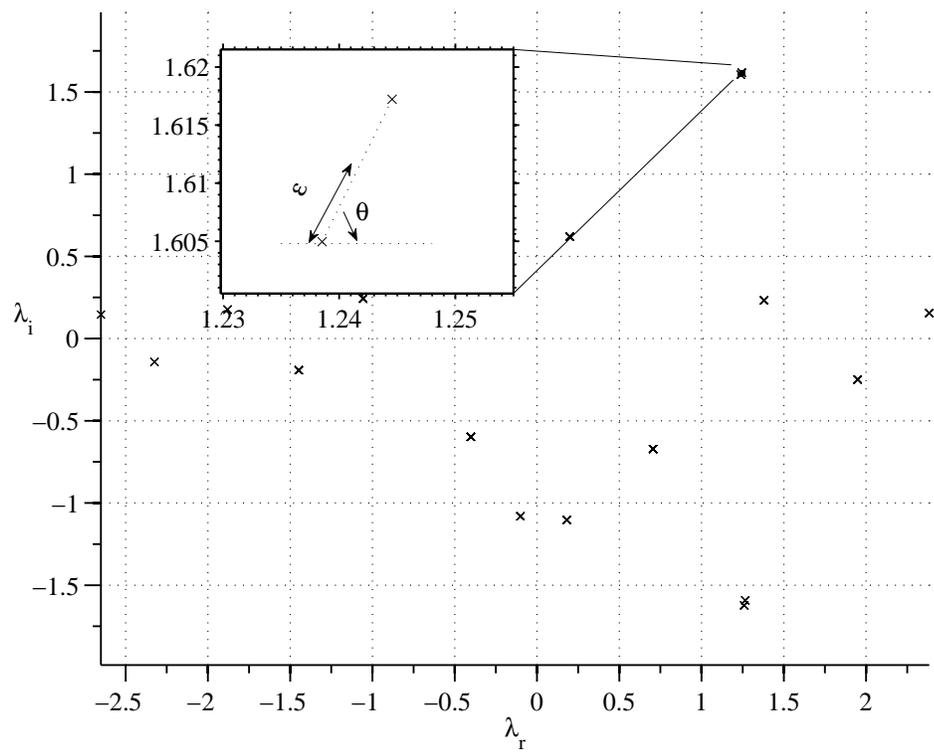


Figure 2.18: Main spectrum eigenvalues with a detailed view of  $\epsilon$  and  $\theta$  for a particular choice of  $(\lambda_R, \lambda_I)$ .

### 2.2.5.3 Verification

A candidate solution,  $\hat{u}(x, t)$ , is determined by evaluating equation (2.41) with the new set of parameters. The candidate solution is then verified by numerical integration into the NSE, equation (2.52) as

$$i\hat{u}_t - \hat{u}_{xx} + 2\sigma|\hat{u}|^2\hat{u} = \mathbf{R} \quad (2.52)$$

Eighth order central finite differences are used to evaluate both the spatial and temporal derivatives. Due to discretization and finite precision approximations, the numerical solution may not identically satisfy the equation, resulting in a non-zero residual. Two criteria were used to verify the solution. First, the  $\|\mathbf{R}(x, t)\|_2$  is compared to  $\|\hat{u}(x, t)\|_2$ , where  $\|\cdot\|_2$  denotes the  $\ell^2$  norm, as

$$\rho = \frac{\|\mathbf{R}(x, t)\|_2}{\|\hat{u}(x, t)\|_2} \quad (2.53)$$

All solutions presented in this work passed this test with  $\rho \ll 1.00\%$ . Second, a grid convergence test was performed. In this test, the quantity  $\|\mathbf{R}(x, t)\|_2 \Delta x \Delta t$  was verified to decrease as the grid size was refined in time and space for all solutions presented. Continuing with the procedure, after passing both tests, the candidate solution,  $\hat{u}(x, t)$ , is accepted as a solution  $u(x, t)$  to the nonlinear Schrödinger equation, as shown in Figure 2.19.

The solution can be analyzed in terms of its maximum amplitude defined in reference [40] as

$$\gamma = a + 2\lambda_I \tag{2.54}$$

where  $a = 1$  for all cases presented in this work. As of yet, unknown solutions with maximum amplitudes even slightly greater than the background can be of interest in systems where extreme waves are of interest. Solutions with large maximum amplitudes are of interest for other practical reasons. The GPU map allows identification of likely regions of solutions with selectable maximum amplitudes.

### 2.2.6 New Rogue-wave Solutions

The GPU map shown in Figure 2.17 contains several branches of candidate periodic solutions. Several horizontal bands can be discerned in this map, the major features of which are symmetric about the x and y axes. Although many solutions have been computed by using this procedure, only three solutions are featured here for brevity. First, focusing on the point  $(\lambda_{RC} = 1.2495, \lambda_{IC} = 1.6125)$  with  $L = 4.44$ , the corrected parameter set of  $(\lambda_R = 1.2415, \lambda_I = 1.61108, \epsilon = 0.006834, \theta = 1.11439)$  is obtained. Based on this parameter set, the solution, shown in Figure 2.19 has been generated. The maximum amplitude is approximately 4.2x the background modulation ( $a = 1$ ). The peaks are more compact in time than in space, dropping to the background level between adjacent crests as time increases.

The next solution examined is for  $L = 7$ . The GPU map for  $L = 7$  is shown in Figure 2.20. This solution exists on the second branch above the real axis in the corresponding GPU map. The starting point of  $(\lambda_{RC} = 0.6653, \lambda_{IC} = 1.2368)$  yields

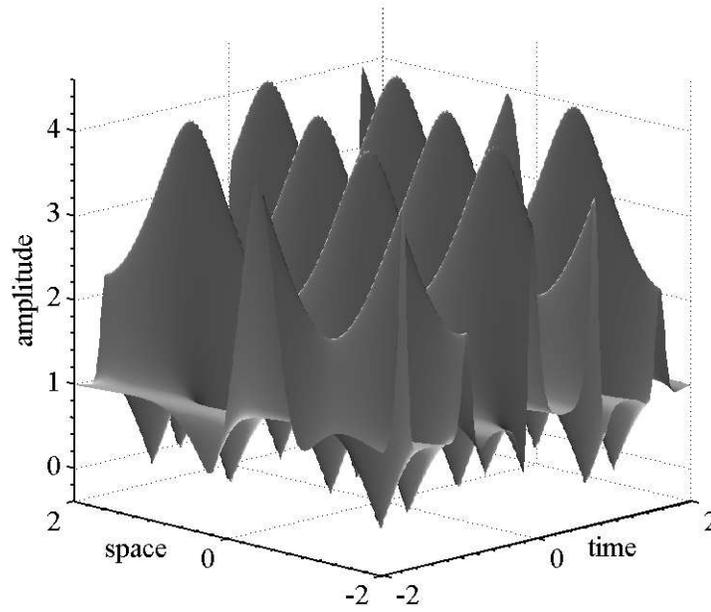


Figure 2.19: Solution to the NSE for  $(\lambda_R = 1.2415, \lambda_I = 1.61108, \epsilon = 0.006834, \theta = 1.11439)$  and  $L = 4.44$ . This solution envelope has periodic temporal peaks, which reach a maximum amplitude of  $\approx 4.2x$  the background.

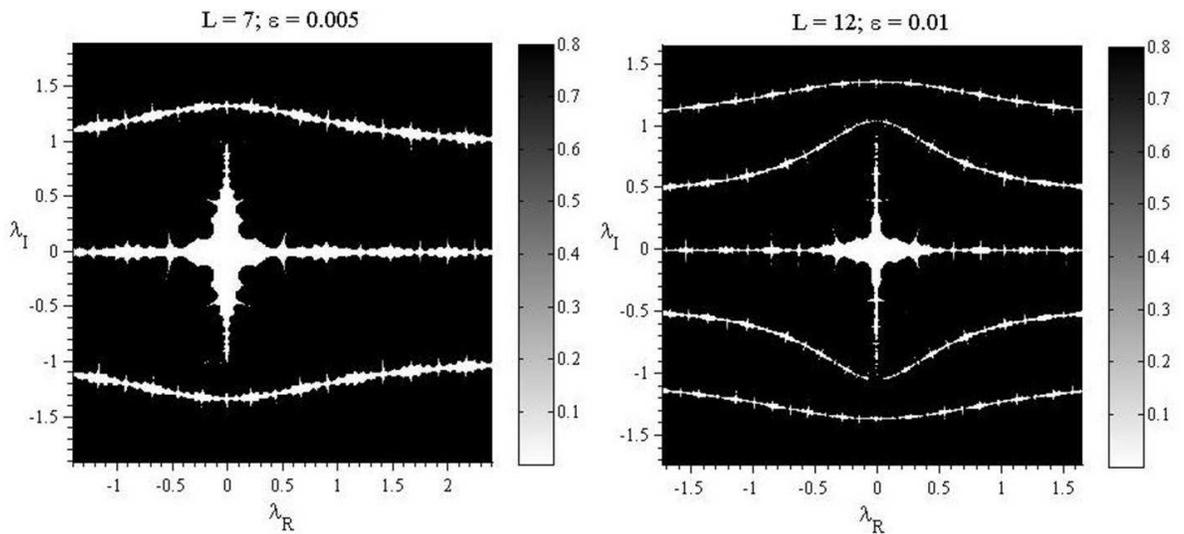


Figure 2.20: GPU maps for  $(L = 7, \epsilon = 0.005)$  and  $(L = 12, \epsilon = 0.01)$ .

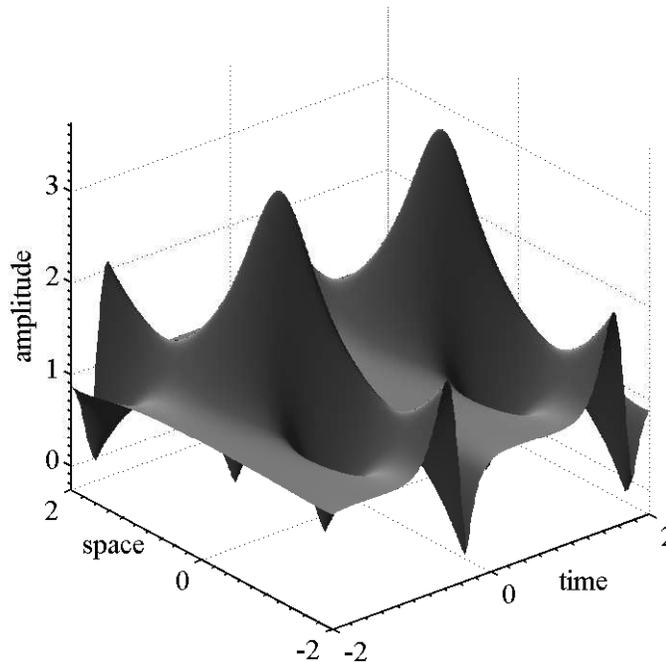


Figure 2.21: Solution to the NSE for  $(\lambda_R = 0.6616, \lambda_I = 1.23660, \epsilon = 0.00276, \theta = 0.87385)$  and  $L = 7$ .

the corrected parameters  $(\lambda_R = 0.6616, \lambda_I = 1.23660, \epsilon = 0.00276, \theta = 0.87385)$  after solving the spectral eigenvalue problem. The corresponding solution is shown in Figure 2.21. The maximum amplitude is 3.47x the background amplitude. This solution is characterized by localized peaks in time separated by flat regions of unit amplitude. The fluctuations again have compact support in the temporal domain.

The final example shown has been computed for  $L = 12$ . The parameter space map for  $L = 12$  is shown in Figure 2.20. The initial point was chosen near  $(\lambda_{RC} = 1.135, \lambda_{IC} = 1.206)$  for which the spectral eigenvalue problem yields the corrected parameters as  $(\lambda_R = 1.1367, \lambda_I = 1.2076, \epsilon = 0.004156, \theta = 0.372019)$ . This solution is illustrated in Figure 2.22. The peaks reach a maximum amplitude of 3.4x of the background and are less compact in time and space than the other

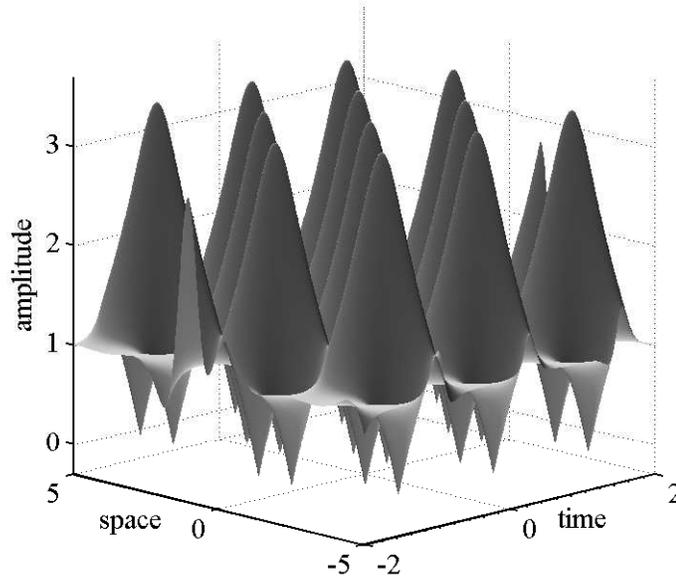


Figure 2.22: Solution to the NSE for  $(\lambda_R = 1.1367, \lambda_I = 1.2076, \epsilon = 0.004156, \theta = 0.372019)$  and  $L = 12$ .

solutions presented. Additional rogue-wave solutions to the NSE are presented in Appendix B.

### 2.2.6.1 Near Peregrine Solution

A singularity exists in the mapped space at  $(0, i)$ . This eigenvalue is associated with the Peregrine solution, in which the temporal and spatial periods  $\rightarrow \infty$ . Numerically, the Peregrine solution has only a single unique eigenvalue and does not conform to the eigenvalue solution procedure detailed above. Interestingly, solutions can exist near  $(0, i)$ . These solutions appear to be quite similar to the Peregrine solution, but demonstrate periodic fluctuations. The solutions also tend to decrease in residual error as  $L$  increases, which is consistent with the spatial support and temporal support of the Peregrine solution.

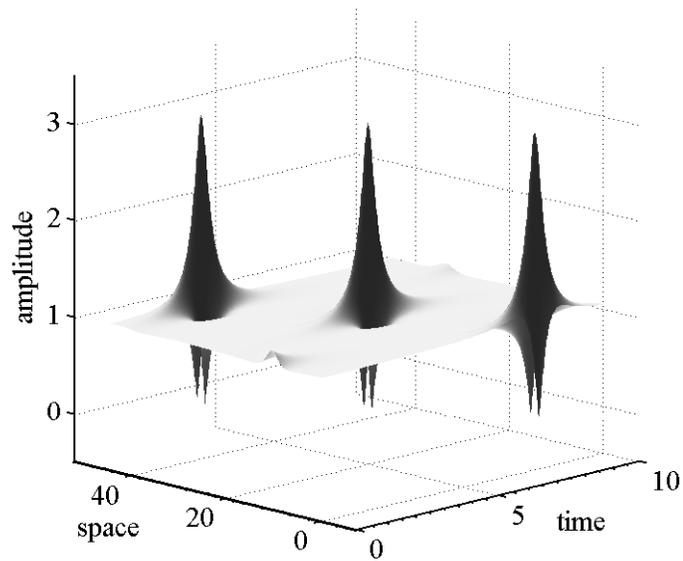


Figure 2.23: “Near Peregrine” solution with eigenvalues ( $\lambda_R = 0.0098, \lambda_I = 1.0068$ ) close to those associated with the Peregrine solution of the NSE.

One example of a “near Peregrine” solution defined by  $\lambda_R = 0.0098, \lambda_I = 1.0068, \epsilon = 0.0075, \theta = -0.3617, L = 15$  with error  $\rho = 0.04\%$  is examined more closely in Figure 2.23. The  $\lambda_i$  values are extremely close to  $(0, i)$ , but remain far enough to be resolved through the corrector procedure. The near Peregrine solution exhibits similar features to the peak of the Peregrine solution such as the peak amplitude and decay profile. This solution is compared to the exact Peregrine more closely in Section 2.2.7.

### 2.2.6.2 Isolated Solution

The bands that appear in the predictive map are sources of many solutions. However, for some combinations of spectral parameters, isolated points exist between the bands. The predictive map containing one such solution for  $L = 10$  is shown

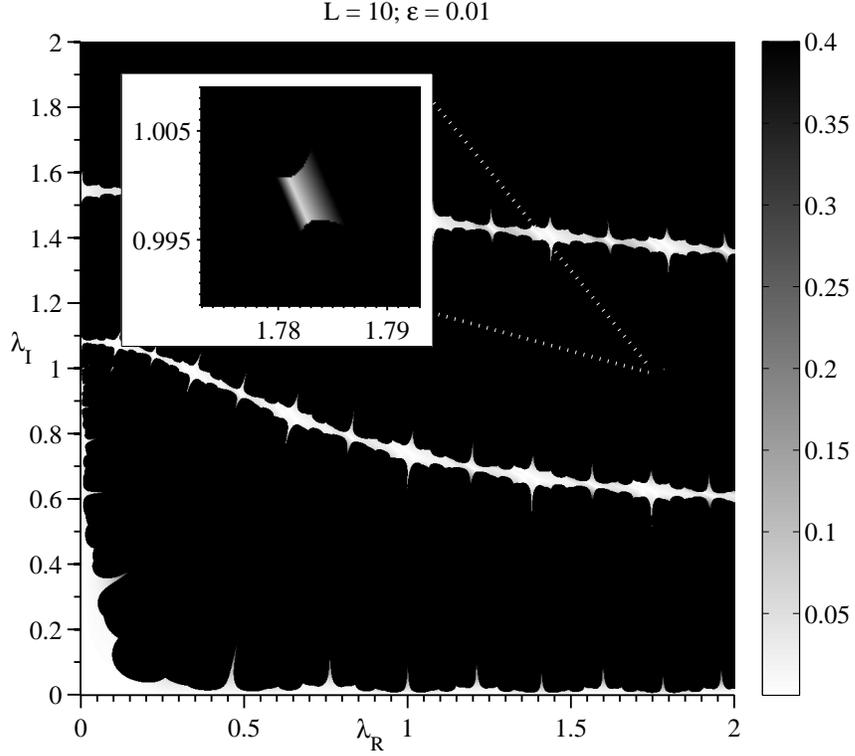


Figure 2.24: Predictive map for isolated solution ( $\lambda_R = 1.7806, \lambda_I = 0.99603$ ).

in Figure 2.24. A detailed view of the isolated point is provided in the inset. The spectral parameters of the corrected solution are ( $\lambda_R = 1.7806, \lambda_I = 0.99603$ ),  $\epsilon = 0.0027655086$ , and  $\theta = 0.007827789$ . This solution satisfied the original NSE with an extremely low error of  $\rho = 0.001\%$ . The solution, which is depicted in Figure 2.25, is found to exhibit large peaks and minor troughs.

### 2.2.6.3 Transition from Rogue Wave to Amplified Wave

As a final feature of the predictive map, the qualitative transition of a rogue-wave to a wave with less pronounced undulations is discussed. The direct relationship between the wave amplitude and the value of  $\lambda_I$  is provided by equation (2.54). The amplitude decreases along with the  $\lambda_I$  value. Solutions are readily determined

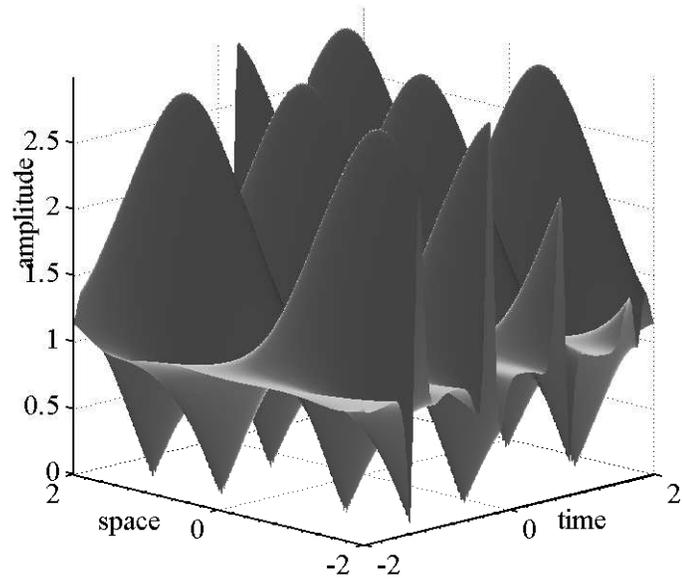


Figure 2.25: Solution to the NSE for  $(\lambda_R = 1.7806, \lambda_I = 0.99603)$ ,  $\epsilon = 0.0027655086$ ,  $\theta = 0.007827789$ , and  $L = 10$ . This solution envelope reaches a maximum amplitude of  $\approx 3x$  the background height.

on quantized bands; however, they do not exist for arbitrary variations in  $\lambda_I$  values. As the spectral parameter  $L$  increases, the solution bands tend to compress towards the real axis. Therefore, solutions with a desired  $\lambda_I$  can be obtained by choosing  $L$  appropriately so that a solution band exists at the desired value of  $\lambda_I$ . Such a variation in  $L$  is presented below. For  $L = [6, 8, \dots, 14]$ , solutions from the highest band of each solution space near the imaginary axis ( $\lambda_R \approx 0$ ) are compared. A composite map containing only the top branch of the predicted space for the given values of  $L$  is shown in Figure 2.26. The corrected  $(\lambda_R, \lambda_I)$  pairs of the progression are indicated with a white colored 'x' in the figure.

A measure of the “peakedness” of a solution,  $P_k$ , can be estimated by defining  $\xi$  as the height of the solution’s maximum and  $\zeta$  as the height of the solution’s maximum saddle point, as shown in Figure 2.27, and calculating

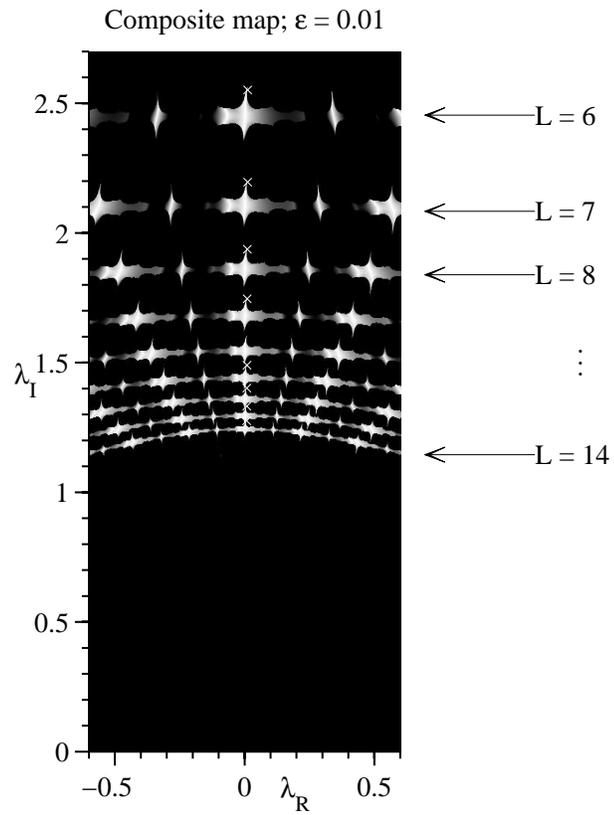


Figure 2.26: Composite predictive map containing the highest solution band for multiple values of  $L$ . Corrected pairs of  $(\lambda_R, \lambda_I)$  for generated solutions are marked with an 'x'.

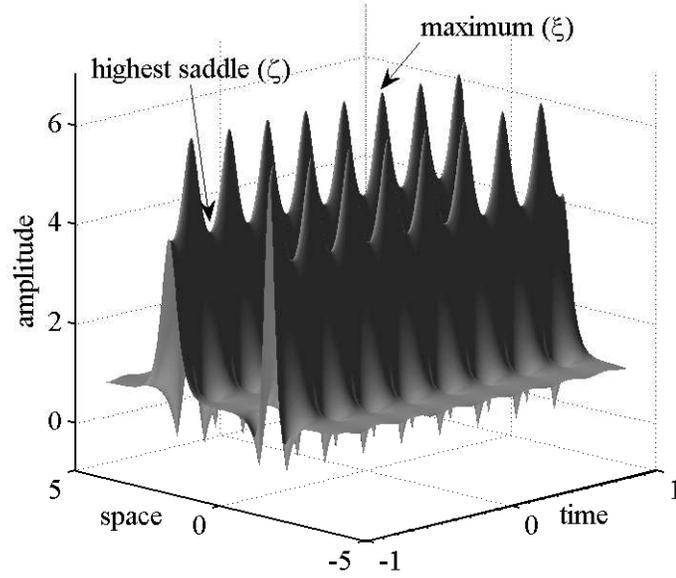


Figure 2.27: Peaked rogue-wave:  $L = 6$ ;  $\lambda = 0.01178 + 2.5512i$ ; and  $A_{max} = 6.1$

$$P_k = \frac{\xi - \zeta}{\xi} \quad (2.55)$$

An observation similar to equation (2.54) for the value of the wave's maximum saddle was determined to be  $A_{saddle} = 2\lambda_I - 1$ . Stated differently, and combined with equation (2.54), the wave's maximum is always two units higher than the height of the saddle point. This relationship was found to be consistent for solutions throughout the  $\lambda$  plane, including the Peregrine breather solution and solutions with  $\lambda_I < 1$ . The theoretical origin of this relationship and its required conditions will be considered in a future effort. Beginning with a solution on the highest branch of  $L = 6$ , the solution exhibits minimal peakedness, as depicted in Figure 2.27.

A collection of solutions from the top branch for several different values of  $L$  are shown in subsequent Figures 2.28 to 2.31. In the progression as  $L$  is increased, the values of  $\lambda_I$  is found to decrease. A solution chosen from the next higher value

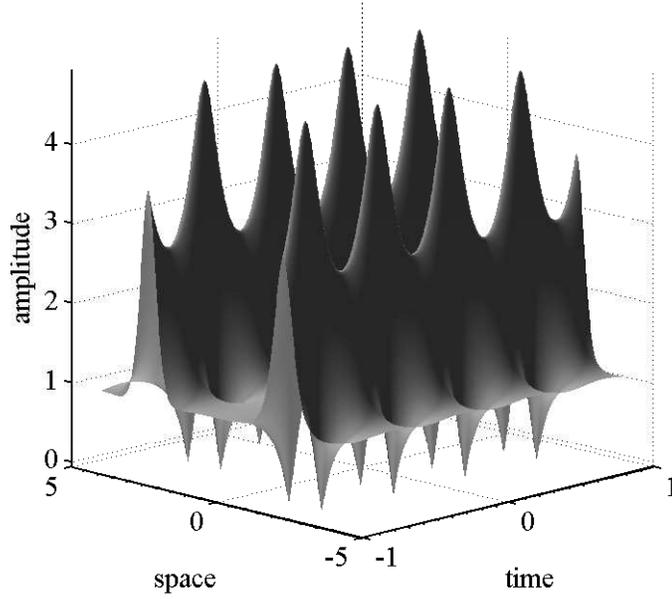


Figure 2.28: Peaked rogue-wave:  $L = 8$ ;  $\lambda = 0.0107 + 1.9373i$ ; and  $A_{max} = 4.87$

Table 2.1: Summary of the peakedness and spectral parameters for the rogue-wave solutions from the upper most band with  $\lambda_R \approx 0$  for  $L$  as indicated.

case	$P_k$	$L$	$(\lambda_R, \lambda_I)$	$\epsilon$	$\theta$
1	0.33	6	(0.0118, 2.551)	0.00259	0.0826
2	0.41	8	(0.0107, 1.937)	0.00252	0.1159
3	0.48	10	(0.0061, 1.549)	0.00259	0.1584
4	0.53	12	(0.0086, 1.402)	0.00260	0.1874
5	0.56	14	(0.008, 1.276)	0.00267	0.2359

of  $L$  (lower  $\lambda_I$ ) in Figure 2.28 exhibits a lower maximum saddle, and thus a smaller difference from the saddle to the background. As one progresses down the branches of the predictive map, the maximum saddle for a solution is found to decrease and the undulation of the peaks become more prominent as shown in the collection of figures. The Peregrine solution represents a limiting case where the maximum saddle and the background are coincident (i.e., 1) and  $L \rightarrow \infty$ .

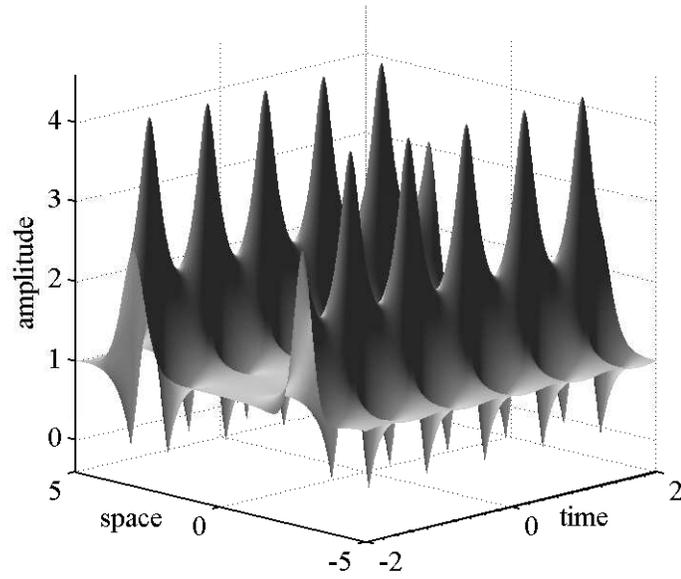


Figure 2.29: Peaked rogue-wave:  $L = 10$ ;  $\lambda = 0.00609 + 1.5485i$ ; and  $A_{max} = 4.2$

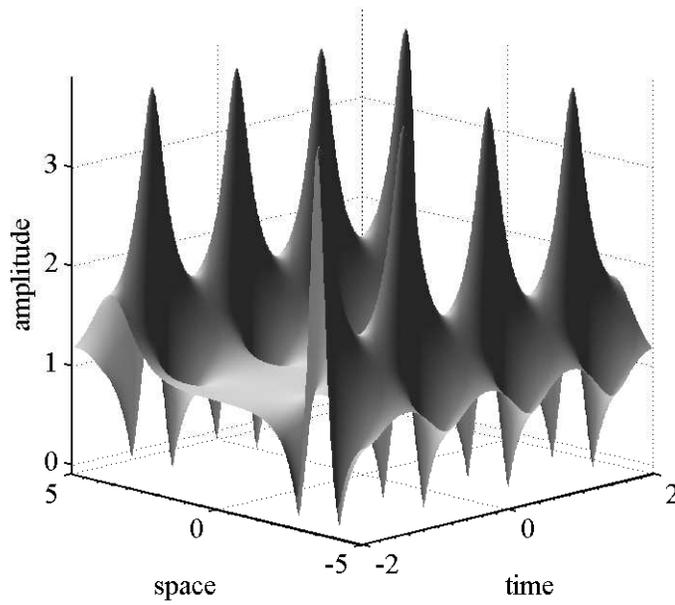


Figure 2.30: Peaked rogue-wave:  $L = 12$ ;  $\lambda = 0.0086 + 1.402i$ ; and  $A_{max} = 3.8$

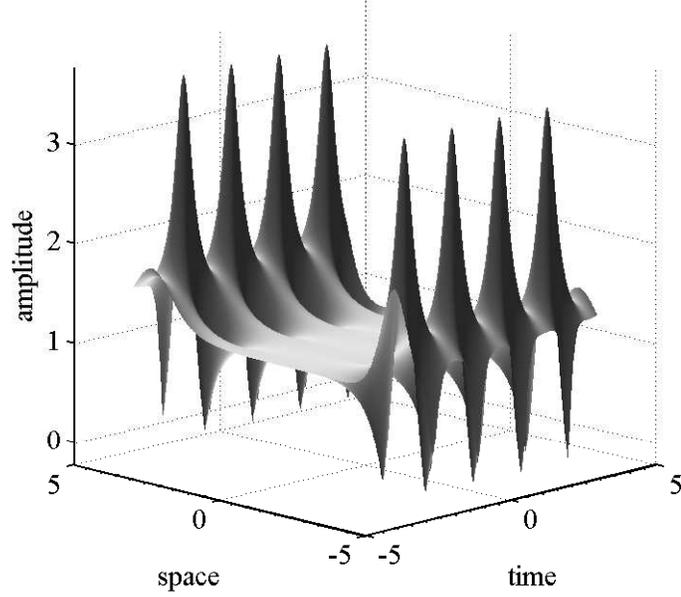


Figure 2.31: Peaked rogue-wave:  $L = 14$ ;  $\lambda = 0.008 + 1.276i$ ; and  $A_{max} = 3.55$

## 2.2.7 Physical Scaling

In order to physically observe the solutions to the NSE, the solutions must be rescaled to physically meaningful dimensions. The dimensional form of the NSE for deep water can be described by

$$i(\psi_t + \frac{\omega_0}{2k_0}\psi_x) - \frac{\omega_0}{8k_0^2}\psi_{xx} - \frac{\omega_0 k_0^2}{2}|\psi|^2\psi = 0 \quad (2.56)$$

where  $t$  and  $x$  are time and spacial coordinates, and  $k_0$  and  $\omega_0$  denote the wave number and the frequency of the carrier wave, respectively. Here, the surface elevation  $\eta(x, t)$  of the water surface is then given by  $\eta(x, t) = Re[\psi(x, t)\exp[i(k_0x - \omega_0t)]]$ . Solutions determined from the predictive map are in nondimensional form, since the governing equation is nondimensional. Several choices exist for rescaling the solutions based on physical length and time scales. The NSE given by equation (2.56)

can be scaled into equation (1.33) by using the rescaling variables

$$T = -\frac{\omega_0}{8k_0^2}t, \quad X = x - \frac{\omega_0}{2k_0}t, \quad \psi = \sqrt{2}k_0^2u$$

where  $X$  is the coordinate in the frame moving with the wave group velocity, and  $T$  is the time variable. It is noted that if  $u(X, T)$  is a solution of equation (1.33), then with the rescaling  $X \rightarrow aX$  and  $T \rightarrow a^2T$ , so is  $au(aX, a^2T)$  where  $a \in \mathbb{R}$ . Chabchoub *et al.* [11] rescaled the Peregrine breather

$$u_p(X, T) = \left(1 - \frac{4(1 + 4iT)}{1 + 4X^2 + 16T^2}\right) e^{2iT} \quad (2.57)$$

by using the transformation

$$aX \rightarrow \sqrt{2}k_0^2a_0(x - c_g t), \quad a^2T \rightarrow -\frac{k_0^2a_0^2\omega_0}{4}t$$

for appropriate reproduction in a water tunnel experiment. Thus, the resulting Peregrine solution is given by

$$\begin{aligned} \psi_p(x, t) = a_0 \exp\left(-\frac{ik_0^2a_0^2\omega_0}{2}t\right) \\ \times \left(1 - \frac{4(1 - ik_0^2a_0^2\omega_0 t)}{1 + [2\sqrt{2}k_0^2a_0(x - \omega_0/2k_0 t)]^2 + k_0^4a_0^4\omega_0^2 t^2}\right) \end{aligned} \quad (2.58)$$

In this case, the rescaling procedure took place in two steps. Here, in the present work, the following single step transform is applied for rescaling to a dimensional form, as this may be more useful for an experimentalist. This collection of transforms is a composite of already existing rescaling procedures presented in previous studies [11, 41].

$$T = -\frac{k_0^2a_0^2\omega_0}{4}t, \quad X = \sqrt{2}k_0^2a_0\left(x - \frac{\omega_0}{2k_0}t\right), \quad \psi = a_0u \quad (2.59)$$

Making use of the above transformations in equation (2.57) leads to the same form of the dimensional Peregrine solution shown in equation (2.58). Thus, the general solution to the dimensional NSE (2.56) can be given by

$$\psi(x, t) = a_0 \frac{\Theta(x, t|\tau, \delta^-)}{\Theta(x, t|\tau, \delta^+)} e^{-\frac{k_0^2 a_0^2 \omega_0}{2} t}. \quad (2.60)$$

where  $\Theta(x, t|\tau, \delta^\pm)$  is determined by substituting equation (2.59) in equation (2.43).

The rescaling procedure with  $(a_0 = 0.01m, \omega_0 = 10.7s^{-1}, k_0 = 11.63m^{-1})$  was applied to the near Peregrine case of Figure 2.23, phase shifted to align its peak location to  $t = 0$ . It is compared to the Peregrine case, which was obtained from equation (2.58). The two solutions appear to be identical, as shown along with the surface elevation in Figure 2.32. Both solutions exhibit similar maxima and minima. Over a larger domain, the periodic nature of the near Peregrine case would become evident.

### 2.2.7.1 Predicted Evolution of a Dimensional Wave Field

The evolution of a wave field governed by the NSE can be predicted based on the solutions and the dimensionalization procedure presented above. The temporal fluctuations observed at specific spatial locations in the wave field of the dimensionalized near Peregrine case defined in equation (2.60) are illustrated in Figure 2.33. The wave field matches closely with that observed in previous work [11], as similar dimensional parameters are used. The near Peregrine case could serve as the basis for an experimental investigation into rogue waves in a similar manner

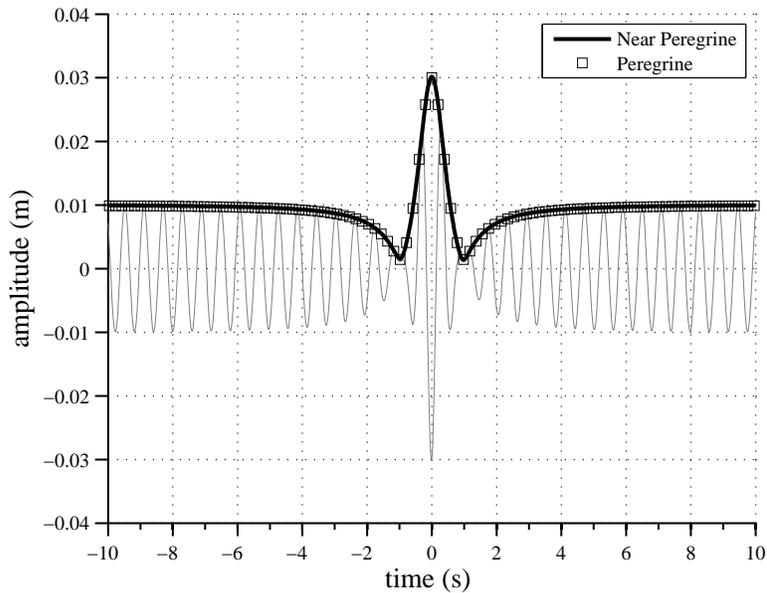


Figure 2.32: Near Peregrine solution (solid line) appears identical to the Peregrine solution (marked with squares) over the interval examined.

as the Peregrine case [11]. The expected result would be quite similar to the pure Peregrine case, with the exception of periodicity in the near Peregrine case. It is believed that further valuable insights can be gained by studying a solution which is far from the Peregrine case. Such far away solutions could potentially have quite different behavior.

The rogue-wave solution presented in Figure 2.22, has been rescaled with parameters  $a_0 = 0.04m$ ,  $\omega_0 = 4.8308s^{-1}$ , and  $k_0 = 2.3813m^{-1}$  for a similar wave field analysis.

The wave field evolution contains several wave packets that mutually interfere with each other, as shown in Figure 2.34. The spatio-temporal localization of wave energy of one of the wave packets is highlighted in the figure at ( $t = 40s$ ,  $distance = 27m$ ). The evolution of this localization can be tracked from a perturbation with two

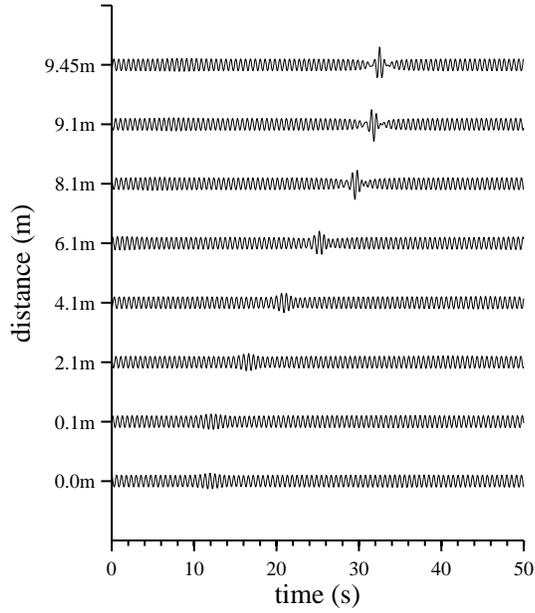


Figure 2.33: Predicted temporal evolution for the near Peregrine solution ( $\lambda_R = 0.0098, \lambda_I = 1.0068$ ). Surface heights are shown at various distances.

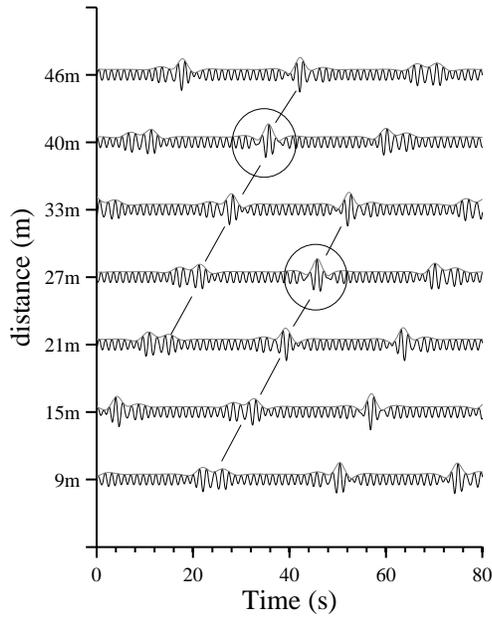


Figure 2.34: A wave field showing the predicted temporal evolution of a rogue-wave solution ( $\lambda_R = 1.1367, \lambda_I = 1.2076$ ) with two localization events highlighted.

distinct peaks close to  $(t = 22s, distance = 9m)$ . This fluctuation localizes into a single large fluctuation at the time highlighted, and then delocalizes at a later time. An additional evolution is highlighted beginning at  $(t = 15s, distance = 21m)$ . A similar localization takes place as the wave packets convect in time and space. The localization leading to maximal amplitude can be identified at  $(t = 38s, distance = 40m)$ .

A detailed view of the surface height and envelope of the first localization is shown in Figure 2.35. While this solution appears to have a similar character to a Peregrine breather, its eigenvalues are quite different, and its peak is more than  $3x$  the surface height, higher than that of the Peregrine solution. This solution is one of many that could be used to gain insights into the degree to which waves in a medium can be modeled by the NSE. Since this wave field is a solution to the NSE, a system which is governed by the NSE should be capable of propagating this solution as predicted. The above wave field predictions may inform an experimentalist to impose an appropriate initial excitation in a wave tank experiment or other medium. Furthermore, similar wave field evolutions may be used to predict a localization event based on limited set of measurements at a single spatial location.

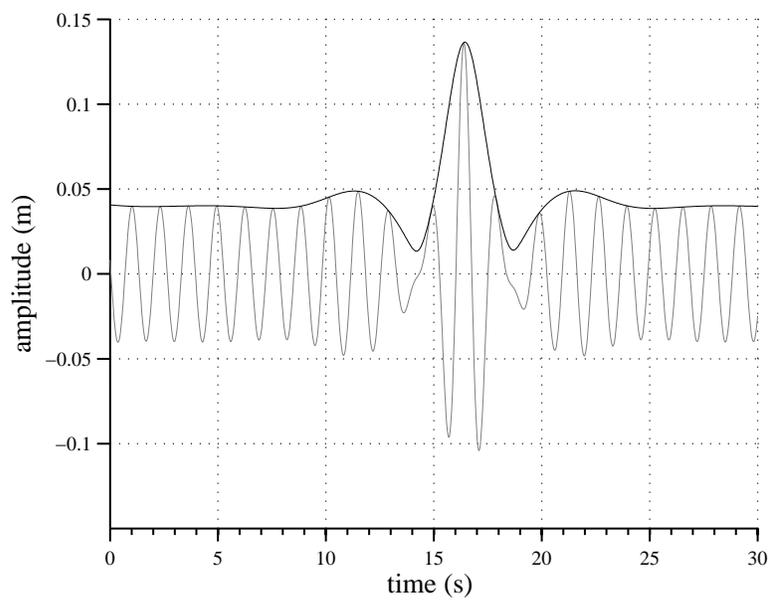


Figure 2.35: Detailed view of dimensional rogue-wave solution with  $\lambda = (1.1367, 1.2076i)$ .

## Chapter 3: Computational Studies of Extreme Energy Localization using Smoothed Particle Hydrodynamics

### 3.1 Literature Review

Rogue waves are commonly defined as waves greater than 2.2 times the significant wave height. These waves occur more frequently than predicted by accepted ocean wave models [6]. While the Benjamin-Fier modulational instability is often cited as the primary mechanism that causes rogue waves, a variety of external interactions can occur in the open ocean [22]. These include wind interactions, linear focusing, and interactions with features of the sea floor [23]. Linear wave focusing can predictably produce waves that are greater than 2.2 times the background wave height, and hence, can be classified as rogue waves [23]. Wave interactions can be better understood, predicted, and modeled through advanced computing resources and modeling efforts. In the current work, the focus is on fundamental simulations of wave interactions. Grid based numerical hydrodynamic simulations are resource intensive, and one can experience difficulties in resolving free surface waves with these simulations [26]. Lagrangian methods, such as smoothed particle hydrodynamics (SPH), in which the domain moves with the material being simulated, offer

several advantages for this type of simulation. First, Lagrangian methods naturally capture the free surface in a hydrodynamic simulation. Second, the simulation domain matches the domain of interest [32]. By contrast, grid based schemes often require the inclusion of cells that only briefly contain useful information; that is, the domain above the free surface that may briefly contain a wave. For these reasons, the authors have chosen SPH to simulate the free surface under several conditions.

The original development of the SPH method was for application to compressible flows. Subsequently, Monaghan [32], Monaghan and Kos [34] indicated the extension to free surface flows using a slightly compressible artificial fluid. This modification, known also as “Weakly Compressible SPH”(WSPH), is being utilized in the present work. In reference [32], Monaghan showed examples of its application to a breaking dam, a bore, the simulation of a wavemaker, and the propagation of waves towards a beach. Arbitrary moving boundaries were included by modelling the boundaries by particles which repel fluid particles as shown in Figure 3.1. In reference [34], Monaghan and Kos described experiments and SPH simulations of the run-up and return of a solitary wave traveling over shallowing water and then onto a dry beach backed by a vertical wall.

In reference [54], Vorobyev used SPH method for studying the hydrodynamics processes related to nuclear engineering problems. The numerical model included the XSPH correction (as discussed in Section 3.2.5) which is incorporated in this work.

In reference [13], the authors use SPH to reproduce linear focusing induced extreme waves to study the dynamics of wave breaking process. The authors sim-

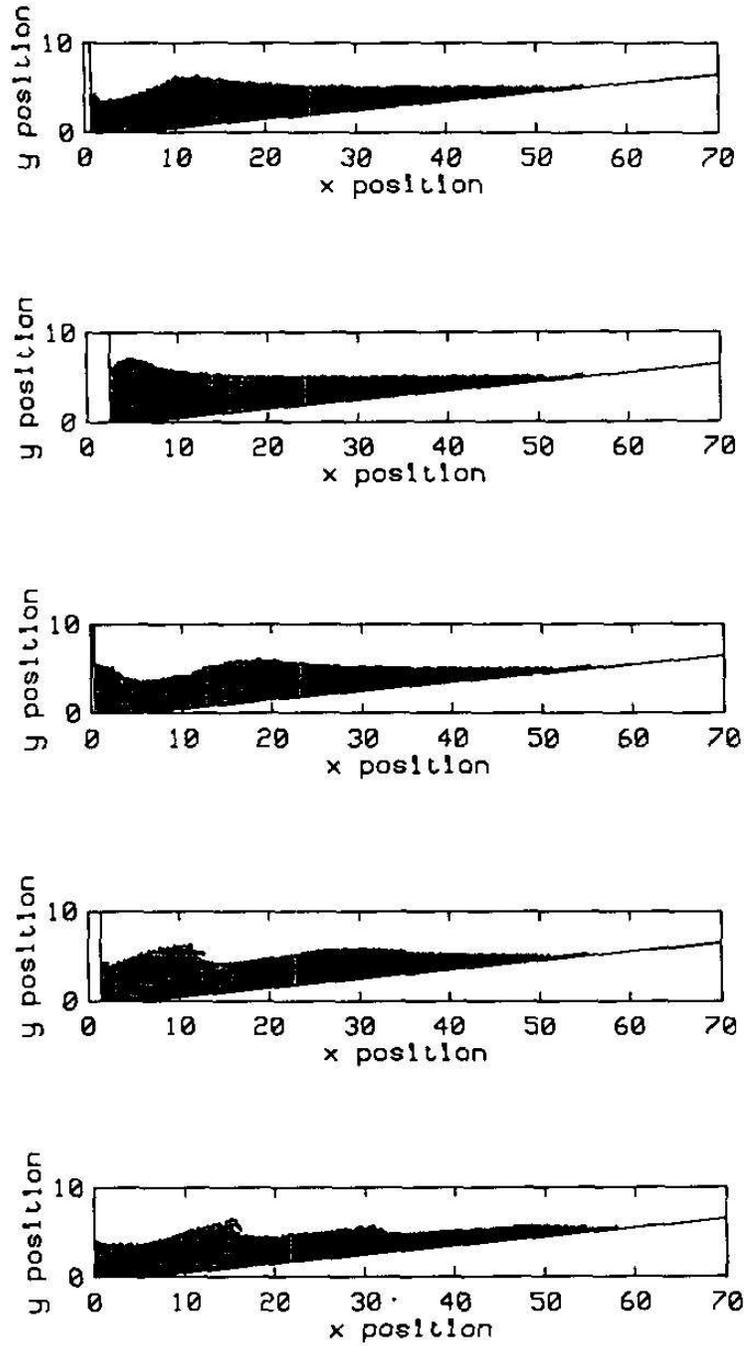


Figure 3.1: SPH simulated waves produced by a wavemaker (Source: [32]).

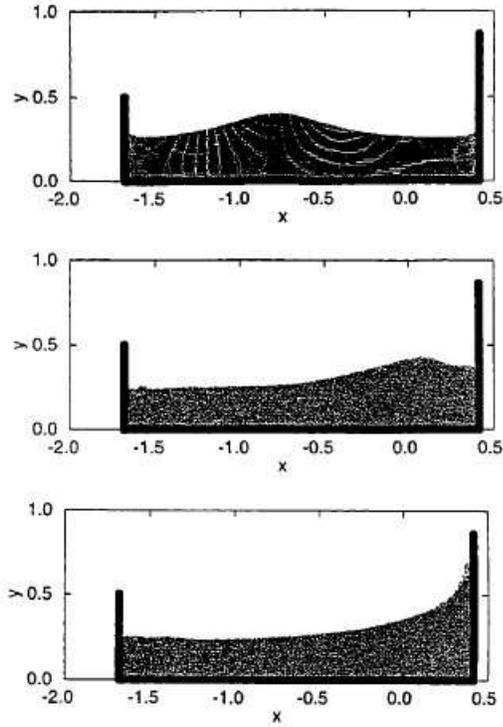


Figure 3.2: Solitary waves in a horizontal tank simulated using SPH (Source: [34]).

ulated a high resolution air-water breaking wave as shown in Figures 3.3 and 3.4 using parallelized SPH code. The parallelization was achieved using Message Passing Interface or MPI.

Lo and Shao [27] studied wave motion impinging on a vertical wall. They found the SPH simulation to agree with analytic predictions of the same event. The impact of a wave on a semi-submersible platform has also been studied by using SPH [45]. In this case, the wave height as it impacted the semi-submersible platform was examined. Wave breaking was also observed.

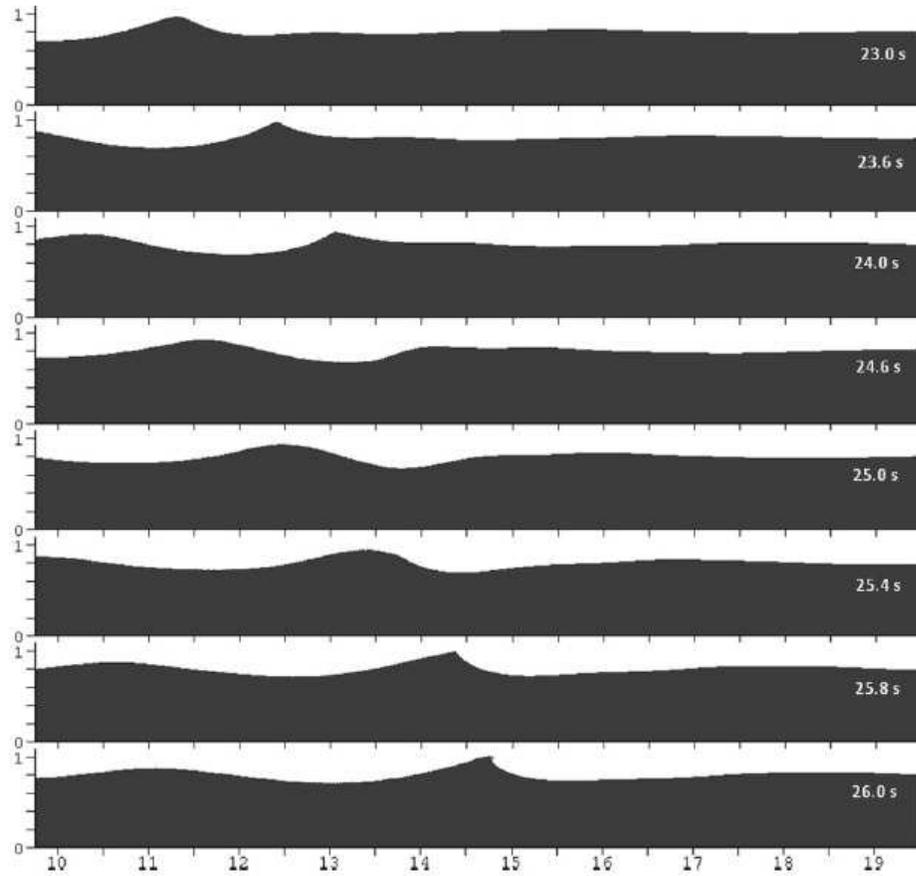


Figure 3.3: Wave packet evolution and focusing (Source: [13]).

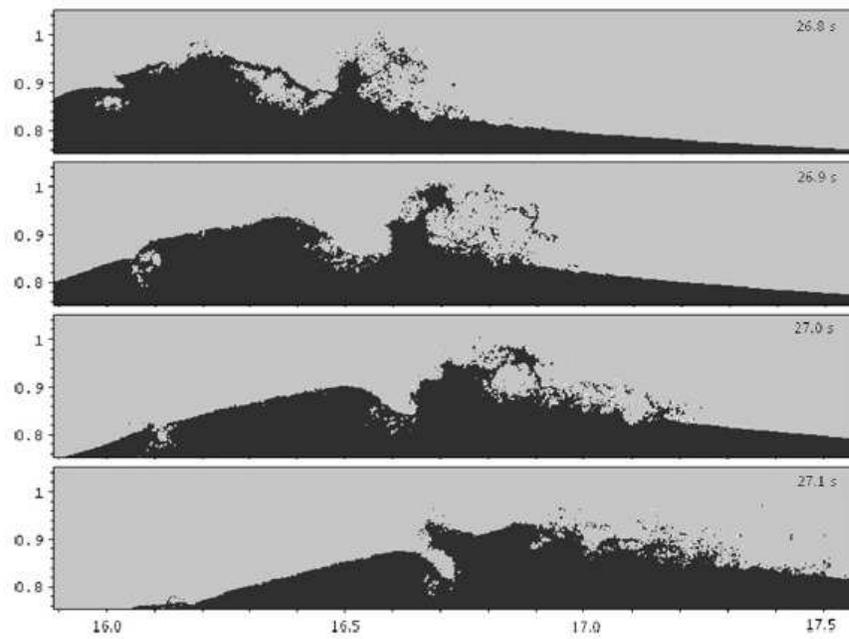


Figure 3.4: Wave breaking and collapsing (Source: [13]).

## 3.2 Smoothed Particle Hydrodynamics

An alternative way to describe fluid dynamics problems is through the Lagrangian description. Unlike the Eulerian description, where the system of coordinates is spatially fixed, the Lagrangian approach uses moving coordinates attached to the material. As the computational nodes move with the simulated medium, the Lagrangian approach has several attractive features compared with the Eulerian approach [26]:

- Convective transfer of physical parameters such as mass, momentum, velocity, energy, etc., is simulated natively by the movement of the nodes. The convective term is thus excluded from the governing equations;
- The time history of all field parameters can be easily tracked, since the computational nodes are rigidly connected to the moving material
- There is no need for computational mesh generation, which significantly simplifies the handling of problems with complicated geometries of the computational domain;
- Since the computational nodes move together with the simulated material, the free surfaces and interfaces are treated natively, without applying particular tracking techniques;
- In contrast to the Eulerian description, where the computational mesh should overlap all regions that would be occupied by the moving material during the

computational period, the nodes in the Lagrangian formulation only represent the volume where the simulated medium is located at the beginning of simulation. This allows using a higher resolution to obtain more detailed information about the simulated system. At the same time, the movement of the computational nodes is not limited by the size of the computational domain, and thus the moving parts of material can be tracked, in principal, as far as desired.

Smoothed Particle Hydrodynamics is a Lagrangian based method capable of effectively simulating free surface flows and gravity waves. It is based on an interpolation technique, which allows the value of any function obtained at a given point, using its values at a number of neighboring points. Following Liu G.R and Liu M.B [26], the approximation can be described in two steps: the continuous integral representation (also referred to as the kernel approximation) and the discrete particle approximation. The following integral representation of a function is used:

$$f(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}')\delta(\mathbf{r} - \mathbf{r}')d\mathbf{r}' \quad (3.1)$$

where  $f(\mathbf{r})$  is a continuous function,  $\mathbf{r}$  is radius vector,  $\delta(\mathbf{r} - \mathbf{r}')$  is the Dirac delta function, defined as:

$$\delta(\mathbf{r} - \mathbf{r}') = \begin{cases} 1, & \mathbf{r} = \mathbf{r}' \\ 0, & \mathbf{r} \neq \mathbf{r}' \end{cases} \quad (3.2)$$

In the first step, the delta function  $\delta(\mathbf{r} - \mathbf{r}')$  in equation (3.1) is replaced by the function  $W(\mathbf{r} - \mathbf{r}', h)$ , which is called the smoothing function or smoothing kernel

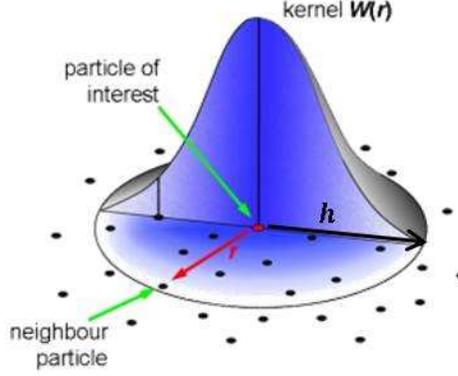


Figure 3.5: Representation of a smoothing kernel in three dimensions.

[31]:

$$f(\mathbf{r}) \cong \langle f(\mathbf{r}) \rangle = \int_{\Omega} f(\mathbf{r}') W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' \quad (3.3)$$

where  $h$  is a smoothing radius,  $W(\mathbf{r} - \mathbf{r}', h)$  is a smoothing kernel, and the angular brackets mark the approximated value of the function of the function  $f$  at the position defined by position vector  $\mathbf{r}$ .

To ensure the correctness of the approximation in equation (3.3), the smoothing function should satisfy several conditions. The first is the normalization condition, which requires the integral of the smoothing kernel over the function domain to be equal to unity:

$$\int_{\Omega} W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' = 1 \quad (3.4)$$

A normalized smoothing kernel ensures invariance of the function  $f$  to changes in smoothing length  $h$  or the number of interactions per particle. The second condition is the delta-function property, which is observed when the smoothing radius approaches zero [26]

$$W(\mathbf{r} - \mathbf{r}', h) \xrightarrow{h \rightarrow 0} \delta(\mathbf{r} - \mathbf{r}') \quad (3.5)$$

In particular cases, besides the conditions in equations (3.4) and (3.5), additional conditions should be applied to the shape of the smoothing kernel, such as the even condition (in which the kernel must be an even function) or compact support domain condition (according to which the kernel must be defined on a compact domain).

In the second step, the continuous integral approximation is converted into the discrete approximation on a number of computational nodes (particles). Following the Lagrangian approach, the continuous medium is represented by a set of particles (computational nodes). For a discrete number of computational nodes, the integration in equation (3.3) can be replaced by a summation, giving the following expression for the function value at the  $i^{th}$  computational node (particle):

$$f_i(\mathbf{r}) = \sum_j f_j W(\mathbf{r}_i - \mathbf{r}_j, h) \Delta V_j \quad (3.6)$$

where  $\Delta V_j = m_j/\rho_j$  is the volume related to the  $j^{th}$  computational node (particle).

After substituting the expression for the volume into equation (3.6), the final SPH approximation of the arbitrarily continuous function can be derived:

$$f_s(\mathbf{r}) = \sum_j f_j \frac{m_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \quad (3.7)$$

where  $m_j$  and  $\rho_j$  are the  $j^{th}$  particle mass and density, respectively;  $f_s(\mathbf{r})$  is the approximated value of the function  $f$  at the point defined by the radius vector  $\mathbf{r}$ . In general, the summation in equation (3.7) is performed over all particles in the computational domain. When smoothing kernels with compact support domains are applied, the summation is limited to a number of neighboring particles.

An approximation of the function gradient  $\nabla f(\mathbf{r})$  is obtained by the use of the gradient of the smoothing kernel. The approximation is as follows:

$$\nabla f_i(\mathbf{r}) = \sum_j f_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \quad (3.8)$$

where  $\nabla W(\mathbf{r}_i - \mathbf{r}_j, h)$  is the gradient of the kernel function. The gradient of the kernel function is calculated using an algebraic derivative of the kernel:

$$\nabla W(\mathbf{r}_i - \mathbf{r}_j, h) = \frac{(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|} \frac{\partial W(\mathbf{r}_i - \mathbf{r}_j, h)}{\partial(\mathbf{r}_i - \mathbf{r}_j)} \quad (3.9)$$

where  $\frac{\partial W(\mathbf{r}_i - \mathbf{r}_j, h)}{\partial(\mathbf{r}_i - \mathbf{r}_j)}$  is an algebraic derivative of the smoothing kernel.

The smoothing radius is a key parameter in the SPH approximation. It defines the distance within which particles interact with each other or, in other words, the distance with a non-zero value of the smoothing kernel (the so called support domain of the kernel). In general, a support domain value is a multiple of a smoothing radius value:

$$R_{s.domain} = k.h \quad (3.10)$$

The value of the constant  $k$  is determined by the choice of the smoothing kernel. For a common case with  $k = 2$ , particles separated at a distance greater than two smoothing radii will have no influence on the parameters at the current point (particle). This is exactly correct when the value of the smoothing function is zero, if the distance to the neighbor point is greater than or equal to  $2h$ .

The choice of the smoothing function has an impact on the accuracy of approximation in equation (3.3). Following is an example of a smoothing kernel based

on the cubic spline functions [31],

$$W(r, h) = C_D \begin{cases} 1 - \frac{3}{2}(r/h)^2 + \frac{3}{4}(r/h)^3, & 0 < r/h \leq 1 \\ \frac{1}{4}(2 - r/h)^3, & 1 < r/h \leq 2 \\ 0, & r/h > 2 \end{cases} \quad (3.11)$$

where  $C_D$  is a constant depending on the number of problem dimensions. For this kernel, the approximation in equation (3.3) has an order of  $O(h^2)$ .

The main equations describing the motion of a viscous incompressible Newtonian fluid are;

- the Navier-Stokes equations (Momentum equation), and
- the Continuity equation.

The Navier-Stokes equations govern the conservation of momentum, while the continuity equation states the conservation of mass. This system of equations can be closed by addition of:

- the Energy equation, and
- the Equation of state.

According to the Lagrangian description, the above equations are written in a coordinate system rigidly connected to a moving medium. This results in the elimination of the advective term in the momentum equation, so long as the system of coordinates moves together with the simulated medium. It is assumed that the computational domain is divided into  $N$  small volumes represented by particles.

Each particle is assigned mass, density, pressure, velocity, acceleration, and other physical parameters. The equations are written for the  $i^{th}$  particle interacting with all other particles representing the system (the  $j$  index).

In order to model solid boundaries in SPH, various methods have been used for different problems. In this work, solid boundaries are modeled as ‘fixed’ fluid particles. The parameters (mass, density, pressure, etc.) of these ‘fixed’ fluid particles are the same as the parameters of the particles representing the liquid medium. During simulation, the interactions of the fluid particles with the wall particles are calculated using equation (3.15) and equation (3.18). Further, coordinates and other parameters are updated only for the fluid particles, while the parameters of the wall particles remain unchanged.

### 3.2.1 Momentum Equation

The equation of momentum conservation for viscous incompressible fluid takes the following form written in the Lagrangian formulation:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla P + V_T + \mathbf{F} \quad (3.12)$$

where  $\mathbf{v}$  is the flow velocity,  $\rho$  is the fluid density,  $P$  is pressure,  $V_T$  is a viscous term, and  $\mathbf{F}$  is the total volumetric force acting on unit mass. The mathematical expression for the derivative of the ratio  $P/\rho$  can be written in the following form:

$$\frac{\nabla P_i}{\rho_i} = \nabla\left(\frac{P}{\rho}\right) + \frac{P_i}{\rho_i^2}\nabla\rho_i \quad (3.13)$$

By using the gradient approximation of the field function (pressure in this case) in equation (3.8) and substituting equation (3.13), we get the following pressure

gradient:

$$\nabla P_i = \rho_i \sum_j m_j \left( \frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W(\mathbf{r}_{ij}, h) \quad (3.14)$$

where  $P_i = P_i^{abs} - P_0$  is the difference between the absolute pressure at a given point and the initial pressure  $P_0$ ,  $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$  is a vector directed from the position of particle  $j$  to the position of particle  $i$ ,  $h$  is a smoothing length (smoothing radius), and  $\nabla_i W(\mathbf{r}_{ij}, h)$  is the gradient of the smoothing function.

Finally, applying the momentum equation (3.12) for the  $i^{th}$  particle constituting the computational domain, and substituting equation (3.14), the equation for the particle acceleration becomes:

$$\frac{d\mathbf{v}_i}{dt} = - \sum_j m_j \left( \frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W(\mathbf{r}_{ij}, h) + V_{Ti} + \mathbf{F} \quad (3.15)$$

$V_{Ti}$  is the viscous term discussed in a later section. For gravitational flows, the only external force  $\mathbf{F}$  is gravity. Where necessary, additional external forces (e.g. wall-on-fluid interaction) can be added to the last term of equation (3.15).

### 3.2.2 Continuity Equation

The continuity equation, which represents the conservation of mass in fluid flow, is usually written in the following form:

$$\frac{d\rho}{dt} = -\nabla \cdot (\rho \mathbf{v}) \quad (3.16)$$

The mathematical expression for the derivative product  $\rho \cdot \mathbf{v}$  can be rewritten as"

$$\rho_i \nabla \cdot \mathbf{v} = \nabla \cdot (\rho \mathbf{v}) - \mathbf{v} \nabla \rho_i \quad (3.17)$$

Using equations (3.8) and (3.17), the final continuity expression in SPH can be written as:

$$\frac{d\rho_i}{dt} = \sum_j m_j \mathbf{v}_{ij} \cdot \nabla_i W(\mathbf{r}_{ij}, h) \quad (3.18)$$

where  $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$  is the relative velocity of the  $i^{th}$  particle w.r.t  $j^{th}$  particle.

The densities are initialized using the following:

$$\rho_i = \sum_j m_j W(\mathbf{r}_{ij}, h) \quad (3.19)$$

### 3.2.3 Equation of State

The equation of state proposed by Batchelor [7] to describe the change of pressure with change of density in real water is used:

$$P = B \left( \left( \frac{\rho}{\rho_0} \right)^\gamma - 1 \right) \quad (3.20)$$

where  $\gamma = 7$ ,  $\rho$  is current density,  $\rho_0$  is the reference density (defined under initial pressure  $P_0$ ), and  $B$  is a constant which is given as

$$B = \frac{\rho_0 c_s^2}{\gamma} \quad (3.21)$$

$$c_s = \frac{V_f}{\sqrt{\eta}} \quad (3.22)$$

where  $\eta = 0.01$  is the compressibility factor,  $c_s$  is the speed of sound corresponding to chosen compressibility and  $V_f$  is the maximal velocity of the fluid medium in the given problem.

The constant  $B$ , also known as the bulk modulus, is a measure of the stiffness of the system and governs the relative density fluctuations during simulations. High

stiffness ensures incompressibility (weakly compressible) of the fluid and low density fluctuations, both of which are crucial for successful free surface gravity wave simulations. Higher order time integration schemes like predictor-corrector and leap-frog allow the use of high bulk modulus. Stable simulations using lower order schemes such as Euler integration are only possible for low  $B$  values which make the fluid compressible and unphysical.

### 3.2.4 Viscosity

The viscous term in the momentum equation (3.15) depends on the second derivative of the velocity. In principle, the direct approximation of the second derivative in the SPH formulation can be obtained using the second derivative of the smoothing kernel. However, the approximation of the second derivative is sensitive to particle disorder, and can lead to instability of the numerical solution [33]. An artificial viscosity was introduced by Monaghan in reference [30] which guarantees stability for the simulations of high velocity flows and for the simulation of free surface. But at the same time it creates high viscous forces and therefore cannot predict correct velocity profiles for low velocity flows. In order to counter these issues, Morris et al. [35] proposed a different estimation of the viscous term given by:

$$V_{Ti}^{Mor} = \sum_j \frac{m_j(\mu_i + \mu_j)}{\rho_i \rho_j} \left( \frac{1}{|\mathbf{r}_{ij}|} \frac{\partial W_{ij}}{\partial r_i} \right) \mathbf{v}_{ij} \quad (3.23)$$

where  $\mu$  is the dynamic viscosity, and  $\frac{\partial W_{ij}}{\partial r_i} = \frac{\partial W(r_{ij})}{\partial r_i}$ . However, Morris's viscosity model could not guarantee suppression of numerical fluctuations. Taking advantage

of both the viscosity models, a combination of both was proposed in reference [55].

The final expression for the combined viscous term reads as following:

$$\mathbf{V}_{Ti} = \begin{cases} V_{Ti}^{Mor} + \sum_j m_j \frac{\beta \mu_{ij}^2}{\bar{\rho}_{ij}} \nabla_i W(\mathbf{r}_{ij}, h), & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ V_{Ti}^{Mor}, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} > 0 \end{cases} \quad (3.24)$$

$$\mu_{ij} = \frac{h \mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{\mathbf{r}_{ij}^2 + 0.01h^2} \quad (3.25)$$

where  $\beta = 0.1$  is a constant,  $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$  is the density averaged between particles  $i$  and  $j$ .

### 3.2.5 XSPH Correction

To close the system of governing equations, an equation of motion (written in the so-called XSPH formulation) is added to the momentum equation (3.15), the continuity equation (3.18) and the equation of state (3.20):

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i + \epsilon \sum_j \frac{m_j}{\bar{\rho}_{ij}} \mathbf{v}_{ji} W(\mathbf{r}_{ij}, h) \quad (3.26)$$

where  $\bar{\rho}_{ij}$  is the average density of the interacting pair of particles ( $i - j$ ),  $\mathbf{v}_{ji} = \mathbf{v}_{ij}$  is the velocity of the  $j^{th}$  particle w.r.t particle  $i$ , and  $\epsilon$  is a constant between 0 and 1. The value of  $\epsilon$  is usually chosen from numerical experience. Using this XSPH correction in the equation of motion makes particle movement more ordered, thus stabilizing the free surface of liquid. It also prevents the unphysical penetration of particles through each other.

### 3.2.6 Density Reinitialization

The evolution equation for the density (equation (3.18)) cannot ensure consistency between mass, density, and area occupied by the particles [21]. To overcome this problem, the density should be periodically reinitialized using a so-called Shepard filter, based on the interpolation technique proposed by Shepard [47]:

$$\rho_i = \sum_j m_j W_{ij}^*(\mathbf{r}_{ij}, h) \quad (3.27)$$

where  $W_{ij}^*$  is defined as:

$$W_{ij}^*(\mathbf{r}_{ij}, h) = \frac{W_{ij}(\mathbf{r}_{ij}, h)}{\sum_j W_{ij}(\mathbf{r}_{ij}, h) \frac{m_j}{\rho_j}} \quad (3.28)$$

This is called the density reinitialization technique.

### 3.2.7 Time Integration

A leap-frog time integration scheme has been implemented to allow higher stiffness and maintain incompressibility in model. The velocities and positions are computed as

$$v_{i+1/2} = v_{i-1/2} + a_i dt \quad (3.29)$$

$$v_{i+1} = v_{i+1/2} + a_i dt/2$$

$$r_{i+1} = r_i + v_{i+1/2} dt \quad (3.30)$$

It should be noted that the leap-frog scheme presented here is a slightly modified version of the conventional formulation. In order to compute the acceleration at time  $t$ , the velocity at time  $t$  is required. However, a leap-frog scheme is used only

to compute velocities at half steps. Therefore, to speed up calculations and reduce step complexity, the integer step velocity  $v_{i+1}$  is computed by taking another half step using the acceleration  $a_i$  (equation (3.29)). This approximation is valid since very small time steps (order of  $10^{-5}$  secs) are used for the current simulations.

### 3.2.8 Parallel Implementation of the Algorithm

The SPH method is in a class of particle methods, which can be accelerated by general purpose computing on a graphics processor unit (GPGPU). The model equations have been implemented in CUDA 6.5, compiled with Visual Studio 2010, and executed on a Tesla C2070 hosted on an Intel Xeon E5607 quad core processor with sufficient RAM, running Win7 x64.

Particle methods such as SPH are well suited to capture “mobile discrete interactions,” which are characterized by elements that undergo local interactions and whose adjacency can change throughout a simulation. A spatial binning algorithm was implemented in order to exploit the local nature of the particle interactions, as detailed in reference [9]. With the help of binning and sorting algorithms to allocate computational resources on mutually close particles, the asymptotic computational complexity can be reduced from  $O(N^2)$  for a full field  $N$  body simulation to  $O(N \log N)$ .

Boundary conditions are enforced by constrained boundary particles. Boundary particles are treated similarly to free particles with the exception that their motions are constrained. For example, the wave maker is simulated by moving the

boundary particles that represent the sides of the tank in a sinusoidal manner rather than according to the equations of motions stated above. Imposing boundary conditions in this manner is common in SPH techniques. It has the benefit that only a single set of equations is required to compute the inter-particle forces. The differences in boundary particles are seen only in the convection equations. However, an explicit no-penetration boundary is absent in the simulation. Therefore, under certain conditions, free particles can penetrate or leak through the walls in the simulation. Several techniques, such as specifically chosen boundary particle spacing, can be applied to mitigate free particle boundary penetration.

A schematic representation of the steps involved in the SPH method is shown in Figure 3.6. Some of the CUDA kernels (implemented in parallel) are provided in Appendix C.1.

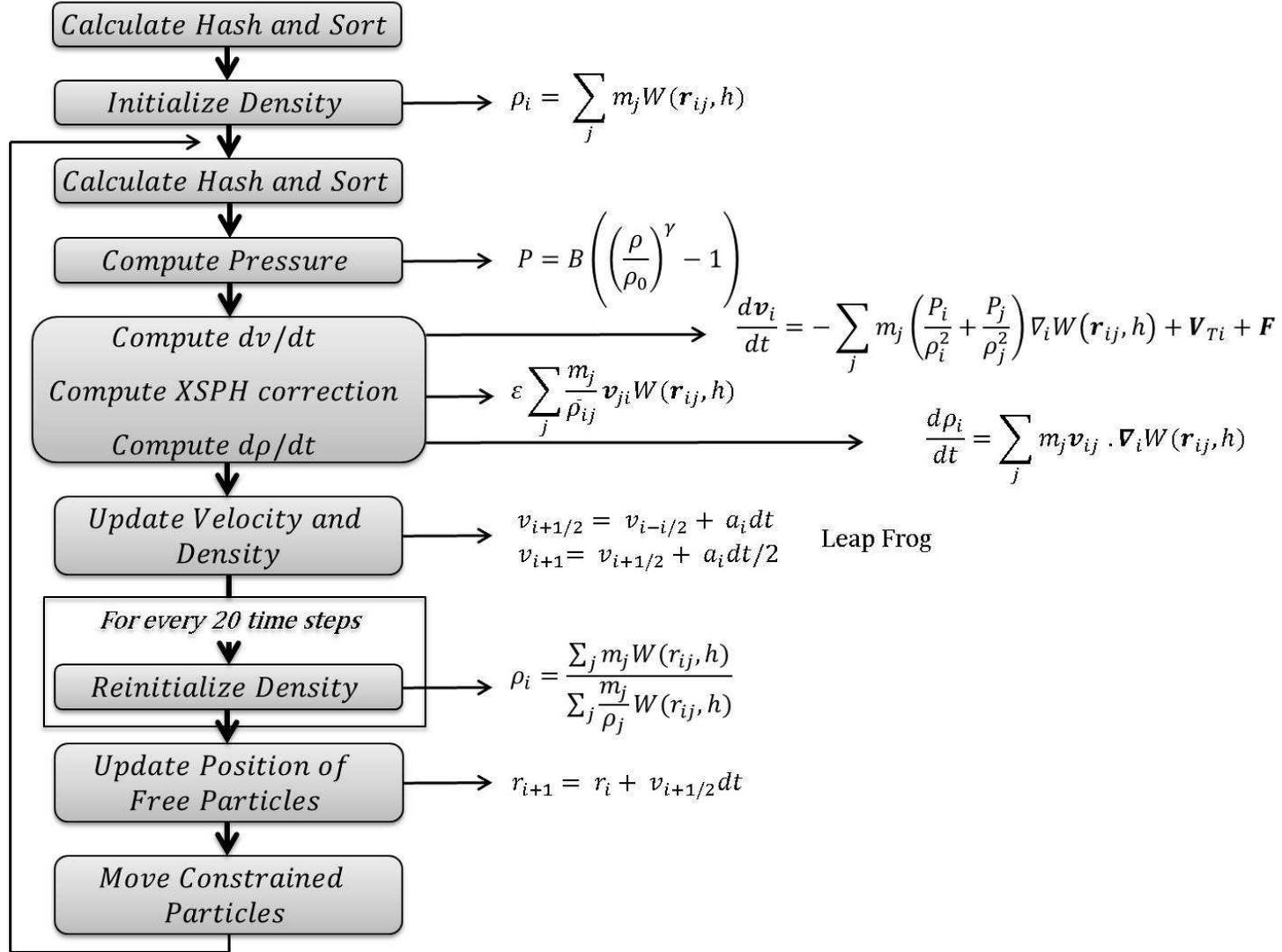


Figure 3.6: Schematic representation of the CUDA-enabled parallel SPH Algorithm.

### 3.3 Numerical Studies in Two-dimensional cases

#### 3.3.1 Smoothing Kernels

The smoothing kernel  $W_{poly6}$  from Muller, Charypar, and Gross [37], adapted for a two dimensional domain is given by

$$W_{poly6}(r, h) = \begin{cases} \frac{4}{\pi h^2} \left(1 - \frac{r^2}{h^2}\right)^3, & 0 < r/h \leq 1 \\ 0, & r/h > 1 \end{cases} \quad (3.31)$$

This smoothing kernel results in a smooth Gaussian like profile that decreases to zero at  $r = h$ , as shown in Figure 3.7. Next, an improved smoothing kernel is used in the pressure force computation, for which the derivative of the kernel is required. This kernel was adapted from Muller *et al.*'s  $W_{spiky}$ . Renormalizing  $W_{spiky}$  for two dimensions helps maintain its characteristic quadratic behavior. A linear response to particle spacing was found to be more desirable for this work, as it helps reduce the intermittent large spikes that occur when particles come into close proximity. The improved smoothing kernel, which has been reformulated to yield a linear response, is given by

$$W_{spiky}(r, h) = \begin{cases} \frac{10}{\pi h^2} \left(1 - \frac{r}{h}\right)^3, & 0 < r/h \leq 1 \\ 0, & r/h > 1 \end{cases} \quad (3.32)$$

$$\frac{\partial}{\partial r} W_{spiky}(r, h) = \begin{cases} -\frac{30}{\pi h^3} \left(1 - \frac{r}{h}\right)^2, & 0 < r/h \leq 1 \\ 0, & r/h > 1 \end{cases} \quad (3.33)$$

$$W_{Improvedspiky}(r, h) = \begin{cases} \frac{6}{\pi h^2} \left(1 - \frac{r}{h}\right)^2, & 0 < r/h \leq 1 \\ 0, & r/h > 1 \end{cases} \quad (3.34)$$

$$\frac{\partial}{\partial r} W_{Improvedspiky}(r, h) = \begin{cases} -\frac{12}{\pi h^3} \left(1 - \frac{r}{h}\right), & 0 < r/h \leq 1 \\ 0, & r/h > 1 \end{cases} \quad (3.35)$$

The linear tendency of the derivative of the improved smoothing kernel as compared to the quadratic response of the derivative of the original smoothing kernel can be identified in Figure 3.7.

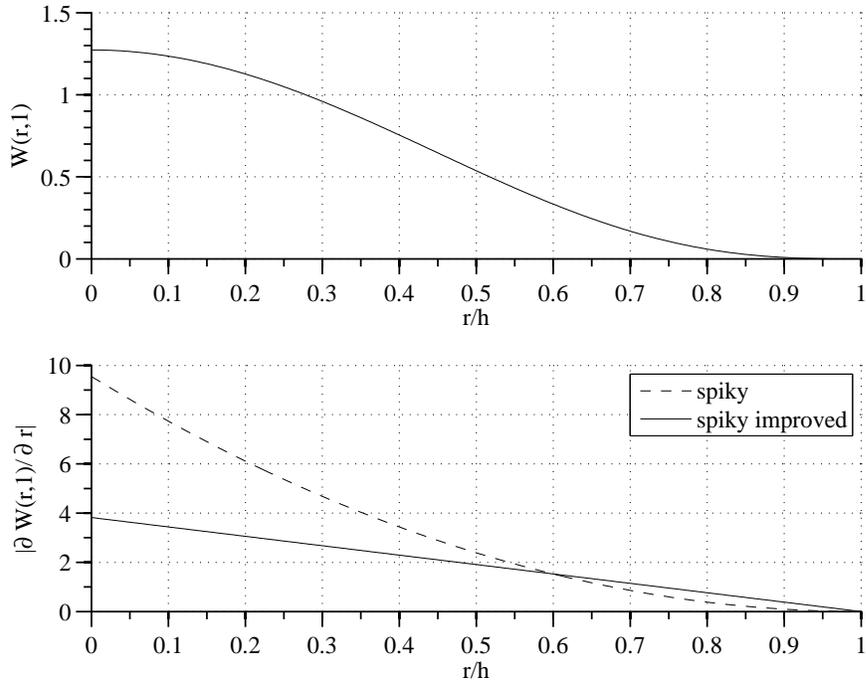


Figure 3.7: The  $0^{th}$  derivative (upper) and  $1^{st}$  derivative (lower) smoothing kernels used in this work.

### 3.3.2 Model Validation

#### 3.3.2.1 Dam Break

Several validation and benchmarking case studies have been carried out in order to validate the computational model. A classic dam break using 7600 free particles is simulated, as shown in Figure 3.8. The tank is chosen to be large enough to contain the fluid. The fluid (water) is allowed to settle down first to form a square column of dimensions  $0.25 \times 0.25$  units at the left end of the tank. The column is then released at  $t = 0$ . As the simulation proceeds, the water column collapses and eventually splashes against the far wall of the tank. After about 1.5 secs of simulated time, the fluid field settles to a uniform surface.

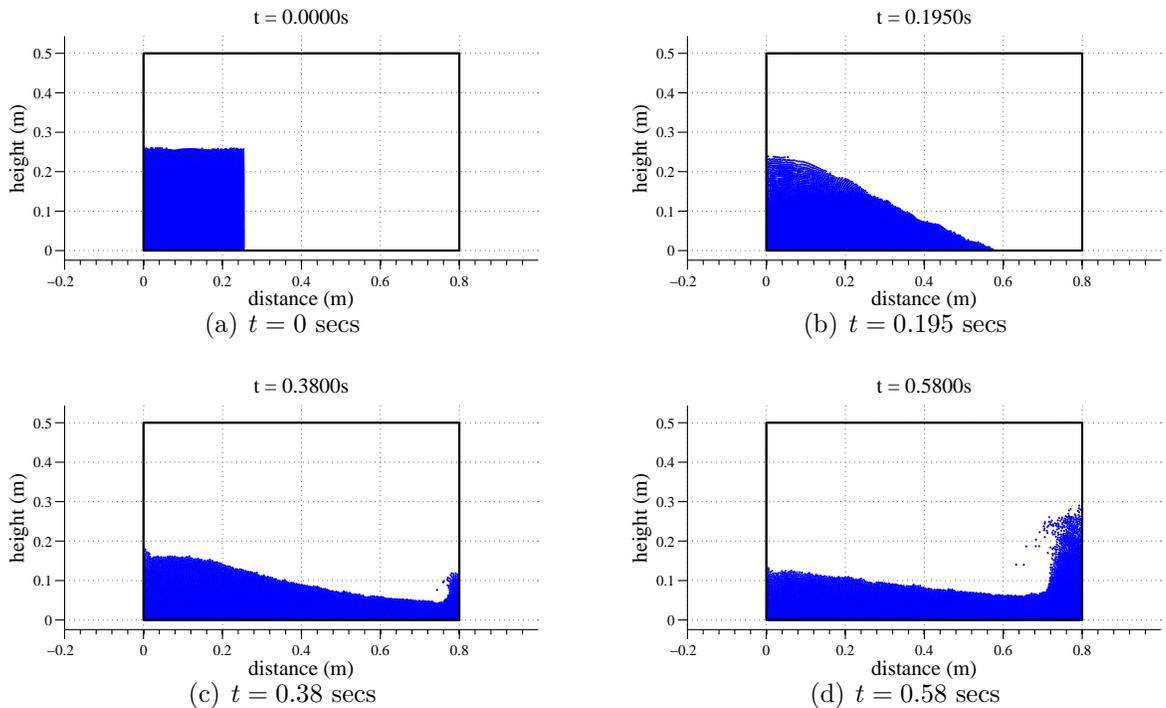


Figure 3.8: The present SPH simulation results exhibit qualitatively similar behavior to previous SPH simulation results for the validation case of a dam break.

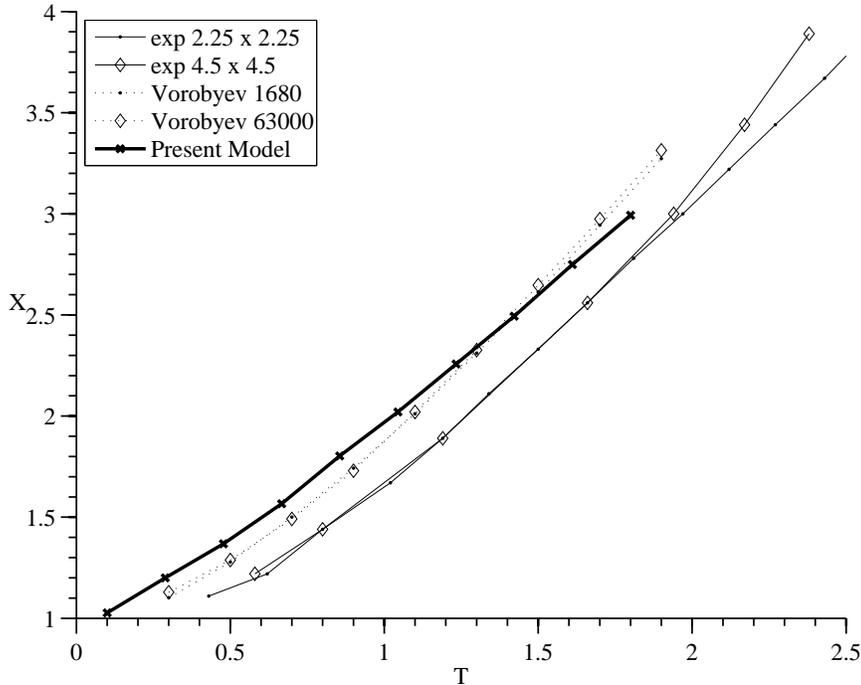


Figure 3.9: For a similar configuration, the results obtained from the present method are seen to be in quantitative agreement with previous experimental [29] and numerical results [54].

The present simulation results compare well with dam break simulations presented in previous published experimental and numerical results. In each case, the distance traversed by the front of the collapsing water column is compared to the elapsed time. Nondimensionalized values of time are defined as  $T = t\sqrt{g/l_o}$ , where  $t$  is wall time,  $g$  is gravity, and  $l_o$  is the initial horizontal width of the water column. The nondimensionalized distance is computed as  $X = x/l_o$ , where  $x$  is the instantaneous distance of the most downstream component of the water column. As shown in Figure 3.10, Martin and Moyce [29] completed a series of experimental studies upon which this dam break configuration was modeled. The results of two experiments are almost identical over the time span of interest as indicated by the solid lines with dots and diamonds in Figure 3.9. In addition, Vorobyev [54],

presented results of his model for a similar configuration shown with dotted lines and dots for 1680 particles and diamonds for 63000 particles as shown in Figure 3.9. Results from the present model, shown with a thick line with dots, agree well with the other numerical model as indicated in the figure. Several parameter variations have been studied including variation of  $\epsilon$  over  $[0.1, 0.25, 0.3]$  and variation of the number of particles. All the variations return similar results to those shown. While a noticeable small offset exists between the experimental results and the present model results, the trends agree well. Overall, the present numerical method shows good quantitative agreement with the other two methods.

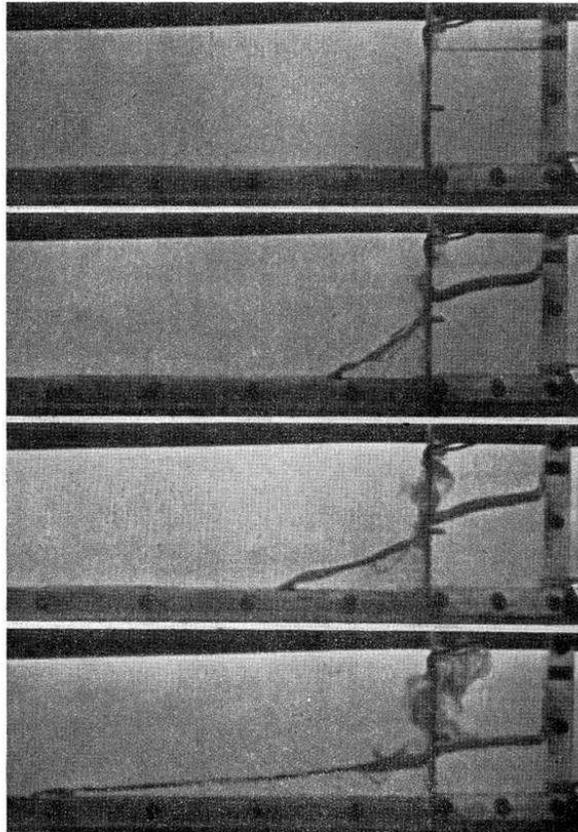


Figure 3.10: Dam break experiment by Martin and Moyce [29].

### 3.3.2.2 Dispersion in the SPH Model

In order to validate dispersion relation allowed by the SPH model, the wavelengths in several numerical experiments are mapped to their corresponding forcing frequencies and compared with predicted values. As shown in Figure 3.11, five different frequencies are chosen for different water depths to realize dispersion for this SPH model. The abscissa of the plot represents ‘Actual  $\omega^2$ ’ calculated directly using the forcing frequencies as  $\omega = 2\pi f$ , where  $f$  is the forcing frequency. ‘Evaluated  $\omega^2$ ’ shown on the y-axis is given by

$$\begin{aligned}\omega^2 &= gk_{SPH}; && \text{Deep Water} \\ &= gk_{SPH} \tanh(k_{SPH}d); && \text{Intermediate Depth} \\ &= gk_{SPH}^2 d; && \text{Shallow Water}\end{aligned}\tag{3.36}$$

where  $k_{SPH} = 2\pi/\lambda_{SPH}$  and  $\lambda_{SPH}$  is evaluated from the simulations as shown in Figure 3.12.

As an example, one of the cases is discussed here. Waves are simulated in a 15 m numerical tank with a water depth of 1 m. The left wall is forced with a sinusoidal function at frequency  $f = 1.0$  Hz. Thus, the generated waves have a temporal angular frequency of  $\omega = 2\pi f = 6.2832$  or  $\omega^2 = 39.478$ . A median measure of the simulated wavelength ( $\lambda_{SPH}$ ) is obtained from the surface profile and zero crossing periods evaluated at each time step. In this case, it is found to be 1.605 m. Since the water depth is greater than half the calculated wavelength, this case qualifies as a deep water simulation. For deep water waves, the dispersion relation is given by

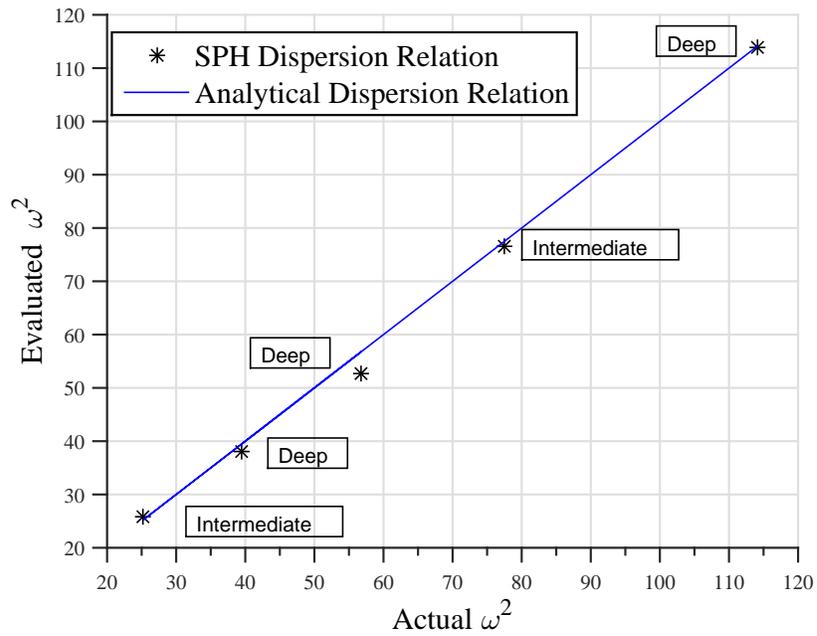


Figure 3.11: Actual  $\omega^2$  versus Evaluated  $\omega^2$ : Data consolidated from five numerical experiments to verify dispersion relation in the current SPH model

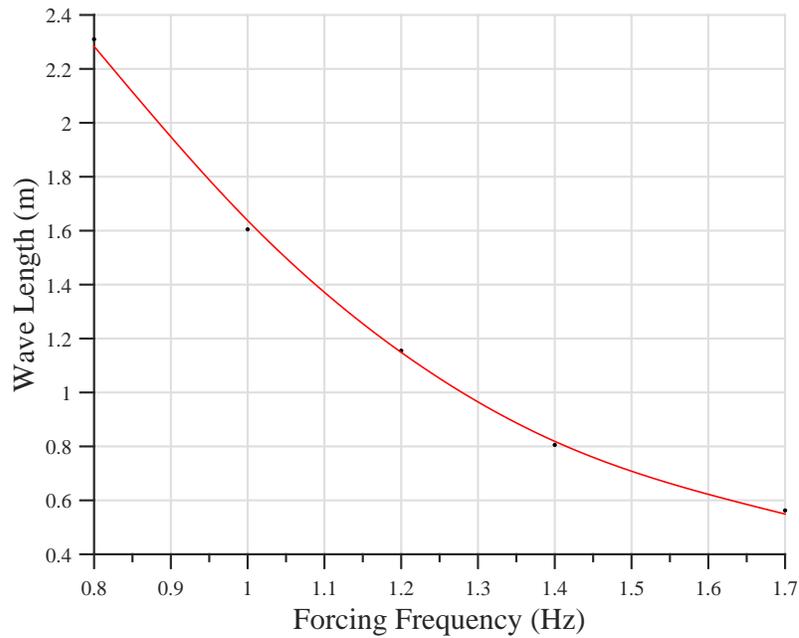


Figure 3.12: Plot showing the variation of SPH simulated wavelength versus the forcing frequency.

$$\omega^2 = \frac{2\pi g}{\lambda_{SPH}} \quad (3.37)$$

Thus the value  $\omega^2$  calculated using the wavelength obtained from the simulation is 38.0094 which has a less than 4% deviation from the predicted value of 39.478. For a forcing frequency of 0.8 Hz for the same tank configuration, the error is less 2%. It is evident that the current SPH model admits dispersion relation, thus making it an effective tool to study extreme energy localizations.

### 3.3.3 Progressive Wave Generation and Dissipation in SPH Model

Although SPH has several advantages over grid based methods in simulating free surface waves, actually generating persistent, progressive waves in a numerical wave tank using SPH is non-trivial. Due to inherent characteristics of the SPH method, for the choice of inappropriate parameters, the waves generated using a paddle or a piston wavemaker tend to dissipate almost immediately even before exhibiting a discernible wave structure. This issue has not been explicitly dealt with in the SPH literature.

Through several numerical experiments it has been realized that dissipation experienced by an SPH model primarily depends on three factors: number of particles ( $N$ ) in the interaction domain defined by the smoothing length, the average measure of the distance between two spatially adjacent particles ( $r_{min}$ ) and the smoothing length ( $h$ ). Although, it might appear that all the three factors imply the same thing, there is a subtle difference. For example, the number of particles

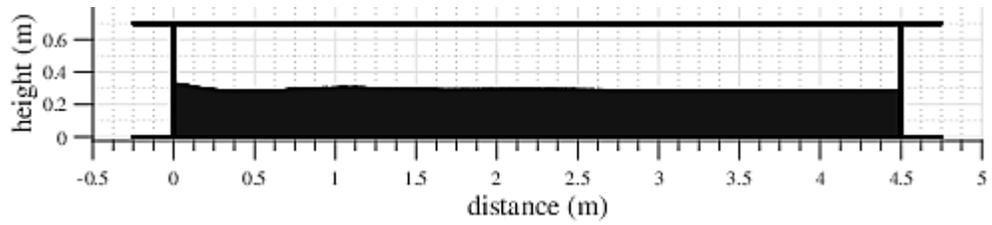
within the interaction domain can be increased or decreased by either changing the smoothing length or by altering the total number of free particles, thus affecting  $r_{min}$ . Through several validation case studies, it was found that the dissipation in an SPH model is inversely proportional to the number of particles in the interaction domain. In other words, the higher the number particles within the smoothing radius, the longer the progressive waves persist.

To illustrate this conclusion, a case study has been presented in Figure 3.13. A 4.5 m numerical wave tank is simulated using 88000 particles for two different smoothing lengths. The water depth for both cases is around 0.35 m, forcing frequency(left wall used as piston wavemaker) 1.4 Hz and stroke amplitude of 2 cm. As shown in Figures 3.13(a) and 3.13(c), the simulation with  $h = 2.8 \times 10^{-3}$  m shows no discernible waves and ones generated at the left wall die out immediately. On the other hand, the wave simulation with  $h = 10^{-2}$  m produce persistent progressive waves as shown in Figures 3.13(b) and 3.13(d). It should be noted that increasing the smoothing length to get a better result is only viable when the tank configuration and the number of particles remain the same. If the size of the tank is increased, a proportional increase in the number of particles would be a better choice than a proportional increase in smoothing length. Thus, the average measure of the distance between two spatially adjacent particles plays a significant role.

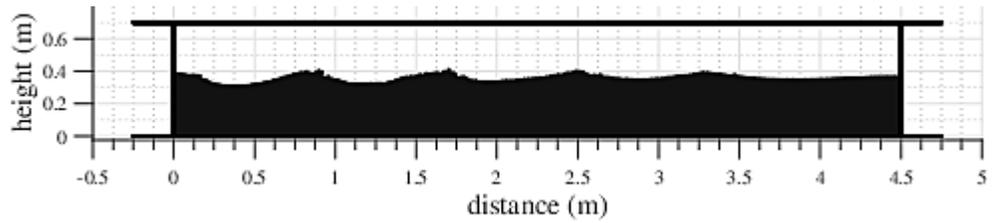
In order to quantify the effect of the three factors  $N$ ,  $r_{min}$  and  $h$ , a metric has been devised. This metric is called the *Interaction Measure* and given by:

$$I = \frac{Nh}{r_{min}} \quad (3.38)$$

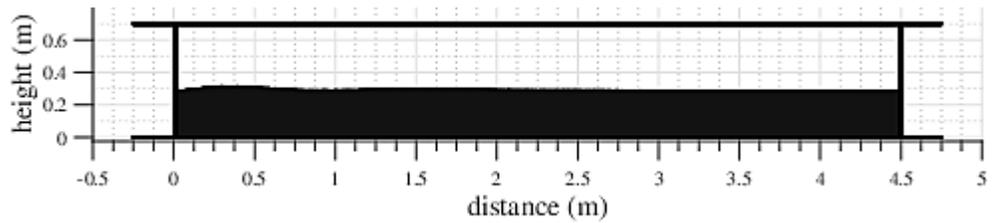
The case with  $h = 2.8 \times 10^{-3}$  m has an Interaction Measure ( $I$ ) = 12,  $N = 8$  and  $roh = r_{min}/h = 0.65$ . However, the simulation with  $h = 10^{-2}$  m has  $I = 465$ ,  $N = 88$  and  $roh = 0.19$ . Thus, higher interaction measure indicates lower dissipation and better progressive wave generation in a numerical SPH wave tank. It must be noted that arbitrarily increasing  $I$  would not continue to produce better results as the effects would converge after a point and further increase in  $I$  would only increase computation time.



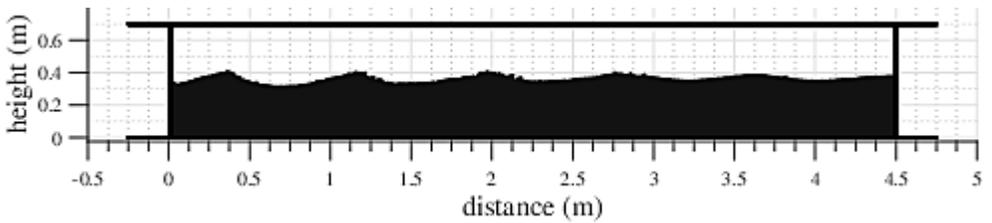
(a)  $t = 5.000$  secs,  $h = 2.8 \times 10^{-3}$  m



(b)  $t = 5.000$  secs,  $h = 10^{-2}$  m



(c)  $t = 6.000$  secs,  $h = 2.8 \times 10^{-3}$  m



(d)  $t = 6.000$  secs,  $h = 10^{-2}$  m

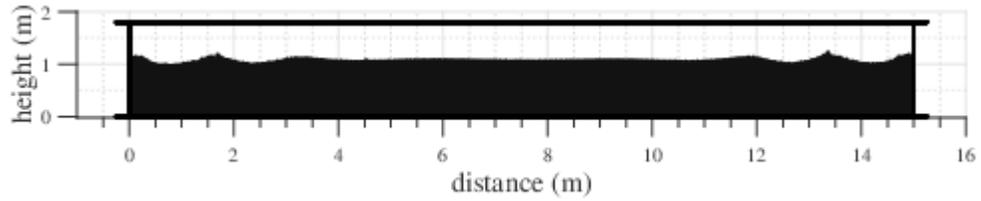
Figure 3.13: Numerical tank of length 4.5 m is simulated by using 88000 particles with water depth of 0.35 m. Left wall is forced with a sinusoidal function of frequency  $f = 1.4$  Hz and stroke amplitude 2 cm. The case study with higher smoothing length ( $h$ ) exhibits considerably lower dissipation.

### 3.3.4 Standing Waves in 1+1 Dimension

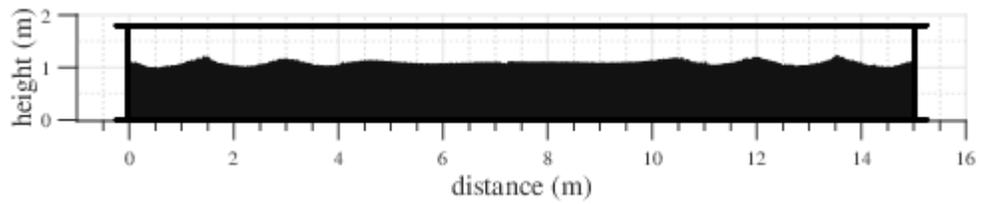
In a numerical experiment, standing waves are generated in a 15 m tank using 90000 particles as shown in Figures 3.14 and 3.15. The left and right walls act as piston wave-makers. Both walls are excited with sinusoidal functions 180 deg out of phase having a frequency of 1.0 Hz and Stroke amplitude of 3 cm. The initial incident wave-fronts traveling towards each other attenuate as shown in Figure 3.14. However, as the waves interfere, they reach a steady state standing wave formation (Figure 3.15). As expected, the waveheight of the standing waves are twice that of the unattenuated incident waves. The execution time for this simulation is approximately 47 ms per time step for a smoothing length ( $h$ ) of  $3.7 \times 10^{-2}$  m.

### 3.3.5 Directional Focusing in 1+1 Dimension

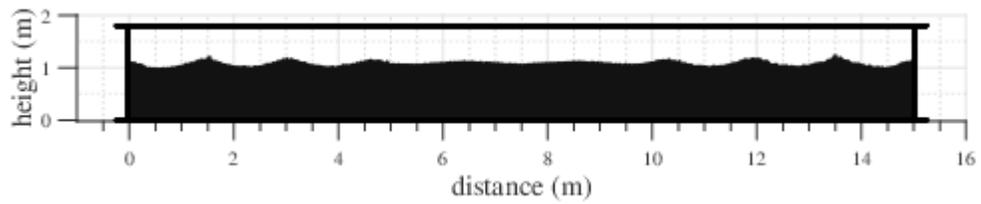
In order to simulate wave interference, a 4.5 m numerical wave tank is simulated using 88000 particles having a water depth of 0.34 m and several numerical experiments are carried out. The simulations take approximately 32 ms per time step to execute for a smoothing length ( $h$ ) of  $10^{-2}$  m. Two case studies for different forcing frequencies and stroke lengths are discussed in this section to demonstrate 1 + 1D directional focusing using SPH. The equilibrium state for the wave tank is shown in Figure 3.20(a). The topology of the wave tank is chosen to allow for waves from opposite ends to convect over the surface and interfere away from the walls. Opposite vertical walls are forced 180° out of phase in a sinusoidal manner. The traveling waves coalesce in the middle of the tank where they exhibit observable in-



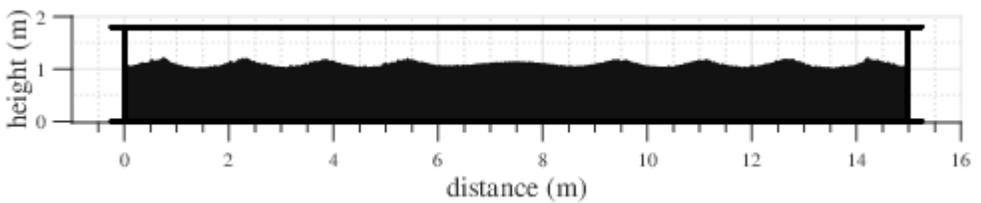
(a)  $t = 4.000$  secs



(b)  $t = 5.865$  secs

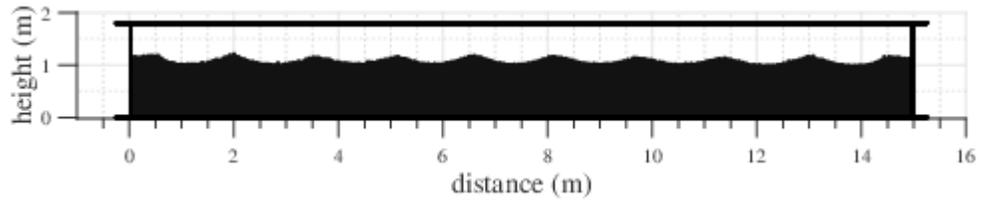


(c)  $t = 6.900$  secs

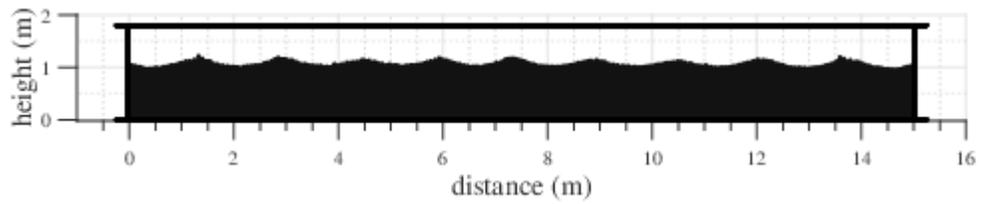


(d)  $t = 8.415$  secs

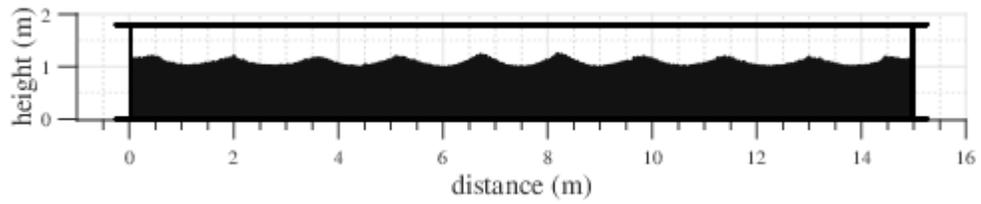
Figure 3.14: 1+1 Dimension Standing Waves Case Study ( $t = 4.000$  secs to  $t = 8.415$  secs): Numerical wave tank of length 15 m and water depth 1.0 m simulated using 90000 particles.



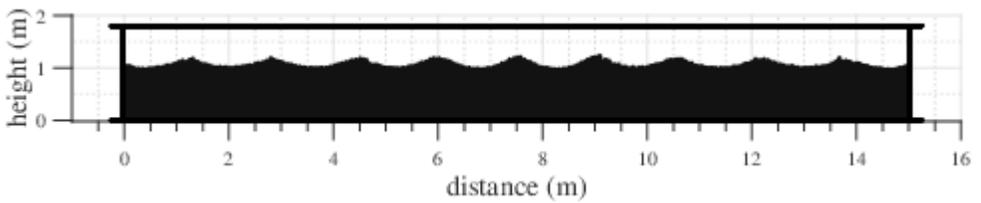
(a)  $t = 10.230$  secs



(b)  $t = 10.800$  secs



(c)  $t = 13.230$  secs



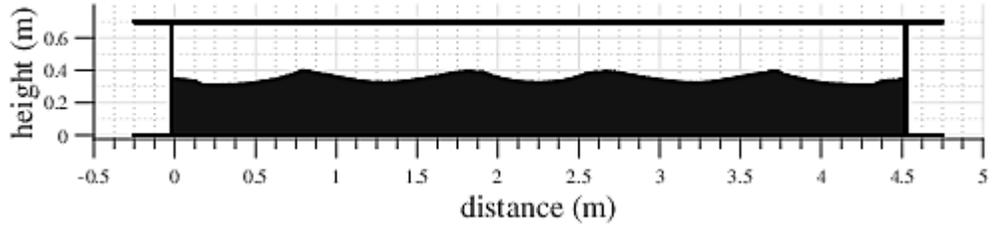
(d)  $t = 13.770$  secs

Figure 3.15: 1+1 Dimension Standing Waves Case Study ( $t = 10.230$  secs to  $t = 13.770$  secs).

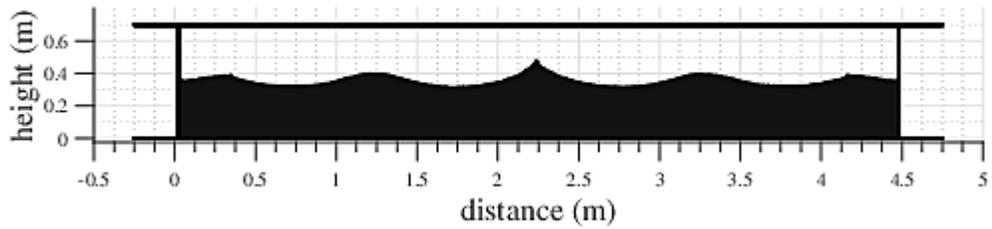
interference behavior. This configuration allows the observation of wave focusing and the associated energy focusing. Validation of the traveling waves with respect to linear wave maker theory is not possible with the current model and configuration. The observed waves are dominated by edge effects and strong interactions with the moving wall.

### 3.3.5.1 Case Study 1: $f = 1.2$ Hz; $S = 2$ cm

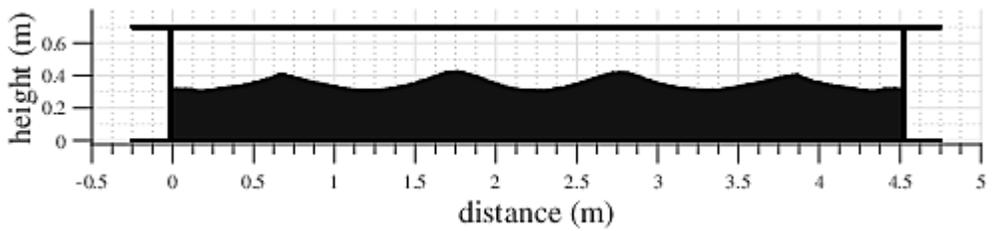
In the first numerical experiment as shown in Figures 3.16 and 3.17, the walls are forced with a sinusoidal function at  $f = 1.2$  Hz and forcing amplitude of 0.02 m or 2 cm. After a warmup cycle, the initial wave-fronts can be seen traveling toward each other in Figure 3.16(a). The wave height of these wave-fronts at  $t = 3.174$  secs is estimated to be 0.043 m as shown in Figure 3.16(a). The waves convect toward each other and focus in the center of the tank. Focusing in this sense is a transient phenomenon wherein a coalesced peak can be observed. A view of this interference is shown in Figure 3.16(b). In this case, the focused wave has a height of 0.5 units, as measured from the base of the tank. As shown in Figure 3.20(a), the mean water level is around 0.35 m. Since the crests of the initial and surrounding waves are at a height of 0.4 m (as measured from the base of the tank) and the initial wave height is 0.043 m, the focused wave height is more than  $3\times$  that of the incident wave fronts. As the system settles down to a steady state, standing waves are formed as shown in Figures 3.16(c) - 3.17(c). As the simulation progresses and the standing waves reach a steady state, it may be noted that the waves exhibit a cnoidal form.



(a)  $t = 3.174$  secs

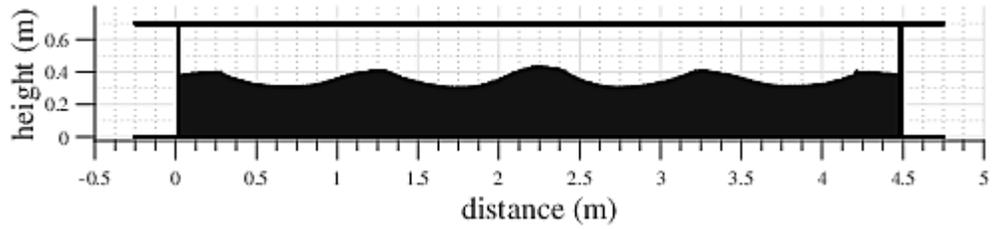


(b)  $t = 3.522$  secs

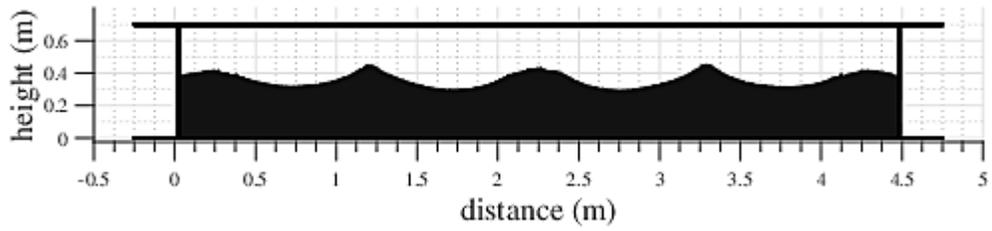


(c)  $t = 3.900$  secs

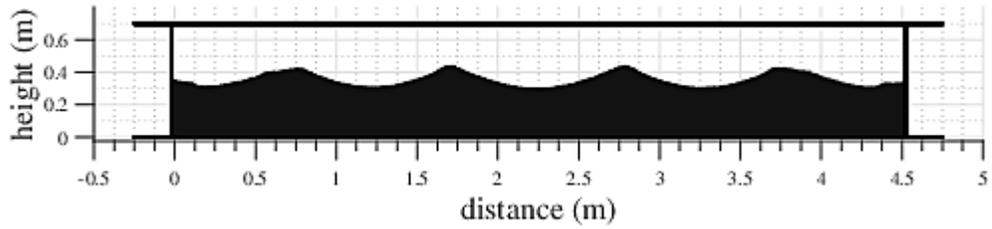
Figure 3.16: 1+1 Dimension Directional Focusing Case Study 1 ( $t = 3.174$  secs to  $t = 3.900$  secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Both, left and right walls excited in a sinusoidal manner with a frequency of 1.2 Hz and stroke amplitude of 2 cm. The wave-fronts interfere in the middle of the tank.



(a)  $t = 4.314$  secs



(b)  $t = 7.740$  secs



(c)  $t = 8.160$  secs

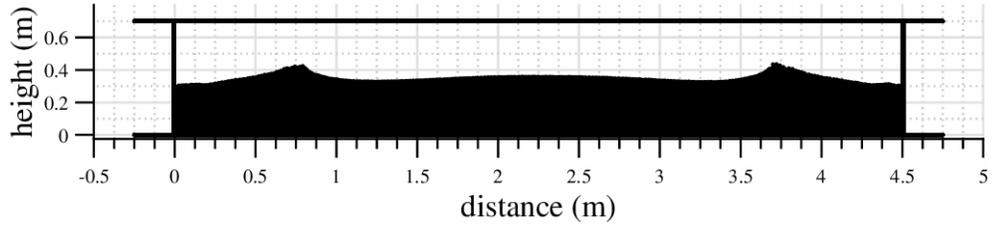
Figure 3.17: 1+1 Dimension Directional Focusing Case Study 1 ( $t = 4.314$  secs to  $t = 8.160$  secs).

### 3.3.5.2 Case Study 2: $f = 1.0$ Hz; $S = 3$ cm

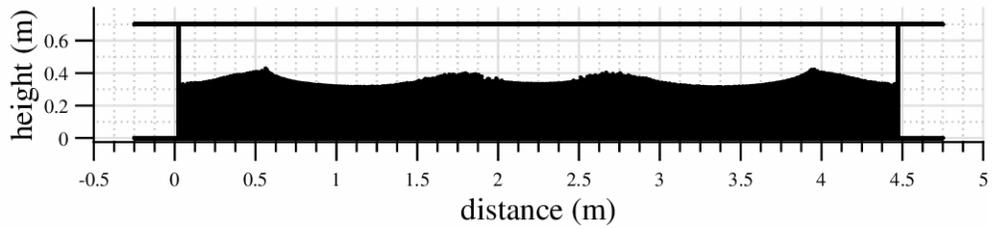
In the second case study shown in Figures 3.18 and 3.19, the walls are forced with a sinusoidal function at  $f = 1.0$  Hz and forcing amplitude  $S = 3$  cm. The initial wave-fronts travel towards each other from opposite directions as shown in Figure 3.18(a). However, due to dissipation, they exhibit slight attenuation before coalescing as seen in a snapshot at  $t = 2.346$  secs (Figure 3.18(b)). The focused wave, shown in Figure 3.18(c), although larger than the incident waves, does not attain a significant height due to the dissipation in the incident waves. Subsequently, as shown in Figure 3.19, the waves exhibit a standing wave pattern with the wave amplitude twice that of the incident waves as expected.

### 3.3.5.3 Case Study 3: $f = 0.8$ Hz; $S = 4$ cm

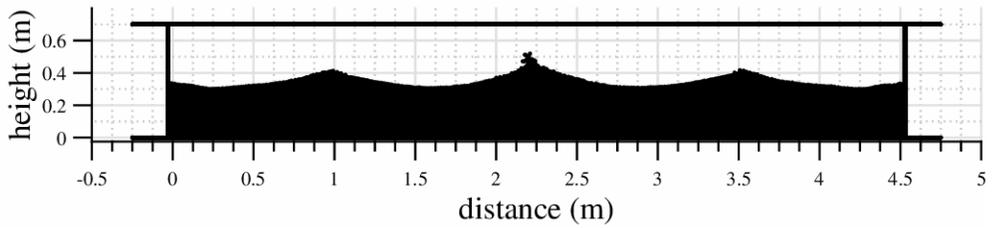
In another case study shown in Figure 3.20, the forcing frequency is  $f = 0.8$  Hz and forcing amplitude is 0.04 m. Longer waves generated in this case do not allow stable standing waves and sloshing effect is observed. However, the focusing of initial wave-fronts can be seen at  $t = 2.820$  secs as shown in Figure 3.20(c). Considering a transient wave height of the incident wave front at  $t = 2.166$  secs (as shown in Figure 3.20(b) of 0.15 m, the focused wave is around  $1.5\times$  higher. However, this is not an accurate measure since the wave breaks as shown in Figure 3.20(c). A breaking wave can also be observed in Figure 3.20(d) at  $t = 3.504$  secs. It should be noted that through numerical experiments in larger wave tanks, the steady state wave height for a 4 cm forcing amplitude was found to be approximately 0.12 m.



(a)  $t = 1.536$  secs

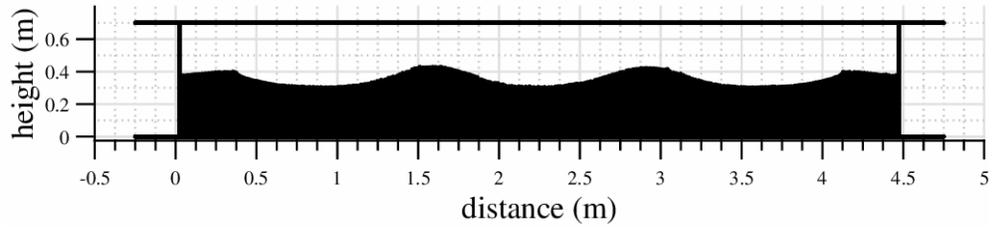


(b)  $t = 2.346$  secs

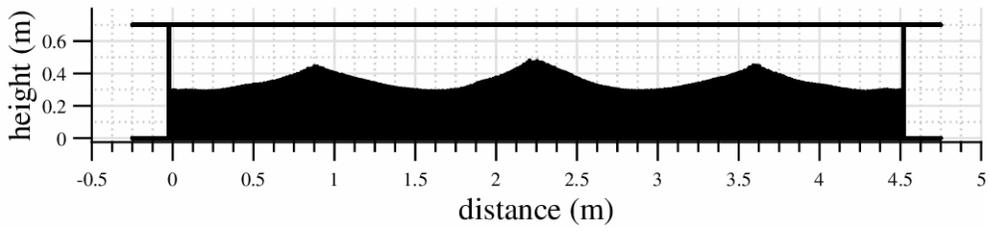


(c)  $t = 2.730$  secs

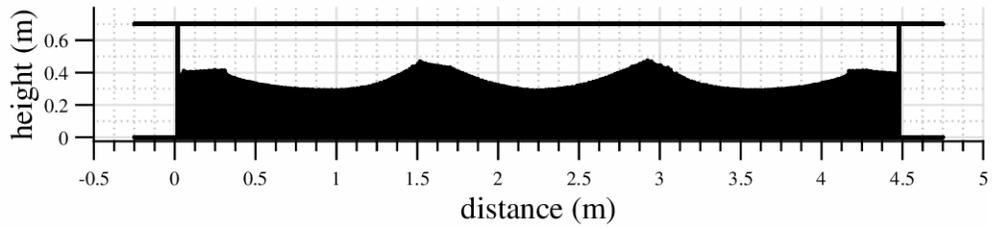
Figure 3.18: 1+1 Dimension Directional Focusing Case Study 2 ( $t = 1.536$  secs to  $t = 2.730$  secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Both, left and right walls excited in a sinusoidal manner with a frequency of 1.0 Hz and stroke amplitude of 3 cm. The wave-fronts interfere in the middle of the tank.



(a)  $t = 3.162$  secs



(b)  $t = 3.642$  secs



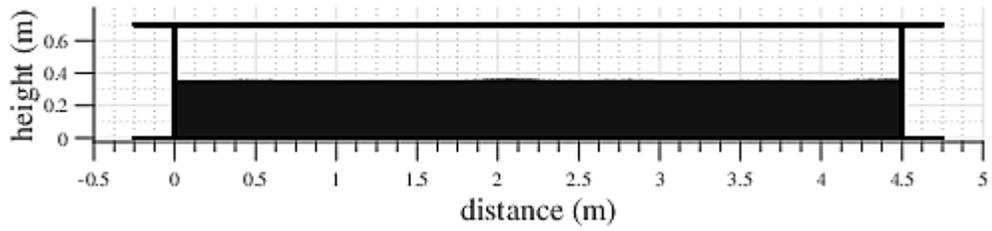
(c)  $t = 4.122$  secs

Figure 3.19: 1+1 Dimension Directional Focusing Case Study 2 ( $t = 3.162$  secs to  $t = 4.122$  secs).

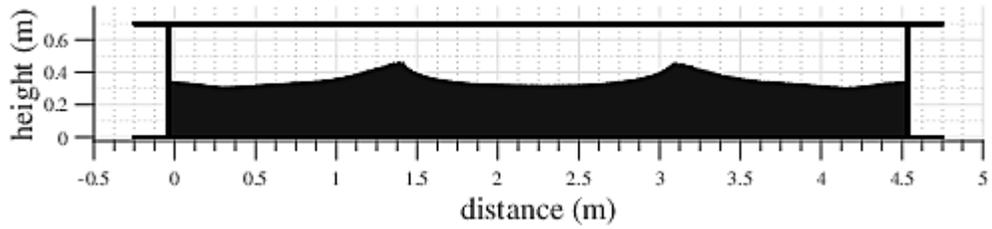
A detailed view of particle velocity for the incident wave is shown in Figure 3.21. Particle motion near the bottom of the tank is sufficiently small, so that interaction with the lower horizontal boundary can be ignored. The particles at the surface of the tank experience the largest velocities, while the velocity magnitude decreases rapidly below the nominal height of the fluid surface for undisturbed flow.

#### 3.3.5.4 Case Study 4: $f = 0.6$ Hz; $S = 5$ cm

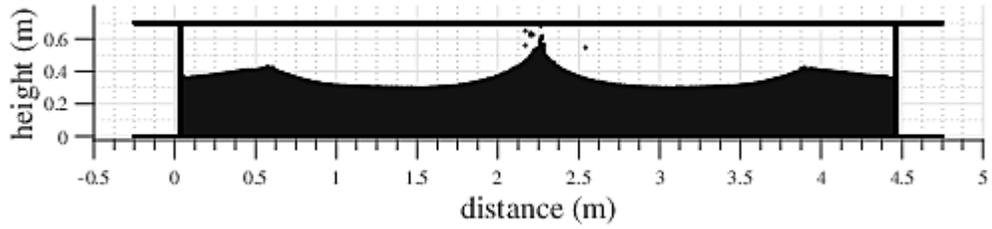
In the final case study discussed in this section, the walls are forced in a sinusoidal manner, 180 deg out of phase, with a frequency of  $f = 0.6$  Hz and a forcing amplitude of  $S = 5$  cm. The incident wave fronts can be seen approaching each other at  $t = 2.424$  secs (Figure 3.22(a)). The transient wave height of these initial wave-fronts measured at this instant is around 7 cm. The wave-fronts focus at the middle of the tank to produce a coalesced wave which is around  $3\times$  higher than the incident waves as shown in Figure 3.22(b). Post interference, the waves travel away from each other in opposite directions and interfere with the incoming wave-fronts to produce two peaks (Figures 3.22(c) and 3.22(d)). A standing wave sort of formation can be observed in Figures 3.22(b), 3.22(d) and 3.23(a). However, since the length of the tank is not large enough, the standing wave formation disintegrates. As more energy is transferred to the system through the wavemakers, the incident waves are much larger (Figure 3.23(b)) than the ones shown in Figure 3.22(a). As shown in Figure 3.23(c), the waves focus, creating a larger and steeper wave that eventually breaks. After this, the waves settle into a standing wave pattern for



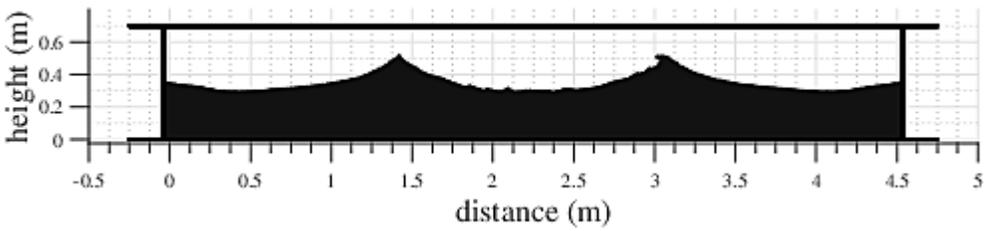
(a)  $t = 0.000$  secs



(b)  $t = 2.166$  secs

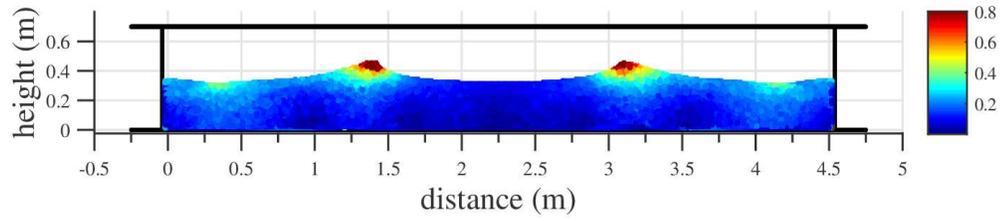


(c)  $t = 2.820$  secs

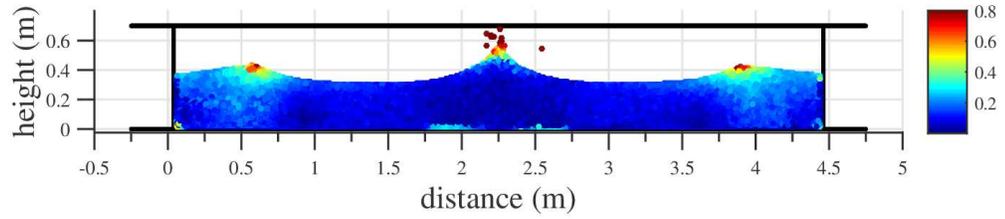


(d)  $t = 3.504$  secs

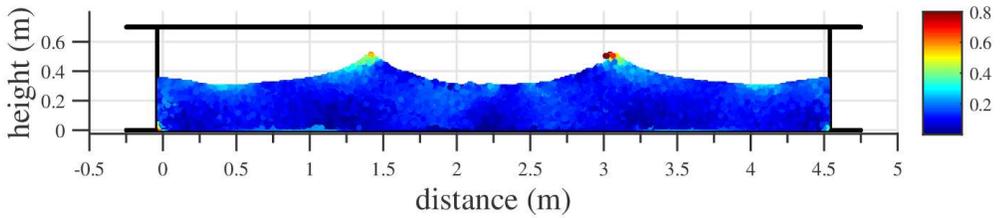
Figure 3.20: 1+1 Dimension Directional Focusing Case Study 3: Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Left and right walls excited in a sinusoidal manner with a frequency of 0.8 Hz and stroke amplitude of 4 cm.



(a)  $t = 2.166$  secs



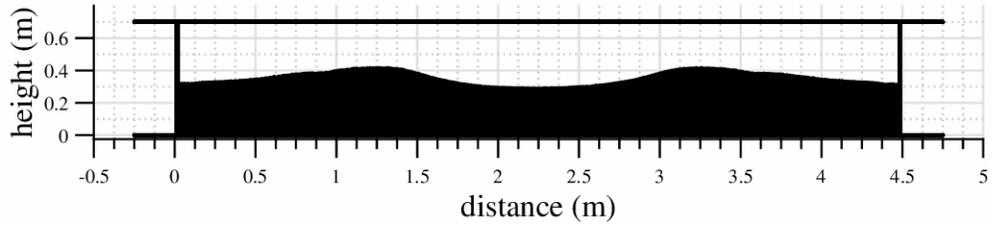
(b)  $t = 2.820$  secs



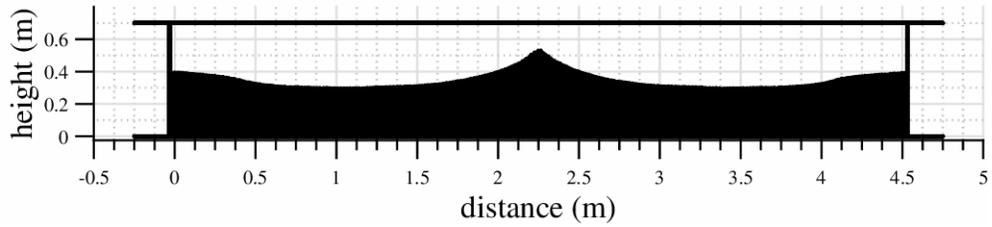
(c)  $t = 3.504$  secs

Figure 3.21: 1+1 Dimension Directional Focusing Case Study 3: Particles colored based on velocity magnitudes.

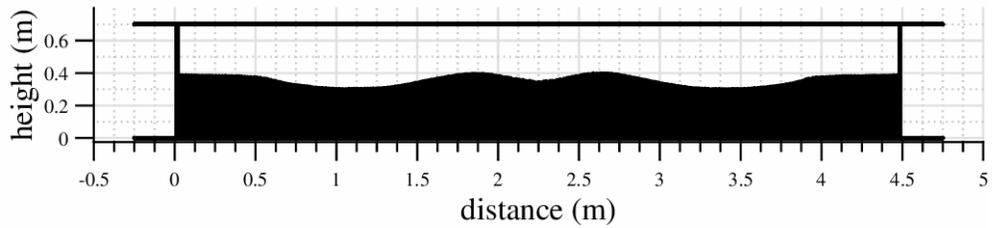
sometime before subsequently exhibiting sloshing.



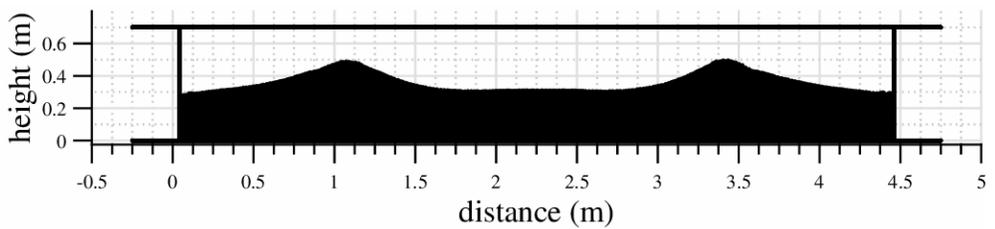
(a)  $t = 2.424$  secs



(b)  $t = 3.156$  secs

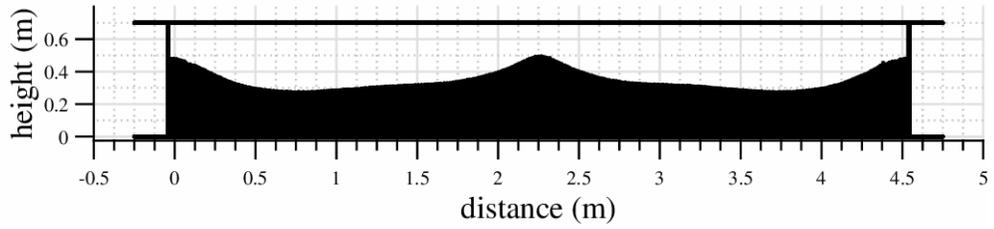


(c)  $t = 3.420$  secs

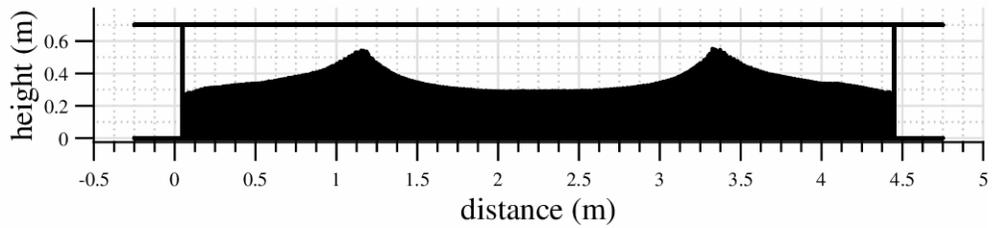


(d)  $t = 3.918$  secs

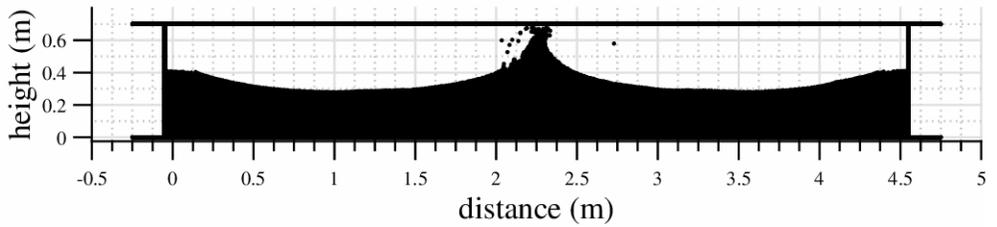
Figure 3.22: 1+1 Dimension Directional Focusing Case Study 4 ( $t = 2.242$  secs to  $t = 3.918$  secs): Numerical wave tank of length 4.5 m and water depth 0.35 m simulated using 88000 particles. Left and right walls excited in a sinusoidal manner with a frequency of 0.6 Hz and stroke amplitude of 5 cm.



(a)  $t = 4.746$  secs



(b)  $t = 5.466$  secs



(c)  $t = 6.246$  secs

Figure 3.23: 1+1 Dimension Directional Focusing Case Study 4 ( $t = 4.746$  secs to  $t = 6.246$  secs).

### 3.3.6 Dispersive Focusing in 1+1 Dimension

Due to dispersion in water waves, as validated in the previous section, sine waves travel with different frequency-dependent velocities. Trailing longer, lower frequency waves overtake shorter, high frequency waves, inducing wave growth due to spatio-temporal superposition. The idea of dispersive enhancement of wavetrains being a possible mechanism for rogue-wave generation was first suggested by Draper [14]. In this section, dispersive focusing is demonstrated using a time-varying excitation rather than a Fourier combination of multiple sine waves [13]. A linear time-varying frequency can be described as

$$\omega(t) = f_0 + kt \tag{3.39}$$

The resulting phase modulation is given by

$$\varphi(t) = \varphi_0 + \int_0^t \omega(\tau) d\tau \tag{3.40}$$

Thus, a wave maker motion corresponding to a frequency described by equation (3.39) would be given by  $\sin(\varphi(t))$ .

For the numerical experiment described in this section, the left wall of a 15 m water tank, with water depth of 1.2 m, is excited in sinusoidal manner with a time-varying frequency of the form

$$\omega(t) = \alpha(k_0 - t) \tag{3.41}$$

This form is chosen so that longer waves fronts are produced after the shorter ones.

For an effective characterization of dispersive focusing, the parameters  $\alpha$  and  $k_0$  are chosen to be 1.63 and 4 respectively. Several case studies have been carried out to arrive at these values. 240000 free particles are used in the simulation and the execution time is approximately 274 ms per time step for a smoothing length ( $h$ ) of  $3.7 \times 10^{-2}$  m.

Integrating equation (3.41), the phase modulation is given by

$$\varphi(t) = \alpha \left( k_0 t - \frac{t^2}{2} \right) \quad (3.42)$$

Therefore, as shown in Figure 3.24, the paddle motion of the left wall is described by

$$\begin{aligned} x(t) &= a_l \sin(2\pi f_0 t) \quad t \leq N/f_0 \\ &= a \sin(\varphi(t)) \quad N/f_0 < t \leq t_f \\ &= x(t_f) \quad t > t_f \end{aligned} \quad (3.43)$$

where  $N$  is the number of constant frequency warm-up cycles before the onset of chirped oscillations,  $f_0$  is the frequency and  $a_l$  is the amplitude of the lead cycles.  $f_0$  is calculated such that there is discontinuity between the lead cycles and the time-varying oscillations.  $a$  is the amplitude of the chirped paddle motion and  $t_f$  is the time at which the wave maker stops. In this case,  $a = 0.05$  m,  $a_l = 0.04$  m and  $N = 9$ .  $a$  is chosen to be greater than  $a_l$  in order to accentuate the focusing effect.

The first wave-front (say wave-front A) produced after the onset of time-varying frequency reaches  $x = 2$  m at  $t = 10.70$  secs, as shown in Figure 3.25(a). The

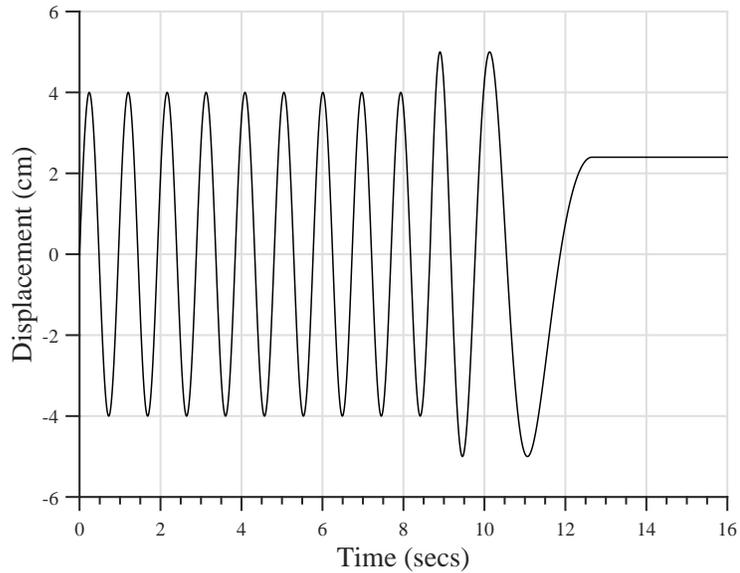
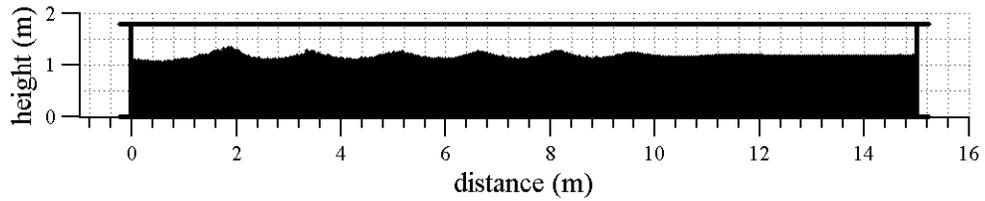


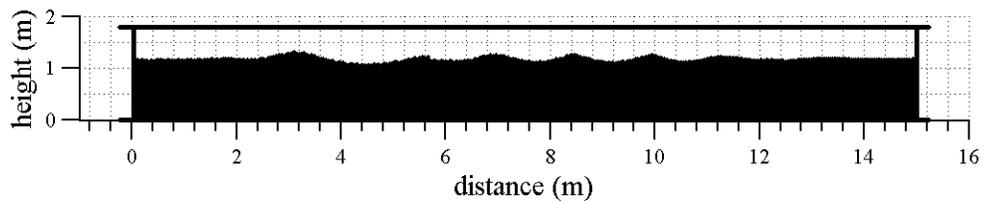
Figure 3.24: Sinusoidal Motion described by the left wall to generate waves. Constant frequency for the first 9 cycles. Frequency variation commences at  $t = 8.65$  secs.

second wave-front (say wave-front B) generated by the chirped motion of the wave maker reaches  $x = 3$  m at  $t = 12.8$  secs as shown in Figure 3.25(b). The coalescing of wave-fronts A and B and subsequent wave growth due to dispersion is captured in Figures 3.25 and 3.26. The initial attenuation of wave-front A, as seen in Figure 3.25(b), is due to the counteracting particle velocities of the lead cycles that are just in front of it. Due to dispersion, the low frequency wave-front B travels faster than A and catches up it, as shown in Figure 3.25(c). Wave-fronts A and B can be seen at around  $x = 8$  m and  $x = 7$  m respectively. At  $t = 15.25$  secs, wave-fronts A and B can be seen at the verge of coalescing (Figure 3.25(d)). Due to interference, the energy of wave-front B is transferred to wave-front A and it exhibits growth as shown in Figures 3.26(a) - 3.26(c). This growth is a transient phenomenon taking place between  $t = 16.10$  secs and  $t = 16.70$  secs while the wave-front travels from

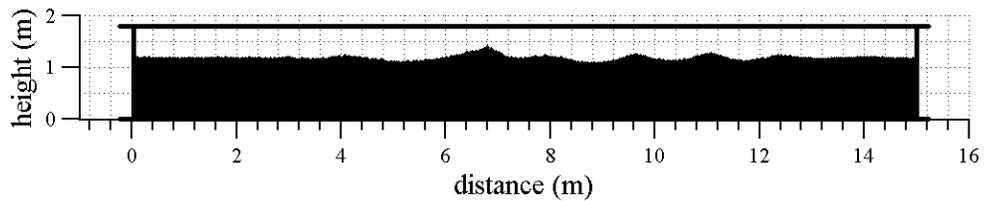
$x = 10.2$  m to  $x = 10.8$  m. The wave begins to break  $t = 16.70$  secs as shown in Figure 3.26(c).



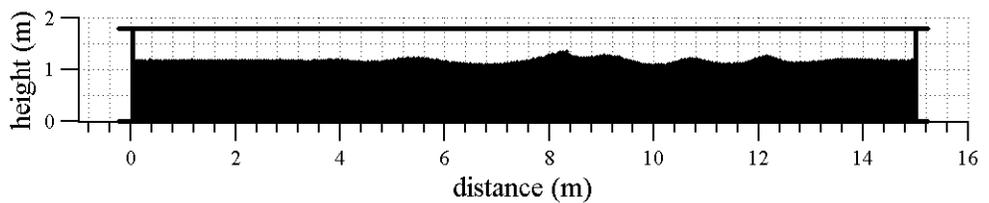
(a)  $t = 10.700$  secs



(b)  $t = 12.800$  secs

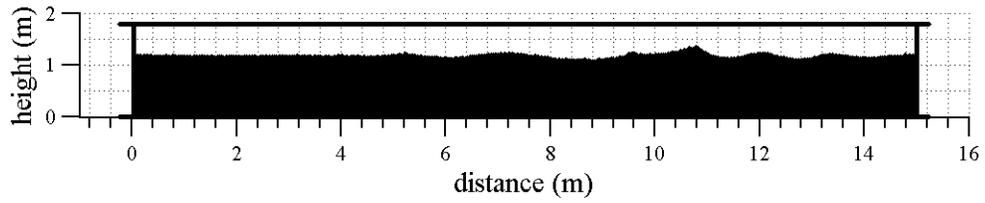


(c)  $t = 14.500$  secs

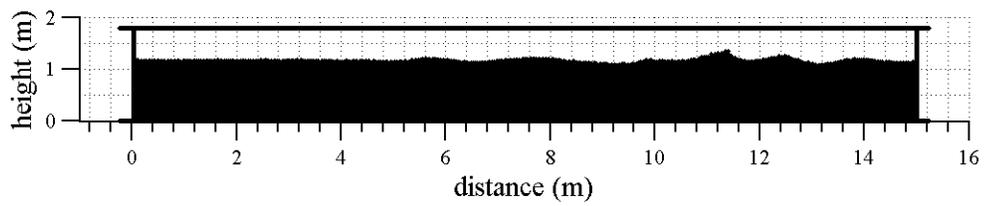


(d)  $t = 15.250$  secs

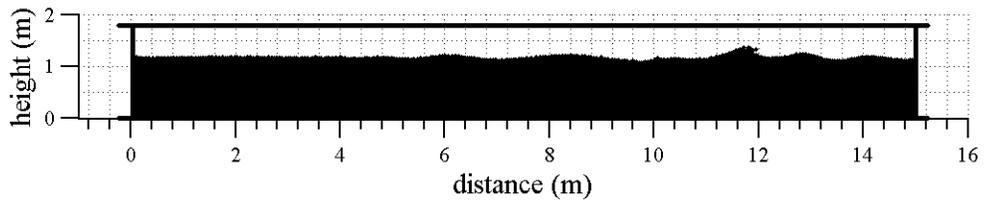
Figure 3.25: 1+1 Dimension Dispersive Focusing Case Study ( $t = 10.700$  secs to  $t = 15.250$  secs): Numerical wave tank of length 15 m and water depth 1.20 m simulated using 240000 particles.



(a)  $t = 16.150$  secs



(b)  $t = 16.450$  secs



(c)  $t = 16.700$  secs

Figure 3.26: 1+1 Dimension Dispersive Focusing Case Study ( $t = 16.150$  secs to  $t = 16.700$  secs): Numerical wave tank of length 15 m and water depth 1.20 m simulated using 240000 particles .

### 3.3.7 Modulational Instability in 1+1 Dimension

By using the insights gained from the study of the Nonlinear Schrödinger equation, the next step is to carry out computational simulations to derive further insights into complexity of extreme wave conditions. Thus, the objective of this section is to identify and qualitatively realize modulational instability through SPH-based numerical experiments.

In Section 2.2.7.1, the predicted evolution of a dimensional wave field has been described. The techniques described in that section can be used to induce a localization based on a single mode solution of the NSE. For the purpose of this study however, the analytical dimensional form of the Peregrine breather has been used to investigate modulational instability. The dimensional form of the Peregrine breather is given by

$$\psi_p(x, t) = a_0 \exp\left(-\frac{ik_0^2 a_0^2 \omega_0 t}{2}\right) \times \left(1 - \frac{4(1 - ik_0^2 a_0^2 \omega_0 t)}{1 + [2\sqrt{2}k_0^2 a_0(x - \omega_0/2k_0 t)]^2 + k_0^4 a_0^4 \omega_0^2 t^2}\right) \quad (3.44)$$

where  $\omega_0$ ,  $k_0$ ,  $a_0$  are the angular frequency, wave number and amplitude of the carrier wave respectively. As shown in Figure 3.29, the Peregrine solution breathes (reaches its maximum height) at  $x = 0$  m and  $t = 0$  secs. Moreover, the solution is symmetric about  $x = 0$  m. Hence, theoretically, using an appropriate variable transformation, the localization can be shifted to a desired location. In such a configuration where the localization target has been shifted, the time history characterized by equation

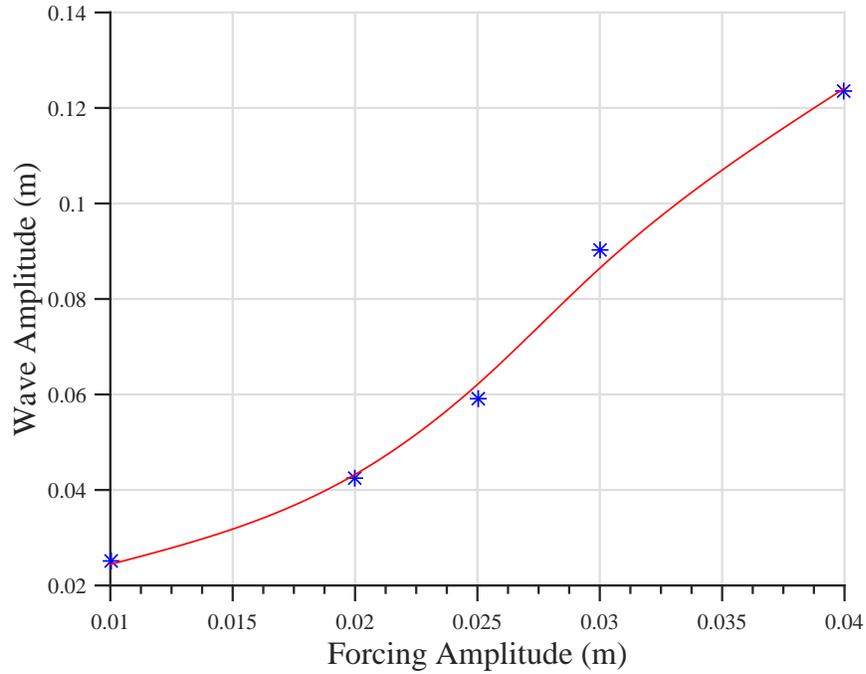


Figure 3.27: Parametric study showing generated wave amplitude versus wave maker amplitude for different water depths.

(3.44) at  $x = 0$  governs the wave maker motion. In other words, a wave maker motion (perturbation at  $x = 0$  m) described by Figure 3.29(a) (or Figure 3.29(g)) would theoretically grow and produce localization at  $x = 8.1$  m.

A parametric study has been carried out to determine the amplitude and wavelength of generated waves as a function of the wave maker stroke length for a given excitation frequency and water depth. The results are shown in Figure 3.27 and Figure 3.28 respectively. According to this study, in order to produce a carrier wave of amplitude (say) 0.025 m, the wave maker stroke amplitude should be 0.01 m. A diagnostic case study is presented where the wave maker motion time history is shown in Figure 3.32 and the theoretical predicted evolution of the initial perturbation (according to equation (3.44)) is shown in Figures 3.30 and 3.31.

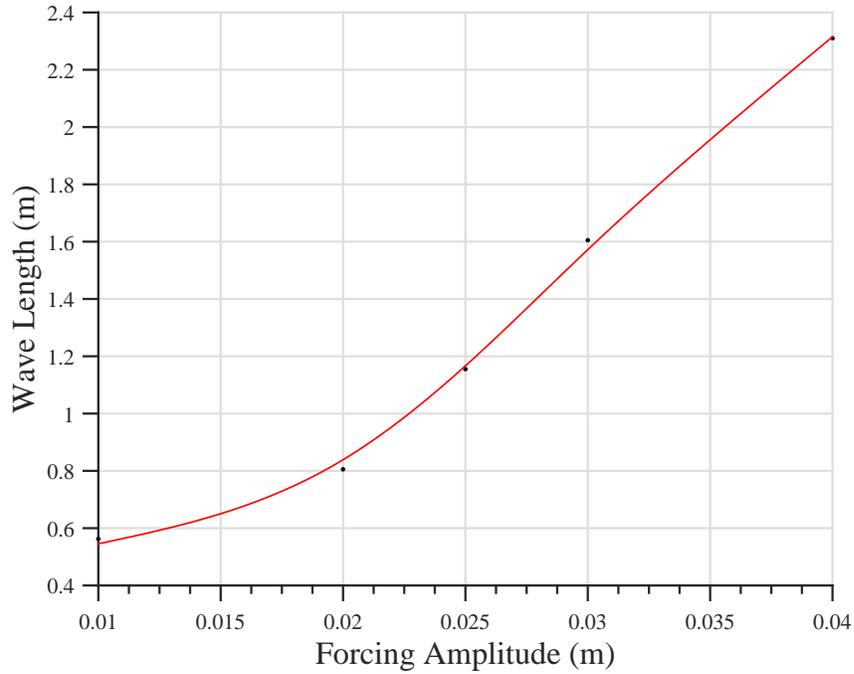
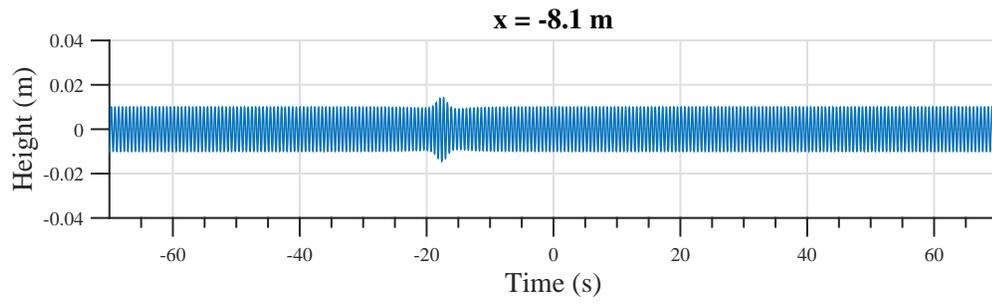


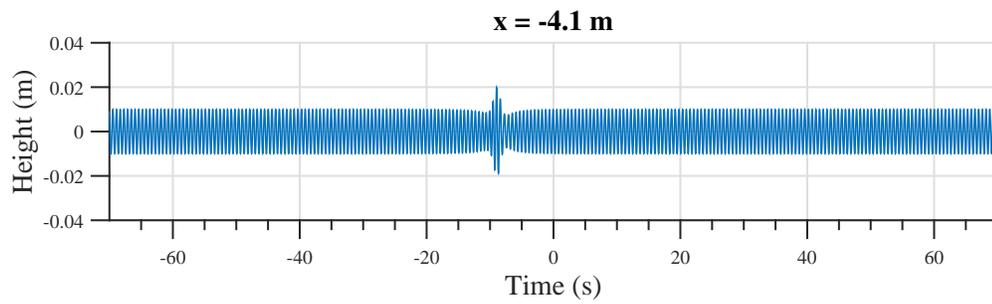
Figure 3.28: Parametric study showing generated wave length versus wave maker amplitude for different forcing frequencies and water depths. These results follow the dispersion relation. See Figure 3.11.

Although, the wave maker motion is described by equation (3.44), its amplitude of motion is reduced by a factor to generate a carrier wave of desired amplitude according to Figure 3.27.

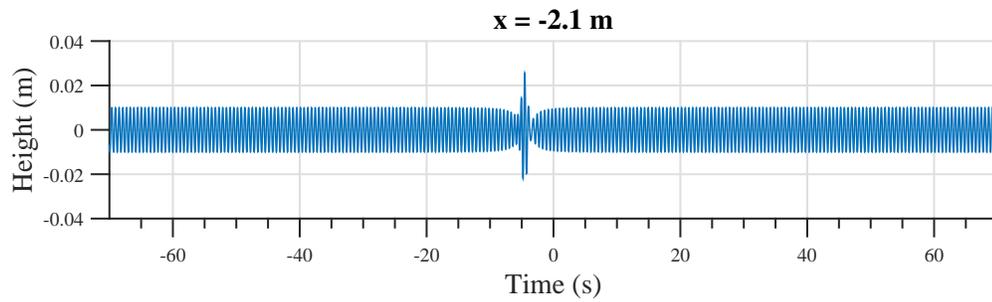
Using the approach described above, several numerical experiments have been carried out to realize modulational instability. Two case studies are presented in this section. Before discussing the case studies in detail, it is critical to understand that the simulation results are not expected to quantitatively match the theoretical results. This is primarily because of two reasons: Firstly, the Nonlinear Schrödinger equation does not provide a complete description of full-field water waves. Although, unstable solutions to the NSE (such as the Peregrine breather) can be used to induce modulational instability in certain cases, it is extremely difficult to tune the



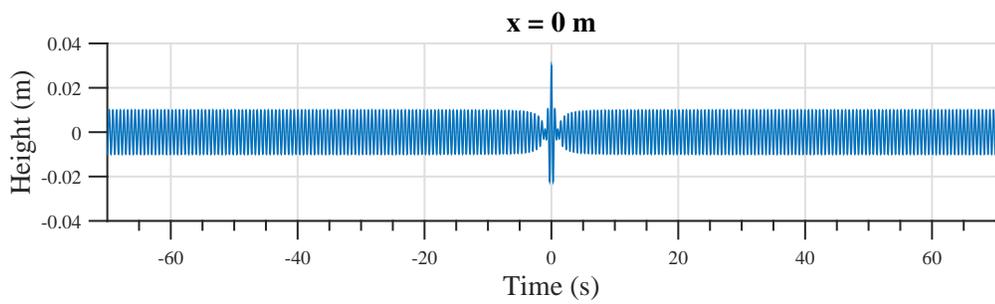
(a)  $x = -8.1$  m



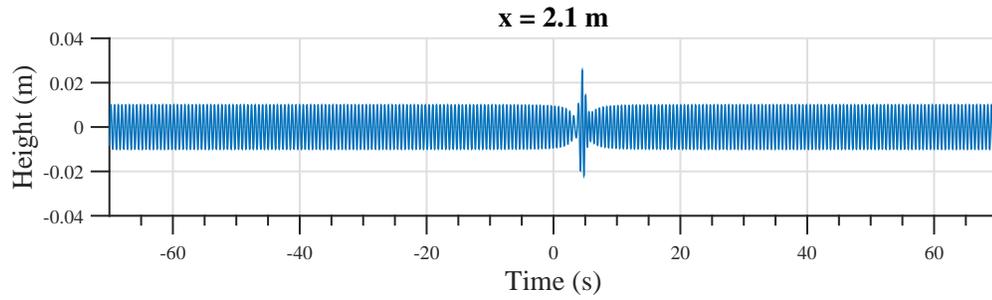
(b)  $x = -4.1$  m



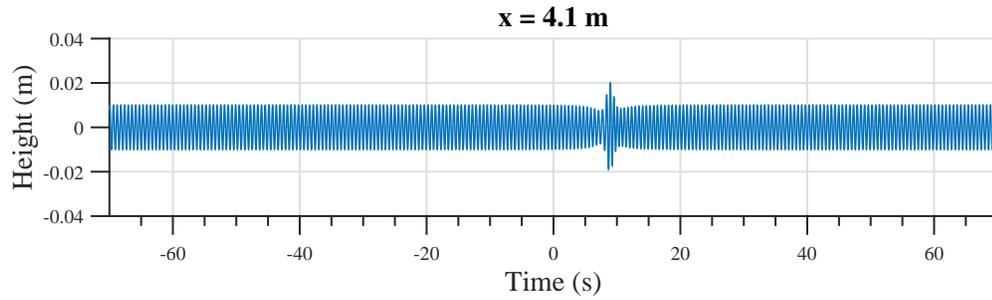
(c)  $x = -2.1$  m



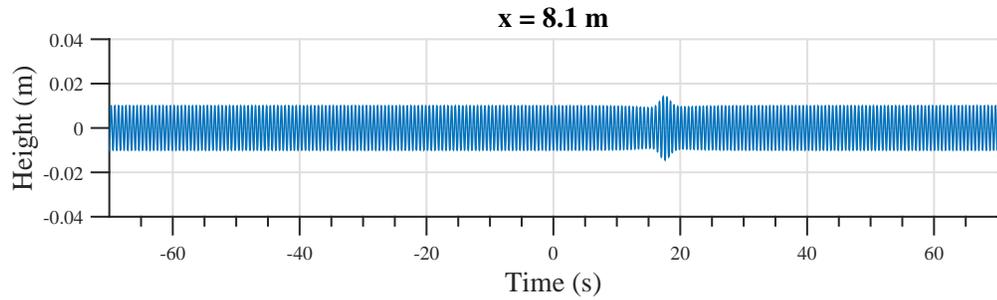
(d)  $x = 0.0$  m



(e)  $x = 2.1$  m

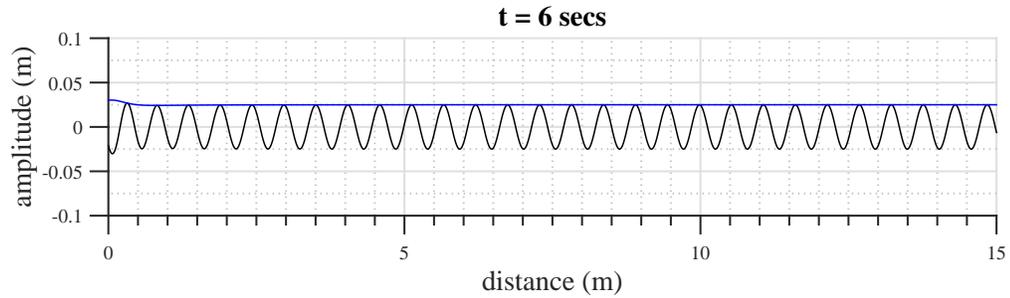


(f)  $x = 4.1$  m

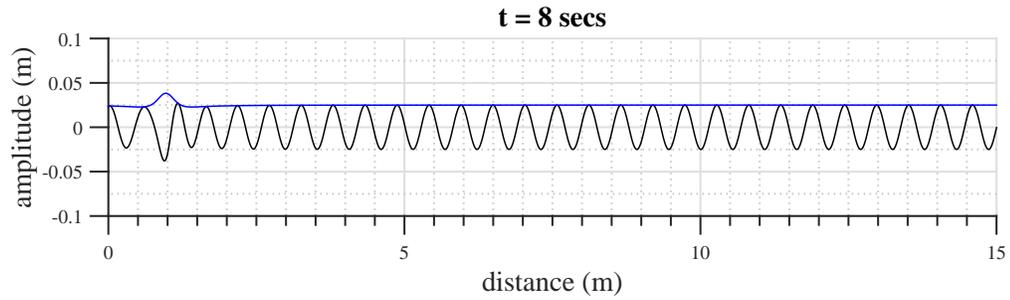


(g)  $x = 8.1$  m

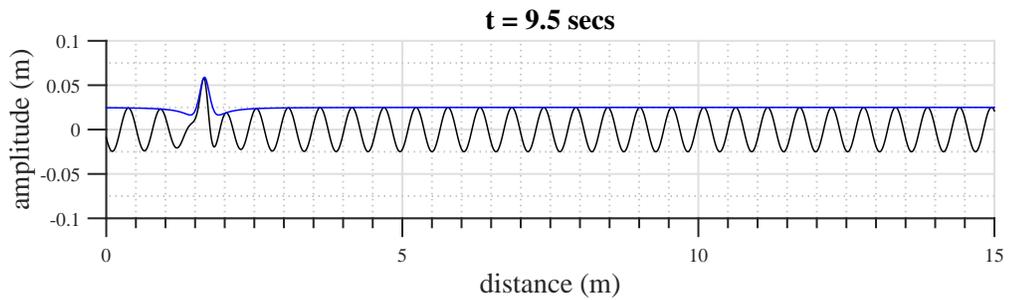
Figure 3.29: Evolution of the dimensional form of Peregrine breather as described by equation (3.44) ( $a_0 = 0.01$  m,  $L_0 = 0.54$  m).



(a)  $t = 6.0$  secs

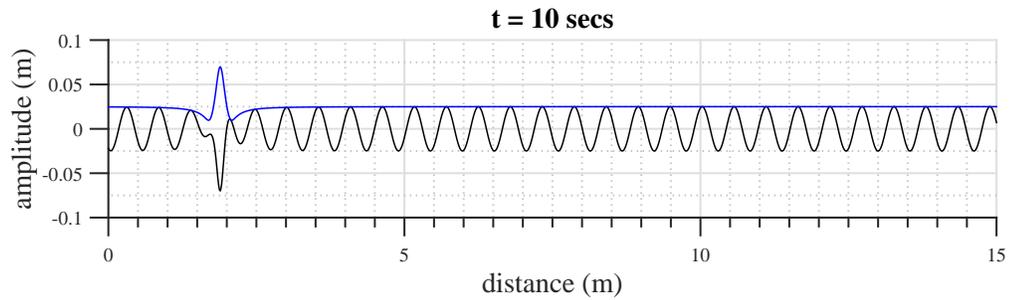


(b)  $t = 8.0$  secs

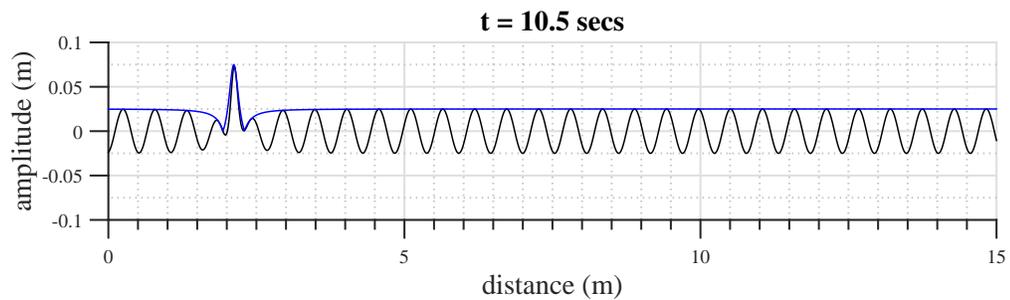


(c)  $t = 9.5$  secs

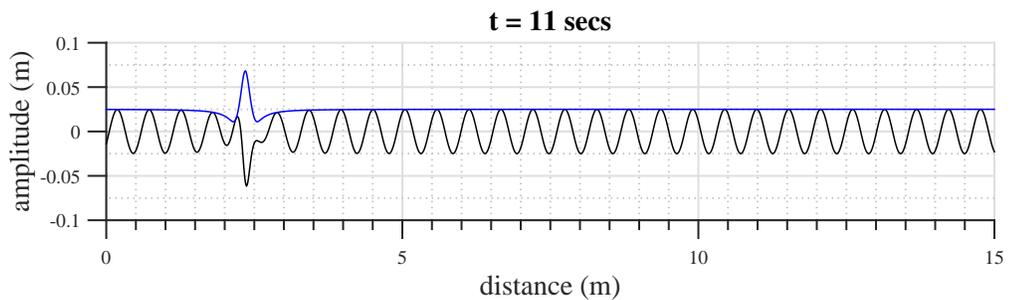
Figure 3.30: Predicted surface profile according to the analytic formulation of the Peregrine breather as described by equation (3.44) ( $t = 6.0$  secs to  $t = 9.5$  secs). The perturbation is introduced after 10 lead cycles. ( $a_0 = 0.025$  m,  $L_0 = 0.54$  m).



(a)  $t = 10.0$  secs



(b)  $t = 10.5$  secs



(c)  $t = 11.0$  secs

Figure 3.31: Predicted surface profile according to the analytic formulation of the Peregrine breather as described by equation (3.44) ( $t = 10.0$  secs to  $t = 11.0$  secs). The perturbation is introduced after 10 lead cycles. ( $a_0 = 0.025$  m,  $L_0 = 0.54$  m).

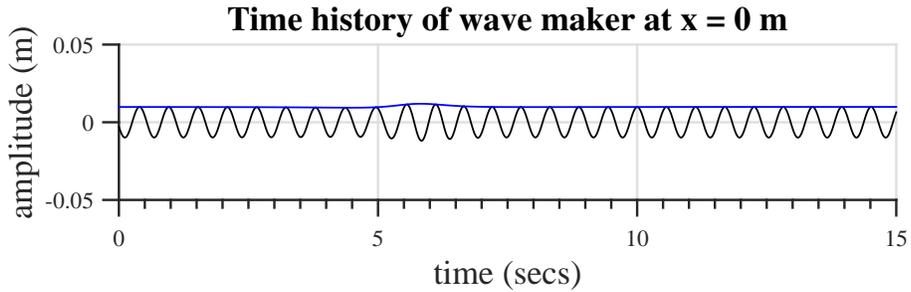


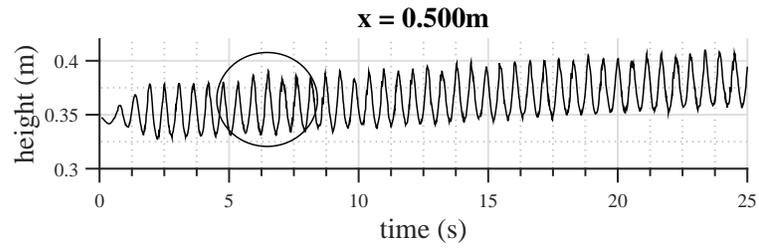
Figure 3.32: Time history of the wave maker to theoretically induce the surface evolution shown in Figures 3.30 and 3.31. The wave maker amplitude is 0.01 m produces a carrier wave of amplitude 0.025 m.

simulation parameters to generate results that perfectly match the predictions made by the mathematical model. Secondly, due to the inherent dissipation prevalent in the SPH model, it is rather difficult to detect growth. Instead modulational instability has been realized by observing the fact that the unstable perturbation resists decay unlike the rest of the carrier wave.

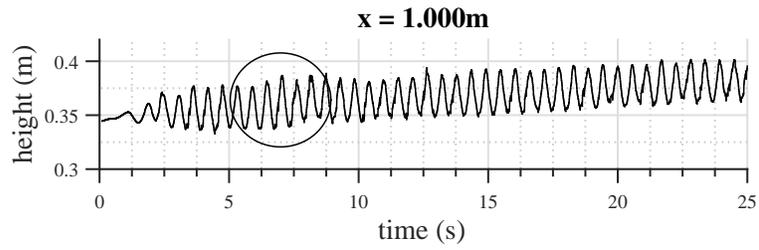
### 3.3.7.1 Case Study 1: $a_0 = 2.5$ cm; $L_0 = 54$ cm

For the first case study, a numerical wave tank of length 4.5 m is simulated using 90000 particles having a mean water depth of 0.35 m. The execution time for this simulation is approximately 32 ms per time step for a smoothing length ( $h$ ) of  $10^{-2}$  m. The left wall is excited according to a phase shifted version of equation (3.44) such that the theoretical self-focusing location of the initial perturbation is 2.1 m. The carrier amplitude is  $a_0 = 0.025$  m and the carrier wavelength (according to Figure 3.28) is  $L_0 = 0.54$  m. Time histories are recorded at various locations

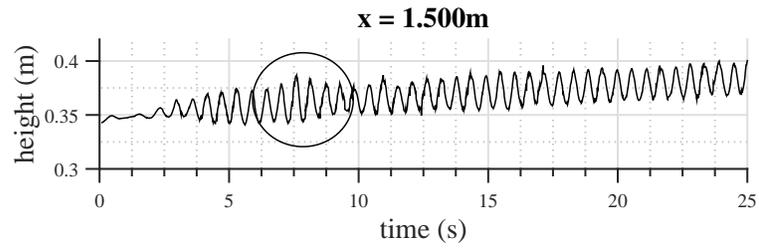
along the length of the numerical wave tank. The propagation of the perturbation is highlighted in each subfigure. A very small perturbation is introduced into the system as shown in Figure 3.33(a). At this stage, the perturbation amplitude is comparable to that of the carrier wave. Initially the perturbation and the rest of the carrier wave decays at the same rate maintaining a comparable wave height upto  $x = 1.0$  m as shown in Figure 3.33(b). Subsequently, due to modulational instability, the perturbation strives to grow unlike the rest of the carrier wave. This growth, however, is stymied by the algorithmic dissipation in SPH. This counteraction of self-focusing and dissipation spawns a differential rate of decay and a distinct modulated wave form begins to appear at  $x = 1.5$  m (Figure 3.33(c)). Thus, the attenuation tendencies due to the inherent dissipation in the SPH algorithm is resisted by the modulation instability of the perturbation. As the wave propagates (as shown in Figures 3.33(d) - 3.35(d)), the modulated waveform loses energy but maintains its rough shape before almost completely disappearing into the background at around  $x = 3.5$  m. At this point, the amplitude of the perturbation is again comparable to that of the background carrier wave.



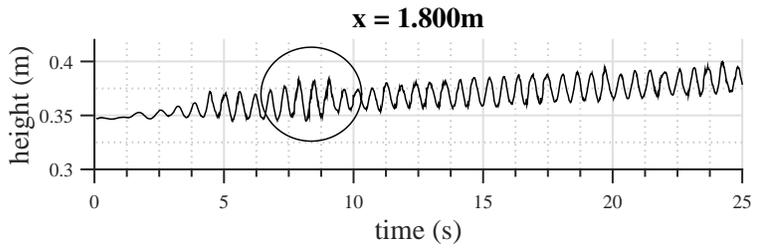
(a)  $x = 0.5$  m



(b)  $x = 1.0$  m

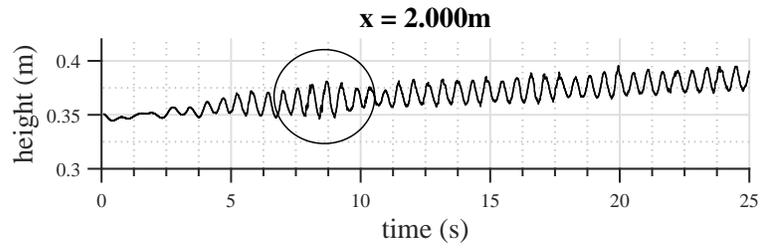


(c)  $x = 1.5$  m

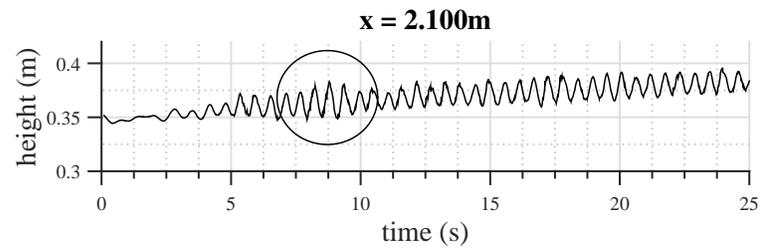


(d)  $x = 1.8$  m

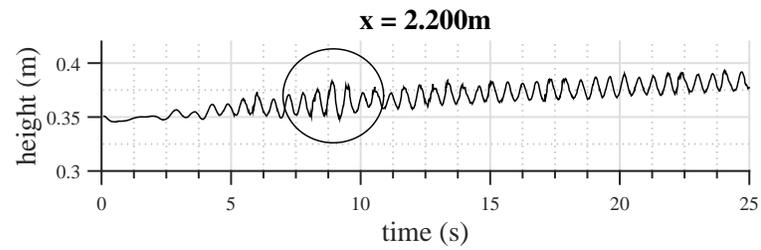
Figure 3.33: 1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$  m,  $L_0 = 0.54$  m): The perturbation is introduced after 10 lead cycles ( $x = 0.5$  m to  $x = 1.8$  m).



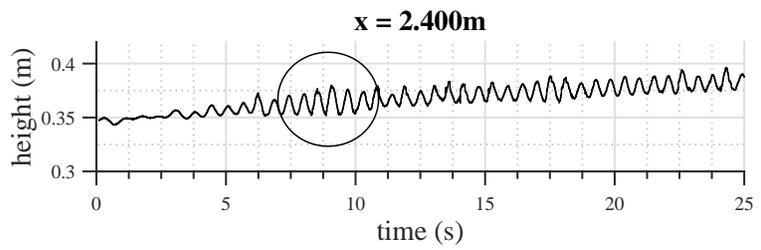
(a)  $x = 2.0$  m



(b)  $x = 2.1$  m

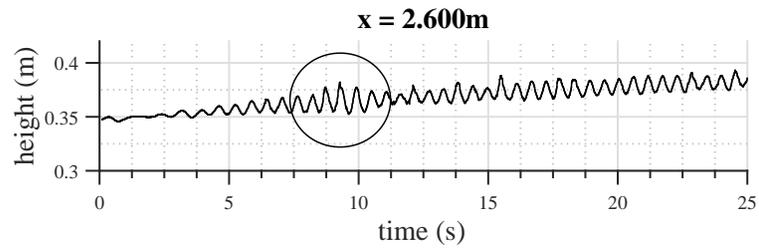


(c)  $x = 2.2$  m

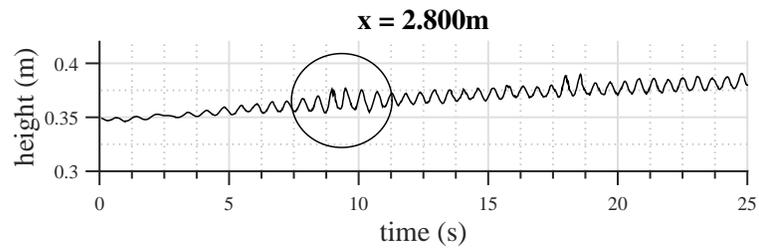


(d)  $x = 2.4$  m

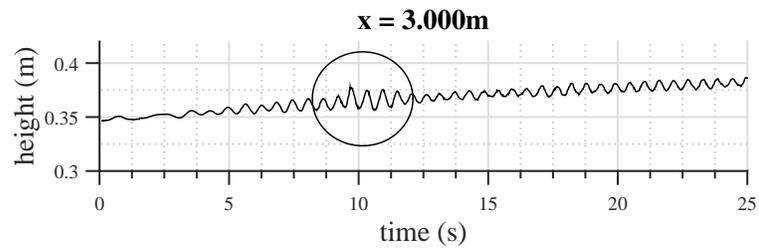
Figure 3.34: 1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$  m,  $L_0 = 0.54$  m): ( $x = 2.0$  m to  $x = 2.4$  m).



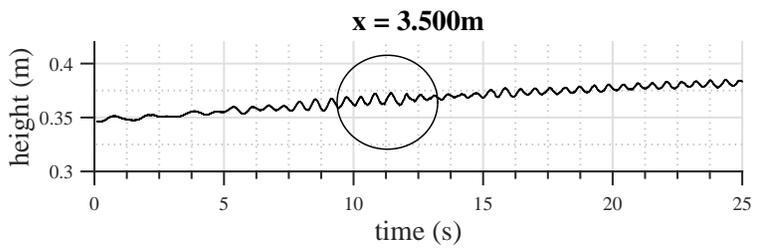
(a)  $x = 2.6$  m



(b)  $x = 2.8$  m



(c)  $x = 3.0$  m



(d)  $x = 3.5$  m

Figure 3.35: 1+1 Dimension Modulational Instability Case Study 1 ( $a_0 = 0.025$  m,  $L_0 = 0.54$  m): ( $x = 2.6$  m to  $x = 3.5$  m).

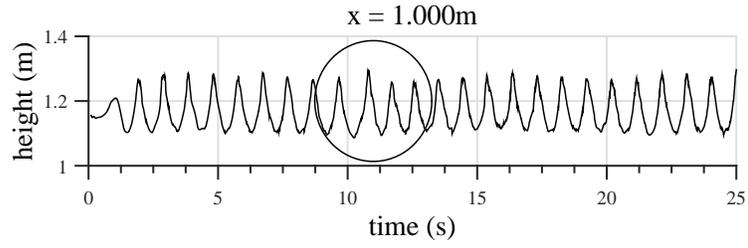
### 3.3.7.2 Case Study 2: $a_0 = 9.0$ cm; $L_0 = 162$ cm

A numerical wave tank of length 15 m is simulated using 240000 particles having a water depth of 1.2 m. The execution time for this simulation is approximately 273 ms per time step for a smoothing length ( $h$ ) of  $3.7 \times 10^{-2}$  m. The left wall is excited such that the theoretical self-focusing location is 2.1 m. The carrier amplitude is  $a_0 = 0.09$  m and the carrier wavelength (according to Figure 3.28) is  $L_0 = 1.62$  m. The wave maker stroke length is 0.03 m based on the parametric study shown in Figure 3.27. An evolution similar to the previous case can be seen here. The perturbation amplitude is comparable to the rest of the carrier wave at  $x = 1.0$  m as shown in Figure 3.36(a). As explained earlier, due to modulational instability, the perturbation retains its amplitude till  $x = 4.0$  m while the rest of the carrier wave decays due numerical dissipation (Figure 3.36(d)). Subsequently, the perturbation decays and its amplitude becomes similar to the background carrier amplitude at around  $x = 8.0$  m.

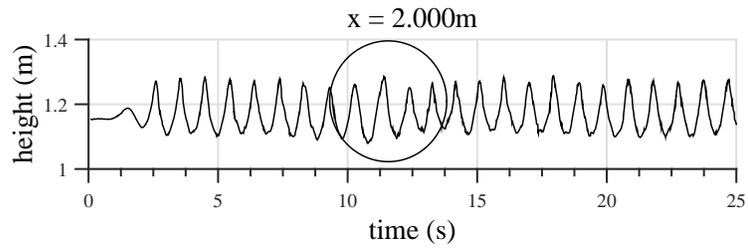
A careful observation of Figures 3.33 - 3.35 in case study 1 would suggest that the relative height of the perturbation with respect to the background carrier wave is maximum at around  $x = 2.1$  m or  $x = 2.2$  m. This is also the target localization point according to the mathematical model. However, for case study 2 (Figures 3.36 - 3.38), relative height of the perturbation seems to be maximum at around  $x = 4.0$  m. This would suggest that for a relatively shorter wave tank, a modulated wavetrain with smaller carrier wave amplitude and large wave number is a better candidate to examine modulational instability as described by NSE. However, this

conclusion is subject to further investigations.

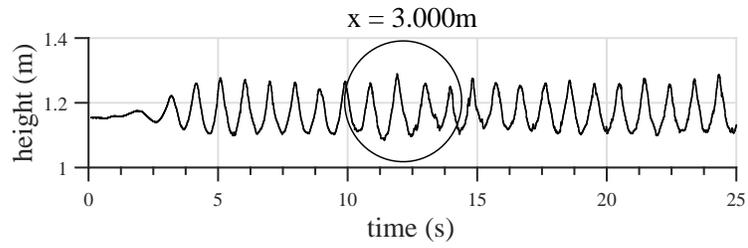
As mentioned before, it should be noted that due to significant dissipation in the current model, it is not possible to quantitatively establish a singular focusing point or a maximum height as predicted by the mathematical model. However, the observations made in the numerical experiments suggest that it is indeed possible to identify modulational instability in water waves using SPH, albeit only to a certain extent.



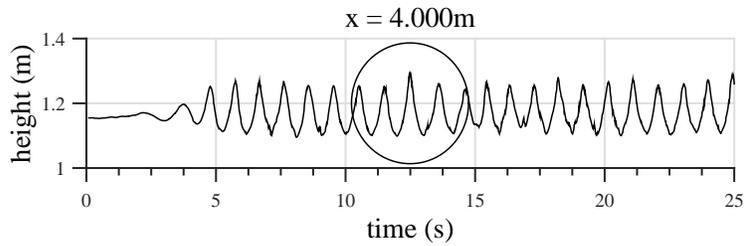
(a)  $x = 1.0$  m



(b)  $x = 2.0$  m

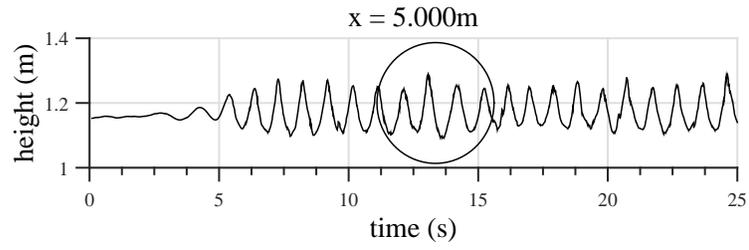


(c)  $x = 3.0$  m

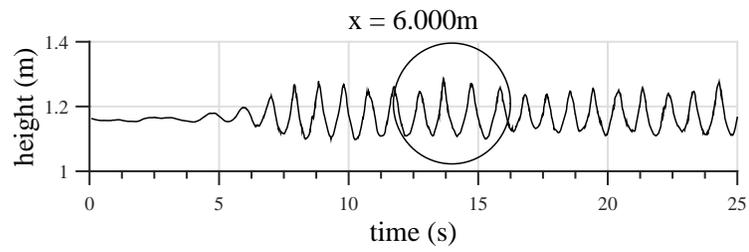


(d)  $x = 4.0$  m

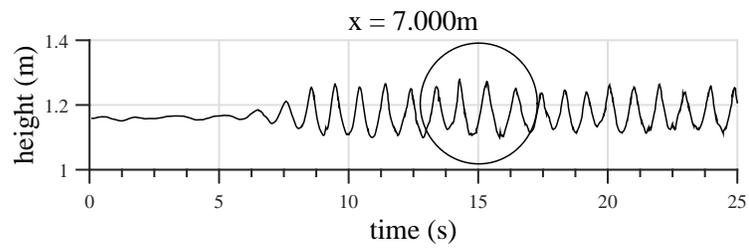
Figure 3.36: 1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$  m,  $L_0 = 1.62$  m): The perturbation is introduced after 10 lead cycles ( $x = 1.0$  m to  $x = 4.0$  m).



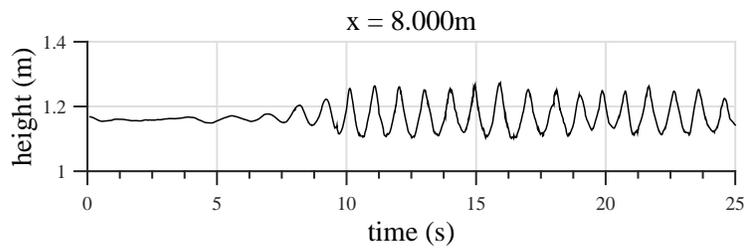
(a)  $x = 5.0$  m



(b)  $x = 6.0$  m

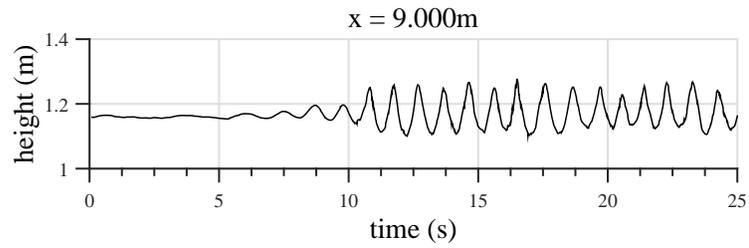


(c)  $x = 7$  m

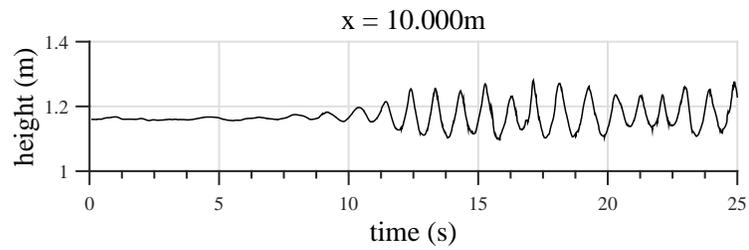


(d)  $x = 8$  m

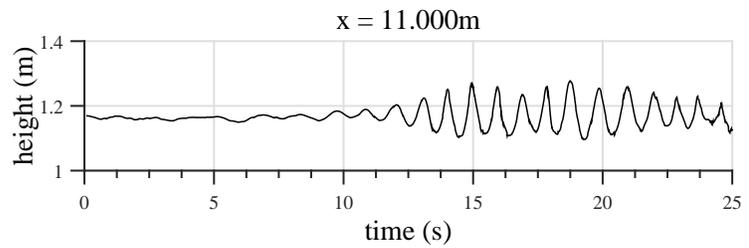
Figure 3.37: 1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$  m,  $L_0 = 1.62$  m): ( $x = 5.0$  m to  $x = 8.0$  m).



(a)  $x = 9.0$  m



(b)  $x = 10.0$  m



(c)  $x = 11.0$  m

Figure 3.38: 1+1 Dimension Modulational Instability Case Study 2 ( $a_0 = 0.09$  m,  $L_0 = 1.62$  m): ( $x = 9.0$  m to  $x = 11.0$  m).

## Chapter 4: Summary and Concluding Remarks

In this dissertation work, mechanisms leading to rogue-wave or extreme wave formations have been investigated through analytic studies of mathematical models as well as Lagrangian based massively parallel N-particle simulations. It should be noted that this work is phenomena based, focused mainly on the qualitative study of the mechanisms. Further advancement of this study can lead to more quantitatively accurate results. Analytical (rogue-wave solutions to the nonlinear Schrödinger equation) and simulational (SPH simulations of mechanisms of extreme wave formation) results presented in this dissertation can act as a precursor to future efforts aimed at enhancing predictive capabilities in the field of extreme wave climatology. Contributions made in this dissertation can also aid the study of structural response to extreme waves and energy harnessing from wave localizations.

### 4.1 Summary of Contributions

The key aspects and contributions of this dissertation are the following:

1. A Predictor-Corrector type algorithm, not available elsewhere in the literature, is presented in this dissertation which allows for the identification of spectral parameters of single mode near homoclinic theta function based so-

lutions to the NSE. While this solution family is already known, examples of variations in the parameters governing the solutions are not readily available in the literature. Furthermore, connections between the features of the numerically generated eigenvalue space and the solutions appearing therein have been seldom highlighted. The eigenvalue map presented in this work provides a quantitatively accurate overview and context of the behavior of the solution space from which these solutions originate. The particular solutions presented here, and the insights provided by the mapping procedure, can substantially enhance the understanding and stimulate further investigation of the NSE and associated rogue waves. Although the computations required to determine the map are quite demanding, it is shown that these computations can be efficiently accelerated with a parallel computing architecture. This is the first time that the power of GPU computing has been used in a combined analytical-numerical study of extreme waves.

2. New wave fields for near homoclinic, single mode rogue-wave solutions of the periodic nonlinear Schrödinger equation are presented. Features of the predictive map are explored and the influences of certain parameter variations are presented.
3. The solutions are rescaled to match the length scales of waves generated in a wave tank. Based on the information provided by the map and the details of physical scaling, it is believed that the framework presented here could serve as a basis for experimental investigations into a variety of rogue waves as well

localizations in wave fields.

4. Dissipation inherent to SPH models makes it extremely difficult to simulate persistent progressive waves. This issue has never been explicitly discussed in the literature. In this dissertation, an observation-based solution to this problem has been provided thus making it feasible to study localization in water waves.
5. One of the primary objectives of this dissertation work was to investigate the mechanisms underlying rogue-wave formation through simulations. Directional focusing, dispersive enhancement and modulational instability have been studied in 1+1 dimensions through advanced SPH simulations to realize the underpinnings of extreme wave localization in a hydrodynamic environment. This is the first time SPH has been used as a simulation tool in such a study. Although an extensive quantitative study has not been carried out, the results obtained in this dissertation work can serve as a precursor to the development of a more advanced simulation based predictive tool for extreme waves.

## 4.2 Recommendations for Future Work

There is a great deal of possible future directions for this research. The unique predictor-corrector type algorithm and GPU-computing based technique developed in this dissertation work to find rogue-wave solutions to NSE can be extended to other mathematical models with similar characteristics such as the Korteweg-de Vries or KdV equation. Moreover, this study can be extended to other systems

such as optical fibers and plasmas. Analytic investigations in rogue-wave formation in 1+1 dimensions can be continued using more advanced mathematical models such as the Dysthe equation. The extended Dysthe equation, which is a weakly nonlinear model like the NSE, provides a better approximation of modulational instability. Analytic studies can also be extended to 2+1 dimensions using Coupled nonlinear Schrödinger Equation. Coupled NSE the interaction of slowly modulated wave trains in two dimensions, thus providing a more realistic picture of rogue-wave formation in the open ocean. Initial conditions described in Section 2.2.7 can be used in experiments to realize modulational instability and energy localization governed by the NSE.

Based on the work done in this dissertation, significant advances can be made in area of SPH based rogue-wave simulations. The massively parallel GPU based SPH code, developed to study extreme waves, has scope of optimization and tuning to better capture extreme wave phenomenon. Further use of shared memory, asynchronous streams and multi-GPU models can significantly improve the performance of this program. Although studies carried out during the course of this dissertation work have been able to address the SPH dissipation issue to some extent, it is essential to carry out further investigations in this area to improve the accuracy of results. The SPH code can be used to simulate various other unstable rogue-wave modes of the NSE as well as realize the insights developed from the study of other mathematical models.

The area of extreme wave simulations is still wide open in 3D. A 3D version of the SPH code has already been developed which can be used to study directional

focusing and modulational instability induced extreme wave formation in 2+1 dimensions. The SPH code can be further modified to incorporate fluid-structure interaction problems to study the impact of extreme waves on offshore structures.

## Chapter A: Mathematical Details of Floquet Theory

The general form for a first-order homogeneous linear system is

$$x' = A(t)x \tag{A.1}$$

where  $A(t)$  is an  $n \times n$  matrix function of  $t$ , continuous for  $t \in E, a \leq t \leq b$ . The homogeneous system in equation (A.1) has two important properties. The first is that the identically zero function  $x(t) = 0$  for all  $t \in E$ , is a solution of equation (A.1), and is the unique solution such that  $x(t_0) = 0$  for any  $t_0 \in E$ . The second is that, if  $x^1(t), \dots, x^m(t)$  are solutions of equation (A.1), then so is

$$c_1x^1(t) + \dots + c_mx^m(t)$$

Let  $x^1(t), \dots, x^n(t)$  be  $n$  solutions of equation (A.1) on an interval  $I$ , and put

$$X(t) = [x^1(t), \dots, x^n(t)], \tag{A.2}$$

where  $X(t)$  is an  $n \times n$  matrix solution of

$$X' = AX \tag{A.3}$$

If  $x^1, \dots, x^n$  are also linearly independent, then  $X$  is a **fundamental matrix** and, if  $X(t_0) = I$ , the identity matrix, then  $X(t)$  is the **principal** fundamental matrix.

Further,

$$W(t) = \det X(t) \tag{A.4}$$

is called the **Wronskian**. If  $X(t)$  is a fundamental matrix solution of equation (A.3), then so is  $X(t)C$  for any non-singular constant matrix  $C$ . Let,

$$Y(t) = X(t)C \tag{A.5}$$

Then  $Y(t)$  is non-singular, and

$$Y' = X'C = AXC = AY$$

The columns of  $Y$  are linear combinations of the columns of  $X$ . Also, the general solution  $c_1x^1(t) + \dots + c_mx^m(t)$  can be written in the form

$$x(t) = X(t)c, \tag{A.6}$$

where  $c$  is an arbitrary  $n$ -vector, with components  $c_1, \dots, c_n$ .

We observe that, as  $t \rightarrow t_0$ ,

$$X(t) = X(t_0) + (t - t_0)X'(t_0) + o(t - t_0)$$

or

$$X(t) = X(t_0) + (t - t_0)A(t_0)X(t_0) + o(t - t_0)$$

where equation (A.3) is used to calculate  $X'(t_0)$ . But, using equation (A.4) for  $W(t)$ , it follows that

$$W(t) = W(t_0)\det[I + (t - t_0)A(t_0)] + o(t - t_0)$$

Now since,

$$\det(I + \epsilon C) = 1 + \epsilon \text{tr}(C) + O(\epsilon^2)$$

We have

$$W(t) = W(t_0)[1 + (t - t_0)\text{tr}A(t_0)] + o(t - t_0)$$

Now, it can also be written

$$W(t) = W(t_0) + (t - t_0)W'(t_0) + o(t - t_0)$$

Hence, on taking the limit  $t \rightarrow t_0$ , we see that

$$W'(t_0) = W(t_0)\text{tr}A(t_0)$$

But  $t_0$  can be any point in  $E$  and hence, for all  $t$  in  $E$ ,

$$W'(t) = W(t)\text{tr}A(t)$$

Integration with respect to  $t$  now gives

$$W(t) = W(t_0)\exp\left[\int_{t_0}^t \text{tr}A(s)ds\right] \quad (\text{A.7})$$

Now, let us consider a general form for a linear homogeneous system with periodic

$$x' = A(t)x \quad (\text{A.8})$$

where

$$A(t + T) = A(t) \quad (\text{for all } t) \quad (\text{A.9})$$

Thus the coefficient matrix is periodic with a period  $T$ . It may be noted that although the coefficient matrix in equation (A.8) is periodic, in general the solutions are not periodic.

Now, since  $X(t)$  is a fundamental matrix, it follows from equation (A.3)

$$X'(t) = A(t)X(t)$$

Let  $Y(t) = X(t + T)$ . Then

$$\begin{aligned} Y'(t) &= X'(t + T) \\ &= A(t + T)X(t + T) \\ &= A(t)Y(t) \end{aligned}$$

where the last line is a consequence of equation (A.9) and definition of  $Y(t)$ . Thus  $Y(t)$  is also a fundamental matrix and, hence, has the form  $X(t)M$  for some constant non-singular matrix  $M$  (from equation (A.5)). Thus, there exists a non-singular constant matrix  $M$  for all  $t$  such that

$$X(t + T) = X(t)M \tag{A.10}$$

Further, using equation (A.7) it can also be shown that

$$\det M = \exp \left[ \int_0^T \text{tr} A(s) ds \right] \tag{A.11}$$

Since, equation (A.10) is true for all  $t$ , the constant matrix  $M$  can be expressed in terms of the fundamental matrix by putting  $t = 0$ :

$$M = X^{-1}(0)X(T) \tag{A.12}$$

It is often useful to choose  $X(t)$  to be the principal fundamental matrix, so that  $X(0) = I$ , and then  $M = X(T)$ . Here,  $M$  is called the **monodromy matrix**.

Let the eigenvalues of  $M$  be  $\rho_1, \dots, \rho_n$  called the **Floquet multipliers** for equation (A.8). The **Floquet exponents**  $\mu_1, \dots, \mu_n$  are defined by

$$\rho_1 = e^{\mu_1 T}, \dots, \rho_n = e^{\mu_n T} \quad (\text{A.13})$$

The Floquet exponents are not unique as we can replace  $\mu_i$  by  $\mu_i + 2\pi ik/T$  ( $i = 1, \dots, n$ ) for any integer  $k = \pm 1, \pm 2, \dots$  without altering the definition in equation (A.13). It may be noted that the Floquet multipliers and, hence, the characteristic exponents, do not depend on the particular choice of fundamental matrix  $X(t)$  and are intrinsic properties of the equation (A.8).

If  $\rho$  and  $\mu$  are as defined by equation (A.13), then there exists a solution  $x(t)$  of equation (A.8) such that for all  $t$

$$x(t + T) = \rho x(t) \quad (\text{A.14})$$

Further, there exists a periodic function  $p(t)$  (i.e.  $p(t + T) = p(t)$  for all  $t$ ) such that

$$x(t) = p(t)e^{\mu t}, \quad (\text{A.15})$$

The above can be proved as follows:

Let  $b$  be an eigenvector of  $M$  corresponding to the eigenvalue  $\rho$ , so that

$$Mb = \rho b$$

Then put

$$x(t) = X(t)b$$

and so  $x(t)$  is a solution of equation (A.8). But now

$$x(t + T) = X(t + T)b = X(t)Mb = X(t)\rho b = \rho x(t)$$

where the first step uses equation (A.10). Next put

$$p(t) = x(t)e^{-\mu t},$$

so that

$$p(t + T) = x(t + T)e^{-\mu t}e^{-\mu T} = \rho x(t)e^{-\mu t}e^{-\mu T} = p(t)$$

where equation (A.13) and equation (A.14) have been used.

It follows from equation (A.14) and equation (A.15) that there exist  $n$  linearly independent solutions of equation (A.8) given by

$$x_i(t) = e^{\mu_i t} p_i(t) \tag{A.16}$$

where each  $p_i(t)$  is a periodic function with period  $T$ . Also, we have that

$$x_i(t + T) = \rho_i x_i(t)$$

$$x_i(t + NT) = \rho_i^N x_i(t)$$

Let

$$P_0(t) = \begin{bmatrix} \left[ p_1 \right] & \dots & \left[ p_n \right] \end{bmatrix} \tag{A.17}$$

Then  $P_0(t)$  is an  $n \times n$  matrix function of  $t$  is non-singular and is a periodic function of  $t$ , so that  $P_0(t+T) = P_0(t)$  for all  $t$ . Now, let  $X_0(t)$  be the fundamental matrix for equation (A.8) from the  $n$  linearly independent solutions shown in equation (A.16),

so that

$$\begin{aligned} X_0(t) &= \left[ \begin{array}{c} \left[ x_1 \right] \dots \left[ x_n \right] \end{array} \right] \\ &= P_0(t)Y_0(t), \end{aligned} \tag{A.18}$$

where

$$Y_0(t) = \begin{bmatrix} e^{\mu_1 T} & & 0 \\ & \ddots & \\ 0 & & e^{\mu_n T} \end{bmatrix}$$

and  $Y_0(t)$  satisfies the equation

$$Y_0' = D_0 Y_0$$

where  $D_0 = \text{diag}[\mu_1, \dots, \mu_n]$ . This is a matrix differential equation with constant coefficients. We can now see that equation (A.8) will have periodic solutions of period  $T$  if and only if there exists a characteristic exponent  $\mu = 0$  or, correspondingly, characteristic multiplier  $\rho = 1$ .

Each characteristic(or Floquet) multipliers fall into one of the following categories:

1. If  $|\rho| < 1$ , then  $\text{Re}(\mu) < 0$  and so  $x(t) \rightarrow 0$  as  $t \rightarrow \infty$
2. If  $|\rho| = 1$ , then  $\text{Re}(\mu) = 0$  and so we have a psuedo-periodic solution. If  $\rho = \pm 1$ , then the solution is periodic with period  $T$ .
3. If  $|\rho| > 1$ , then  $\text{Re}(\mu) > 0$  and so  $x(t) \rightarrow \infty$  as  $t \rightarrow \infty$

The entire solution is stable if all the characteristic multipliers satisfy  $|\rho^i| \leq 1$

Chapter B: Additional Physical Forms of New Rogue Wave Solutions  
to the NSE

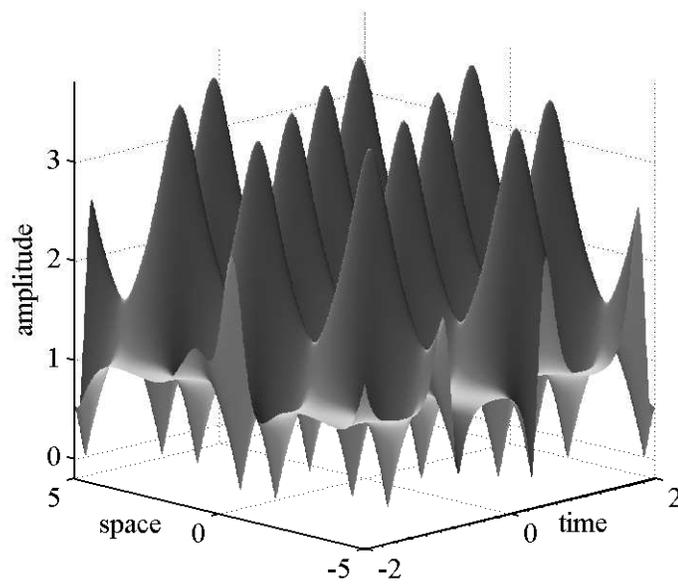


Figure B.1: Solution to the NSE for  $(\lambda_R = 0.87528, \lambda_I = 1.3008)$ ,  $\epsilon = 0.008014$ ,  $\theta = 0.629216906$ , and  $L = 5.44$ . This solution envelope reaches a maximum amplitude of  $\approx 3.6x$  the background height.

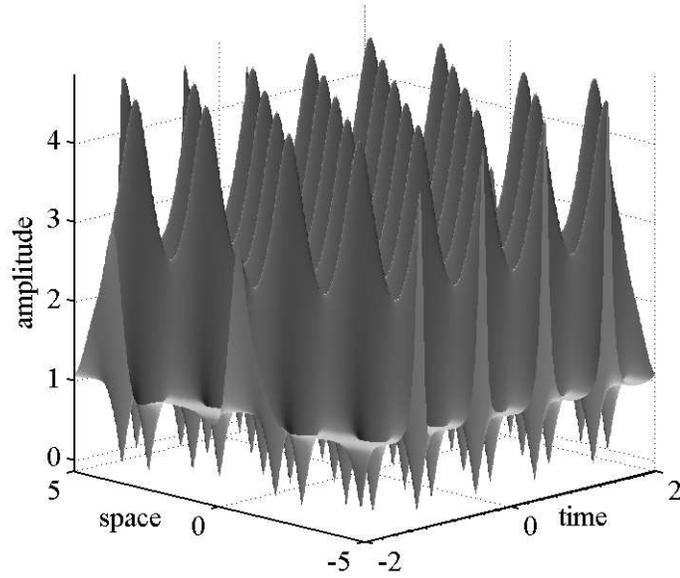


Figure B.2: Solution to the NSE for  $(\lambda_R = 1.22, \lambda_I = 1.852)$ ,  $\epsilon = 0.0026265$ ,  $\theta = 1.047105344$ , and  $L = 4.44$ . This solution envelope reaches a maximum amplitude of  $\approx 4.7x$  the background height.

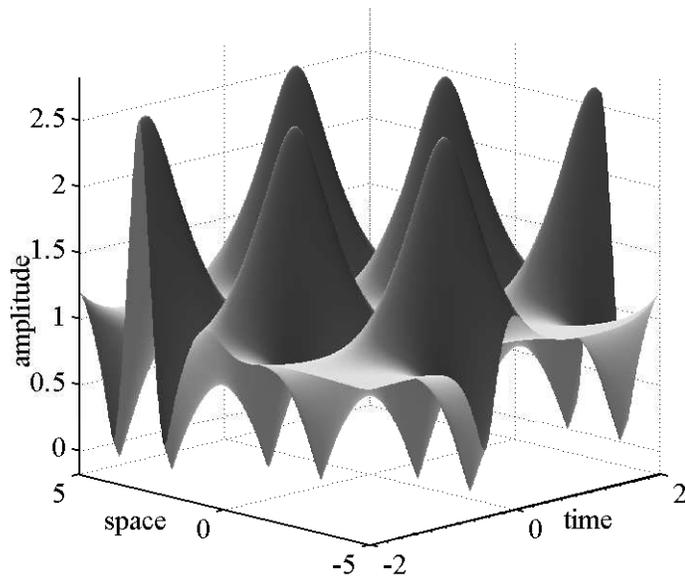


Figure B.3: Solution to the NSE for  $(\lambda_R = 0.7356, \lambda_I = 0.8237)$ ,  $\epsilon = 2.53E-03$ ,  $\theta = -1.051080567$ , and  $L = 20$ . This solution envelope reaches a maximum amplitude of  $\approx 2.65x$  the background height.

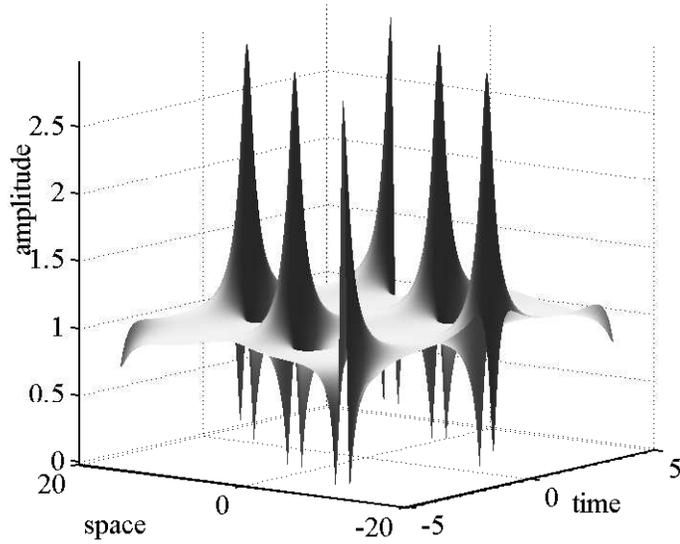


Figure B.4: Solution to the NSE for  $(\lambda_R = -0.0167, \lambda_I = 0.9805)$ ,  $\epsilon = 0.019545839$ ,  $\theta = 1.178877343$ , and  $L = 10$ . This solution envelope reaches a maximum amplitude of  $\approx 3x$  the background height.

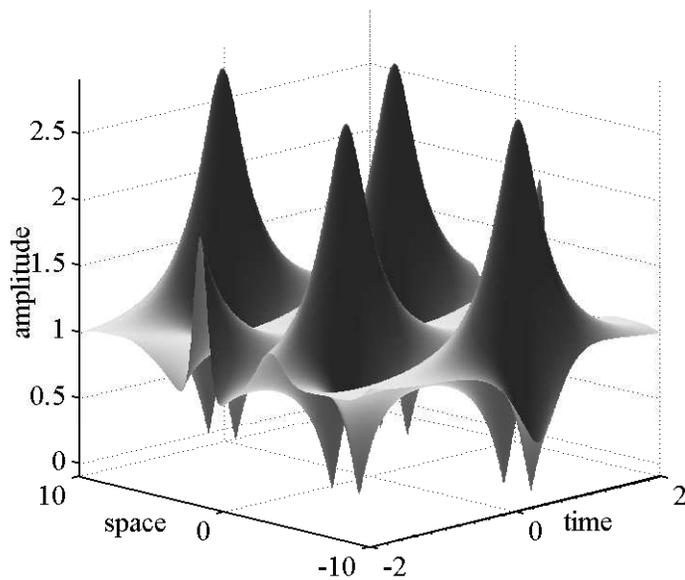


Figure B.5: Solution to the NSE for  $(\lambda_R = 0.14615, \lambda_I = 0.90249)$ ,  $\epsilon = 0.024303445$ ,  $\theta = -0.835951856$ , and  $L = 10$ . This solution envelope reaches a maximum amplitude of  $\approx 2.8x$  the background height.

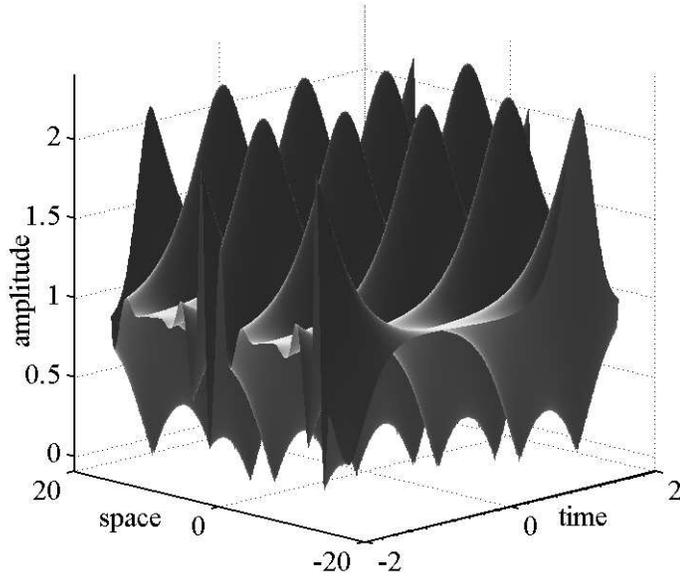


Figure B.6: Solution to the NSE for  $(\lambda_R = 0.54774, \lambda_I = 0.654)$ ,  $\epsilon = 0.007805263$ ,  $\theta = -0.676122251$ , and  $L = 15$ . This solution envelope reaches a maximum amplitude of  $\approx 2.3x$  the background height.

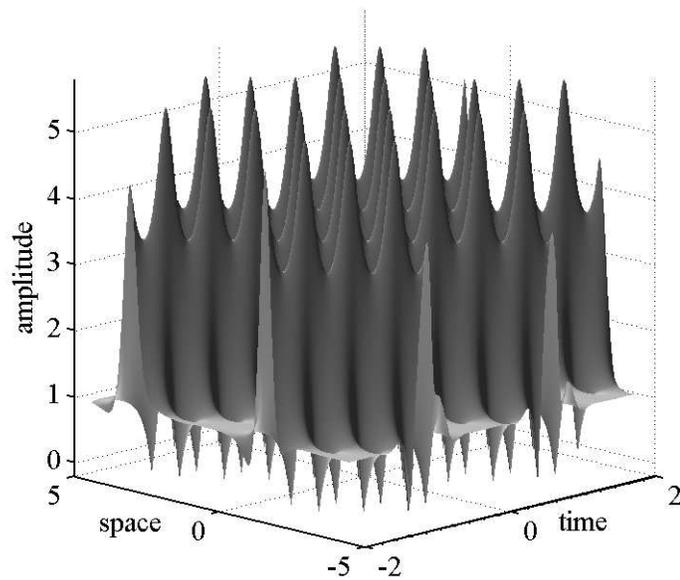


Figure B.7: Solution to the NSE for  $(\lambda_R = 0.60503, \lambda_I = 2.2946)$ ,  $\epsilon = 0.000495576$ ,  $\theta = 1.236466627$ , and  $L = 4.44$ . This solution envelope reaches a maximum amplitude of  $\approx 5.6x$  the background height.

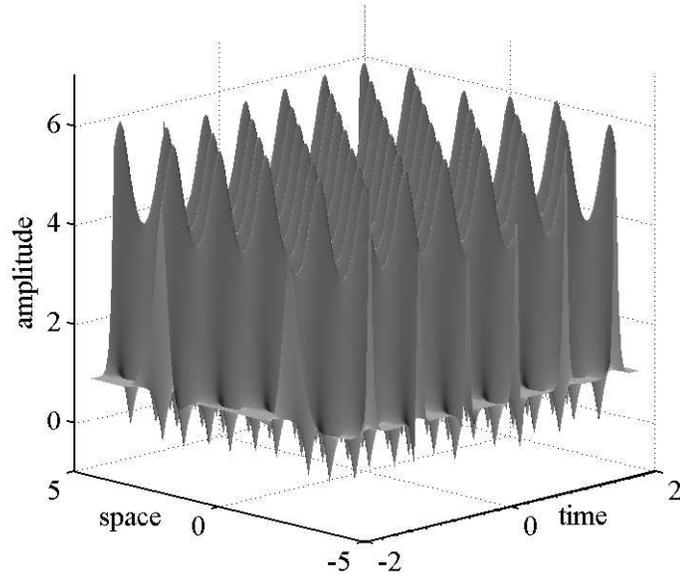


Figure B.8: Solution to the NSE for  $(\lambda_R = 1.5948, \lambda_I = 2.5409)$ ,  $\epsilon = 0.000160675$ ,  $\theta = -0.475103634$ , and  $L = 4.44$ . This solution envelope reaches a maximum amplitude of  $\approx 6x$  the background height.

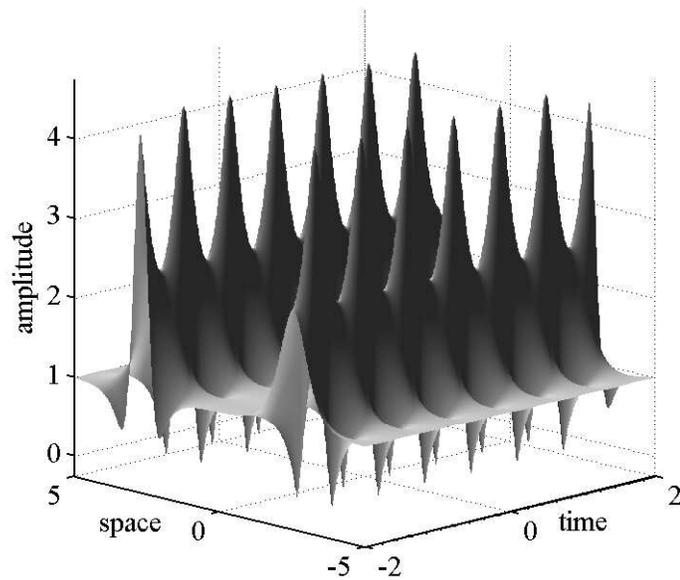


Figure B.9: Solution to the NSE for  $(\lambda_R = 0.0099, \lambda_I = 1.7498)$ ,  $\epsilon = 0.002485633$ ,  $\theta = 0.14954754$ , and  $L = 9$ . This solution envelope reaches a maximum amplitude of  $\approx 4.5x$  the background height.

## Chapter C: Sample Codes

### C.1 Smoothed Particle Hydrodynamics CUDA C++ Code Snippets

#### C.1.1 CUDA Kernel to Initialize Density

The densities of particles are initialized using the following:

$$\rho_i = \sum_j m_j W(\mathbf{r}_{ij}, h) \quad (\text{C.1})$$

where  $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$  is a vector directed from the position of particle  $j$  to the position of particle  $i$ ,  $h$  is a smoothing length (smoothing radius), and  $W(\mathbf{r}_{ij}, h)$  is the the smoothing function. The positions of the particles are read from the sorted array and their  $x, y$  addresses in the grid of bins are calculated. For every neighboring bin (including its own), a device function, `densityInCell2`, is called to compute density. For the computed hash value of the neighbor cell passed in, density is computed using an all pairs approach.

```
1 #include <cuda.h>
2 #include "math.h"
3 #include "cuda_runtime.h"
4 #include "device_launch_parameters.h"
5 #include "SPH2DCPPCuda.h"
6 #include <iostream>
7 #include "smoothingKernels.cuh"
8
9
10 __device__ double densityInCell2(int2 neighbor, int index, double posX, double posY ←
    , struct particleStructure* pparticles, struct paramsType* pparams);
11
12 __global__ void initializeDensity(struct particleStructure* pparticles, struct ←
    paramsType* pparams) {
13
```

```

14     int index = blockIdx.x*blockDim.x+threadIdx.x;
15
16     if (index >= (*pparams).nTotal) return;
17
18     //read particle data - host particle
19     double posXi = pparticles->sortedX[index]; //these are sorted, I is the ←
        receiver
20     double posYi = pparticles->sortedY[index];
21
22     //get address in grid
23     int tempX = floor((posXi-(*pparams).globalOriginX)*(*pparams).cellSizeRecip);
24     int tempY = floor((posYi-(*pparams).globalOriginY)*(*pparams).cellSizeRecip);
25
26     int2 gridPos = {tempX,tempY}; // grid position of host particle
27
28     //examine neighboring cells
29     double density = 0; //need density
30     for (int y = -1;y<=1;y++) {
31         int currentY = gridPos.y+y;
32         if ((currentY>-1) && (currentY<(*pparams).nCellsY)) {
33
34             for (int x = -1;x<=1;x++) {
35                 int currentX = gridPos.x+x;
36                 if ((currentX>-1) && (currentX<(*pparams).nCellsX)) {
37                     int2 neighbor = {currentX,currentY}; //2D index in grid
38                     density += densityInCell2(neighbor,index,posXi,posYi,←
                        pparticles,pparams);
39                 }
40             }
41         }
42     }
43
44
45     //write the density to the sorted position to be used later in the forces ←
        routine
46     pparticles->sortedRho[index] = density; //this one is not used
47
48     int originalIndex = pparticles->gridParticleIndex[index];
49     pparticles->density[originalIndex] = density; //this one is used
50
51     //also initialize dRhodt = 0 for all particles; order doesn't matter
52     pparticles->sorteddRhodt[index] = 0;
53
54     return;
55 }
56
57
58 // loop over the particles in the host cell and surrounding cells; compute ←
    density
59 __device__ double densityInCell2(int2 neighbor,int index,double posXi,double ←
    posYi,struct particleStructure* pparticles, struct paramsType* pparams) {
60
61     //compute 1D hash value
62     int hash = neighbor.y*(*pparams).nCellsX+neighbor.x;
63
64     int startIndex = pparticles->cellStart[hash];
65     double density = 0;
66     if (startIndex != 0xffffffff) {
67         int endIndex = pparticles->cellEnd[hash];
68
69         for (int ind1 = startIndex; ind1 < endIndex; ind1++) {
70
71             //remember to include self density
72             double posXj = pparticles->sortedX[ind1]; //get position of sending ←
                particles
73             double posYj = pparticles->sortedY[ind1];
74             double m2 = pparticles->mass[0]; //mass; right now these are ←
                identical for all particles

```

```

75 //compute density; We use Monaghan's formulation with Muller's ←
      skPoly6 smoothing kernel normalized to 2D
76 //The kernel is W =
77 double rSq = (posXi-posXj)*(posXi-posXj)+(posYi-posYj)*(posYi-←
      posYj);
78 double diffSq = (*pparams).h2-rSq;
79 if (diffSq>=0) {
80     double r = sqrt(rSq);
81     double r0h = r/pparams->h;
82     density += m2*poly6((*pparams).constDensity, r0h); // 4/(pi h^2)←
      *(1-r^2/h^2)^3
83
84     }; //end checking closeness
85 }; //end the for loop
86 }; //end the if statement
87
88 return density;}

```

## C.1.2 CUDA Kernel to Compute State Rates

Although the kernel is named `computedVdt`, it actually computes pressure,  $\frac{dv}{dt}$ ,  $\frac{dp}{dt}$ , and XSPH influence. These computations are carried out by using the formulation provided in Sections 3.2.1 to 3.2.5. The positions of the particles are read from the sorted array and their  $x, y$  addresses in the grid of bins are calculated. For every neighboring bin (including its own), a device function, `forcesInCell2`, is called to evaluate the different state rates into a states vector. `states[0] =  $f_x$` , `states[1] =  $f_y$` , `states[2] =  $XSPH_x$` , `states[3] =  $XSPH_y$` , and `states[4] =  $\frac{dp}{dt}$` . For the computed hash value of the neighbor cell passed in, the above mentioned state rates are computed using an all pairs approach. The computed state rates are assigned to corresponding locations in the unsorted arrays. These unsorted arrays are subsequently used to update the actual states.

```

1 #include <cuda.h>
2 #include "math.h"
3 #include "cuda_runtime.h"
4 #include "device_launch_parameters.h"
5 #include "SPH2DCPPCuda.h"
6 #include <iostream>
7 #include "smoothingKernels.cuh"

```

```

8
9  __device__ void forcesInCell2(int2 neighbor, int index, double posX, double posY, ←
    double velX, double velY, double rho, double pressurei, struct ←
    particleStructure* pparticles, struct paramsType* pparams, double* stateRates) ←
    ;
10
11
12  __global__ void computedVdt(struct particleStructure* pparticles, struct ←
    paramsType* pparams) {
13
14      int index = blockIdx.x*blockDim.x+threadIdx.x;
15
16      if (index >= (*pparams).nTotal) return;
17
18      double rhoRef      = pparams->rRef;
19      double tenVMaxSq   = pparams->tenVMaxSq;
20
21      //read primary particle data - this is sorted data
22      double posXi      = pparticles->sortedX[index];
23      double posYi      = pparticles->sortedY[index];
24      double velXi      = pparticles->sortedVx[index];
25      double velYi      = pparticles->sortedVy[index];
26      double rhoi       = pparticles->sortedRho[index];
27      double pressurei = computePressure(rhoi, rhoRef, tenVMaxSq);
28
29      //get address in grid
30      int tempX = floor((posXi-(*pparams).globalOriginX)*(*pparams).cellSizeRecip);
31      int tempY = floor((posYi-(*pparams).globalOriginY)*(*pparams).cellSizeRecip);
32      int2 gridPos = {tempX, tempY};
33
34      //examine neighboring cells
35      double stateRates [5] = {0,0,0,0,0}; //pointer to array of {fx,fy,XSPHx, ←
    XSPHy,drhodt}
36
37      for (int y = -1;y<=1;y++) {
38          int newY = gridPos.y+y;
39          if ((newY>-1) && (newY<(*pparams).nCellsY)) {
40
41              for (int x = -1;x<=1;x++) {
42                  int newX = gridPos.x+x;
43                  if ((newX>-1) && (newX<(*pparams).nCellsX)) {
44                      int2 neighbor = {newX,newY}; //2D index in grid
45                      forcesInCell2(neighbor, index, posXi, posYi, velXi, velYi, rhoi, ←
    pressurei, pparticles, pparams, stateRates);
46                  }
47              }
48          }
49
50      }
51
52      // revised - no longer requires compute dRhoDt and SPHinfluence
53      int originalIndex = pparticles->gridParticleIndex[index];
54      pparticles->fx[originalIndex] = stateRates[0];
55      pparticles->fy[originalIndex] = stateRates[1];
56
57      pparticles->XSPHVelX[originalIndex] = stateRates[2];
58      pparticles->XSPHVelY[originalIndex] = stateRates[3];
59
60      pparticles->sorteddRhodt[index]      = stateRates[4];
61
62      return;
63  }
64
65
66  // loop over the particles in the host cell and surrounding cells; compute ←
    density
67  __device__ void forcesInCell2(int2 neighbor, int index, double posXi, double posYi, ←
    double velXi, double velYi, double rhoi, double pressurei, struct ←

```

```

particleStructure* pparticles, struct paramsType* pparams, double* stateRates←
) {
68
69 //compute 1D hash value
70 int hash = neighbor.y>(*pparams).nCellsX+neighbor.x;
71
72 //required parameters
73 double rhoRef = pparams->rhoRef;
74 double tenVMaxSq = pparams->tenVMaxSq;
75 double constantSpikyImprovedD = pparams->spikyImprovedD;
76 double constantPoly6 = pparams->constDensity;
77
78 int startIndex = pparticles->cellStart[hash];
79 //double2 forces = {0,0};
80 if (startIndex != 0xffffffff) {
81     int endIndex = pparticles->cellEnd[hash];
82     for (int ind1 = startIndex; ind1 < endIndex; ind1++) {
83         //remember to exclude self-force; stay within desired domain
84         //many SPH references cite pAB = pA - pB; where A is the primary ←
            particle.
85         //following this convention we have
86         //This gives a vector pointing from particle B to particle A
87
88         double posXj = pparticles->sortedX[ind1]; //get position of sending ←
            particles
89         double posYj = pparticles->sortedY[ind1];
90         double dx = (posXi-posXj);
91         double dy = (posYi-posYj);
92         double rSq = dx*dx+dy*dy;
93
94         if ((rSq<(*pparams).h2) && (rSq>0)) { //if they are close enough, ←
            proceede
95
96             double h = pparams->h;
97             double dist = sqrt(rSq); //expensive but necessary
98             double dvxij = velXi-pparticles->sortedVx[ind1]; //velocity ←
                of sending particle
99             double dvyij = velYi-pparticles->sortedVy[ind1];
100             double r0h = dist/h;
101             double mj = pparticles->mass[0];
102             double rhoj = pparticles->sortedRho[ind1]; //rho of sender
103             double pressurej = computePressure(rhoj,rhoRef,tenVMaxSq);
104
105             double normalizedGradientInfluence = 1/dist*spikyImprovedD(←
                constantSpikyImprovedD,r0h);
106
107             double vMorTiInner = mj*2*(pparams->mu)/(rhoi*rhoj)*←
                normalizedGradientInfluence;
108             double vMorTiX = vMorTiInner*dvxij;
109             double vMorTiY = vMorTiInner*dvyij;
110             double dirVel = dvxij*dx+dvyij*dy;
111             double rhoBarij = (rhoi+rhoj)/2; //used in XSPH as well
112
113             double vTix = vMorTiX; //it gets vTi regardless
114             double vTiy = vMorTiY;
115
116             if (dirVel<0) //it may get additional terms
117             {
118                 double muij = (h/2)*dirVel/(rSq+0.01*(h/2)*(h/2));
119
120                 double addedViscosity = mj*pparams->viscoBeta*muij*muij/←
                    rhoBarij*normalizedGradientInfluence;
121                 vTix += addedViscosity*dx; //already has 1/|rij|
122                 vTiy += addedViscosity*dy;
123             }
124
125             double sharedTerm = mj*(pressurei/(rhoi*rhoi)+pressurej/(rhoj*←
                rhoj))*normalizedGradientInfluence;

```

```

126
127         double term1X      = sharedTerm*dx;
128         double term1Y      = sharedTerm*dy;
129
130
131         //these are actually accelerations
132         stateRates[0] += -term1X+vTix; //fx
133         stateRates[1] += -term1Y+vTiy; //fy
134
135         //      forces.x += -term1X+vTix;
136         //      forces.y += -term1Y+vTiy;
137
138         //XSPH
139         double mutualInfluence = pparams->epsilon*mj/rhoBarij*poly6(↔
140             constantPoly6,r0h);
141         stateRates[2] += -mutualInfluence*dvxij; //XSPHx; dxvij = -dvxji↔
142             ; dxvj is called for in the definition
143         stateRates[3] += -mutualInfluence*dvyij; //XSPHy; dxvij = -dvxji↔
144             ; dxvj is called for in the definition
145
146         //DRho/dt
147         stateRates[4] += mj*normalizedGradientInfluence*dirVel;
148
149     }; //end excluding self
150 }; //end looping over the cell
151 }; //if start index is not empty
152
153 //return forces;
154 return;
155 }

```

### C.1.3 CUDA Kernel to Update Velocity and Position

The velocities and positions are updated using Leap-frog time integration scheme described in Section 3.2.7.

```

1 #include <cuda.h>
2 #include "math.h"
3 #include "cuda_runtime.h"
4 #include "device_launch_parameters.h"
5 #include "SPH2DCPPCuda.h"
6 #include "stdio.h"
7
8 __global__ void updateVelWithXSPH(struct particleStructure* pparticles, struct ↔
9     paramsType* pparams) {
10     int index = blockIdx.x*blockDim.x+threadIdx.x;
11
12     if (index<(*pparams).nFree){ //operate over free particles
13
14         //correct the velocity with the XSPH correction
15         pparticles->vx[index] += pparticles->XSPHVelX[index];
16         pparticles->vy[index] += pparticles->XSPHVelY[index];
17     }
18     return;
19 }

```

```

1 #include <cuda.h>
2 #include "math.h"
3 #include "cuda_runtime.h"
4 #include "device_launch_parameters.h"
5 #include "SPH2DCPPCuda.h"
6 #include <iostream>
7
8 __global__ void updateVelocity(struct particleStructure* pparticles, struct ←
    paramsType* pparams) {
9
10     int index = blockIdx.x*blockDim.x+threadIdx.x;
11
12     if (index<(*pparams).nFree){ //only operate over free particles
13         //the accelerations are stored; simply add gravity to the y-dir
14         //and incorporate the XSPH terms
15
16         //F = m a;
17         //a = F/m
18
19         //Euler 1st order
20 #if 0
21         double accelX = pparticles->fx[index]; //fx actually stores an ←
            acceleration; no need to divide by mass
22         double accelY = pparticles->fy[index]+pparams->gravity; //need to add ←
            gravity
23
24         double dt = pparams->dt;
25         double vNewx = pparticles->vx[index]+accelX*dt; //XSPH is incorporated ←
            previously
26         double vNewy = pparticles->vy[index]+accelY*dt; //
27         //store the updated velocity
28         pparticles->vx[index] = vNewx;
29         pparticles->vy[index] = vNewy;
30 #endif
31
32         //Leapfrog
33 #if 1
34
35         double dt = pparams->dt;
36         if (pparams->ind1 == 0) {
37             pparticles->vxH[index] = pparticles->vx[index] + pparticles->fx[index]←
                *dt/2;
38             pparticles->vyH[index] = pparticles->vy[index] + (pparticles->fy[←
                index]+pparams->gravity)*dt/2;
39             pparticles->vx[index] = pparticles->vxH[index] + pparticles->fx[index]←
                *dt/2;
40             pparticles->vy[index] = pparticles->vyH[index] + (pparticles->fy[←
                index]+pparams->gravity)*dt/2;
41         }
42         else {
43             pparticles->vxH[index] += pparticles->fx[index]*dt;
44             pparticles->vyH[index] += (pparticles->fy[index]+pparams->gravity)*dt←
                ;
45             pparticles->vx[index] = pparticles->vxH[index] + pparticles->fx[index]←
                *dt/2;
46             pparticles->vy[index] = pparticles->vyH[index] + (pparticles->fy[←
                index]+pparams->gravity)*dt/2;
47         }
48 #endif
49     }
50     return;
51 }
52

```

```

1 #include <cuda.h>
2 #include "math.h"
3 #include "cuda_runtime.h"
4 #include "device_launch_parameters.h"
5 #include "SPH2DCPPCuda.h"
6 #include "stdio.h"
7
8 __global__ void updatePositionFreeParticles(struct particleStructure* pparticles, ←
      struct paramsType* pparams) {
9
10     int index = blockIdx.x*blockDim.x+threadIdx.x;
11
12     if (index<(*pparams).nFree)
13     { //only operate over free particles
14
15         double dt = pparams->dt;
16         //1st order Euler
17 #if 0
18         pparticles->x[index] += pparticles->vx[index]*pparams->dt;
19         pparticles->y[index] += pparticles->vy[index]*pparams->dt;
20 #endif
21
22
23         //Leapfrog
24 #if 1
25         pparticles->x[index] += pparticles->vxH[index]*dt;
26         pparticles->y[index] += pparticles->vyH[index]*dt;
27 #endif
28
29     } //end looping over free
30
31     return;
32 }

```

## C.1.4 CUDA Kernel to Reinitialize Density

Density re-initialization computations are carried out by using the formulation provided in Section 3.2.6. Similar to other functions, the positions of the particles are read from the sorted array and their  $x, y$  addresses in the grid of bins are calculated. For every neighboring bin (including its own), a device function, `computeComponents`, is called to evaluate the components of filtered density. For the computed hash value of the neighbor cell passed in, the above mentioned state rates are computed using an all pairs approach. The function returns a `double2` type variable, `numDenom`, where `numDenom.x` = numerator and `numDenom.y` = denominator. The filtered density is given by the ratio of the numerator to the denominator.

```
1
2 #include <cuda.h>
3 #include "math.h"
4 #include "cuda_runtime.h"
5 #include "device_launch_parameters.h"
6 #include "SPH2DCPPCuda.h"
7 #include "stdio.h"
8 #include "smoothingKernels.cuh"
9
10
11 __device__ double2 computeComponents(int2 neighbor, int index, double posX, double posY, struct particleStructure* pparticles, struct paramsType* pparams);
12
13
14 __global__ void reinitializeDensity(struct particleStructure* pparticles, struct paramsType* pparams) {
15
16     int index = blockIdx.x*blockDim.x+threadIdx.x;
17
18     if (index >= (*pparams).nTotal) return;
19
20     //read particle data - host particle
21     double posXi = pparticles->sortedX[index]; //these are sorted
22     double posYi = pparticles->sortedY[index];
23
24     //get address in grid
25     int tempX = floor((posXi-(*pparams).globalOriginX)*(*pparams).cellSizeRecip);
26     int tempY = floor((posYi-(*pparams).globalOriginY)*(*pparams).cellSizeRecip);
27
28     int2 gridPos = {tempX,tempY}; // grid position of host particle
29
30     //examine neighboring cells
31     double2 numDenom = {0,0}; //need
32     double2 temp = {0,0};
```

```

33     for (int y = -1;y<=1;y++) {
34         int currentY = gridPos.y+y;
35         if ((currentY>-1) && (currentY<(*pparams).nCellsY)) {
36
37             for (int x = -1;x<=1;x++) {
38                 int currentX = gridPos.x+x;
39                 if ((currentX>-1) && (currentX<(*pparams).nCellsX)) {
40                     int2 neighbor = {currentX,currentY}; //2D index in grid
41                     temp = computeComponents(neighbor,index,posXi,posYi,←
42                         pparticles,pparams);
43                     numDenom.x += temp.x;
44                     numDenom.y += temp.y;
45                 }
46             }
47         }
48
49
50         //if particles exceede the boundaries they will have denom = 0 and rhoi = 0;
51         //In this case, they are no longer an important part of the calculation, so
52         //set their rhoi = density of a single particle
53
54         double filteredDensity = numDenom.x/numDenom.y;
55
56         if (numDenom.y==0)
57         {
58             //maybe keeping it the same is the answer
59             filteredDensity = 2*pparticles->mass[0]*(pparams).constDensity;
60             printf("particle exceeded boundary\n");
61         }
62
63
64         if (filteredDensity==0) {
65             printf("something is wrong\n");
66         }
67
68
69         //store the density in a temporary array
70         pparticles->sortedRhoFiltered[index] = filteredDensity; //sortedRhoFiltered ←
71         no longer exists
72     }
73
74
75     // loop over the particles in the host cell and surrounding cells; compute ←
76     density
77     __device__ double2 computeComponents(int2 neighbor,int index,double posXi,double←
78     posYi, struct particleStructure* pparticles, struct paramsType* pparams) {
79
80         //compute 1D hash value
81         int hash = neighbor.y*(pparams).nCellsX+neighbor.x;
82
83         int startIndex = pparticles->cellStart[hash];
84         double2 numDenom = {0,0};
85         if (startIndex != 0xffffffff) {
86             int endIndex = pparticles->cellEnd[hash];
87
88             for (int ind1 = startIndex; ind1 < endIndex; ind1++) {
89                 //no reason to include self in drhodt
90                 double posXj = pparticles->sortedX[ind1]; //get position of sending ←
91                 particles
92                 double posYj = pparticles->sortedY[ind1];
93                 double m2 = pparticles->mass[0]; //mass; right now these are ←
94                 identical for all particles
95
96                 //The kernel is W =
97                 double dxij = (posXi-posXj);
98                 double dyij = (posYi-posYj);

```

```

95
96     double rSq    = dxij*dxij+dyij*dyij;
97     double diffSq = (*pparams).h2-rSq;
98     if (diffSq>=0)
99     {
100         double rhoj = pparticles->sortedRho[ind1];
101         double dist = sqrt(rSq);
102         double r0h  = dist/pparams->h;
103
104         double kernelInfluence = poly6(pparams->constDensity, r0h);
105         numDenom.x      += kernelInfluence*m2;           //numerator
106         numDenom.y      += kernelInfluence*m2/rhoj;     //denominator
107
108     }; //end the for loop
109 }; //end the if statement
110 };
111
112 return numDenom;}

```

## C.2 Predictor-Corrector MATLAB Code Snippets

Based on the candidate Eigenvalue chosen from the prediction step, this function generates the initial condition which then undergoes Floquet analysis.

```

1 function [actualEigVal, ActualEpsilon, Actualtheta] = spectralEigenvalue(L, epsilon, ←
    CandidateEigVal)
2
3
4 global ComplexEnvelope dx x SurfaceElevation Xplot
5 global k0 w0 A
6
7 %Discretization
8 M = 500; % Number of discretized Points
9 X = linspace(0,0+L,M); % Spatial Domain
10 dx = X(2)-X(1);
11 Xplot = linspace(0,0+3*L,3*M);
12
13 % Evaluate the spectral parameters
14 theta=0;
15 e(1)=epsilon*exp(1i*theta); % 1i;
16 e(2)=conj(e(1));
17 sigma(1)=1;sigma(2)=-1;
18
19 lamda(1)= CandidateEigVal;
20 lamda(2)=conj(lamda(1));
21
22 for k=1:2
23     K(k)=-2*sqrt(A^2+lamda(k)^2);
24     Omega(k)=2*lamda(k)*K(k);
25     delplus(k)=pi+1i*log(lamda(k)-(1/2)*K(k))+1i*log(sigma(k)*lamda(k)-((-1)^k←
        *1/2*K(k)));
26     delminus(k)=pi+1i*log(lamda(k)+(1/2)*K(k))+1i*log(sigma(k)*lamda(k)-((-1)^k←
        *1/2*K(k)));
27 end
28 k=0;
29 for k=1:2
30     for j=1:2
31         if k==j
32             Tau(j,j)=1/2+(1i/pi)*log(K(j)^2/e(j));
33         else Tau(k,j)=(1i/(2*pi))*log((1+lamda(k)*lamda(j)+(1/4)*K(k)*K(j))/(1+←
            lamda(k)*lamda(j)-(1/4)*K(k)*K(j)));
34         end
35     end
36 end
37
38
39 Tau=pi*Tau;
40 j=0;k=0;
41 Cg=1/2*w0/k0;
42 beta=w0/(8*k0^2);
43 lambda=sqrt(2)*k0^2;
44
45 T=[0];
46
47 x=Xplot;
48 t=T;
49
50 % Evaluate complex envelope based on the candidate eigenvalue at T = 0
51 for j=1:length(x)

```

```

52     for k=1:length(t)
53         S1=0;S2=0;
54         for m1=-10:10
55             for m2=-10:10
56                 S1=S1+exp(1i*(m1*K(1)*x(j)+m2*K(2)*x(j)+m1*Omega(1)*t(k)+m2*Omega(2)*t(k)+m1*delplus(1)+m2*delplus(2))+(1i/1)*(m1*m1*Tau(1,1)+...
57                     ...
58                     +m2*m1*Tau(2,1)+m1*m2*Tau(1,2)+m2*m2*Tau(2,2)));
59                 S2=S2+exp(1i*(m1*K(1)*x(j)+m2*K(2)*x(j)+m1*Omega(1)*t(k)+m2*Omega(2)*t(k)+m1*delminus(1)+m2*delminus(2))+(1i/1)*(m1*m1*Tau(1,1)+...
60                     ...
61                     +m2*m1*Tau(2,1)+m1*m2*Tau(1,2)+m2*m2*Tau(2,2)));
62             end
63         end
64         U(j)=A*(((S2/S1)*exp(1i*2*(A^2)*t(k))));
65     end
66 end
67
68
69 ComplexEnvelope=U(1:1,1:M);
70 size(ComplexEnvelope)
71 save initalCond U
72 SurfaceElavation=U.*exp(1i*(k0*x)); % Complex Free Surface Elevation
73 x = x(1:1,1:M)
74 Plots(U,dx,Xplot,SurfaceElavation);
75
76 SolveEigValue;
77
78 [actualEigval,Actualepsilon,Actualtheta] = CheckEigValue(lamda(1));
79
80 save CorrectedParameters actualEigval Actualepsilon Actualtheta

```

```

1 function SolveEigValue
2
3 % The initial condition is run through Floquet Analysis at every grid point in the domain of search to generate the eigenvalue map
4
5 global ComplexEnvelope dx x SurfaceElavation
6
7 % Domain of search
8 lambdaR = -1.5:0.1:1.5;
9 lambdaI = -1.5:0.1:1.5;
10 i=1;
11
12 options = optimset('Display','off'); % Turn off display
13 for ind1=1:length(lambdaR)
14     for ind2=1:length(lambdaI)
15
16         % Solve for 1/2* Tr(Monodromy Matrix) = +1
17         [q,fval,exitflag] = fsolve(@floquet1_mex,lambdaR(ind1)+1i*lambdaI(ind2),options);
18
19
20         if exitflag==1
21             EigVal(i)=q;
22             i=i+1;
23         end
24     end
25 end
26
27 m=length(EigVal);

```

```

28 ind1=0;ind2=0;i=m+1;
29 for ind1=1:length(lambdaR)
30     for ind2=1:length(lambdaI)
31
32         % Solve for 1/2* Tr(Monodromy Matrix) = -1
33         [q,fval,exitflag] = fsolve(@floquet2_mex,lambdaR(ind1)+1i*lambdaI(ind2),←
            options);
34
35         if exitflag==1
36             EigVal(i)=q;i=i+1;
37         end
38     end
39 end
40
41
42 save EigVal EigVal
43 fName = 'eigenvalues_1.txt';
44 fid = fopen(fName,'wt');
45 fclose(fid);
46 dlmwrite(fName,EigVal,'-append',... %% Print the matrix
47     'delimiter','\t',...
48     'newline','pc');
49
50 figure
51 plot(real(EigVal),imag(EigVal),'kx')
52 grid on
53 xlabel('real')
54 ylabel('imaginary')
55 hold on

```

```

1 function F=floquet1(l)
2
3 global ComplexEnvelope dx x
4 S=[1+0*1i 1+0*1i;
5     1+0*1i 1+0*1i];
6 %tic
7 lengthComplexEnv = length(ComplexEnvelope);
8 for ind=1:length(x)
9     k=sqrt(-((abs(ComplexEnvelope(lengthComplexEnv+1-ind)))^2+1^2));
10
11     T=[cosh(k*dx)-(1i*1/k*sinh(k*dx)) ComplexEnvelope(lengthComplexEnv+1-ind)/k*←
        sinh(k*dx);
12     -(ComplexEnvelope(lengthComplexEnv+1-ind))/k*sinh(k*dx) cosh(k*dx)+(1i*1*←
        /k*sinh(k*dx))];
13     %S=S*T;
14
15     S=[S(1,1)*T(1,1)+S(1,2)*T(2,1) S(1,1)*T(1,2)+S(1,2)*T(2,2);
16     S(2,1)*T(1,1)+S(2,2)*T(2,1) S(2,1)*T(1,2)+S(2,2)*T(2,2)];
17 end
18 %toc
19 F=1/2*(S(1,1)+S(2,2))-1;

```

```

1 %program to numerically verify solutions of the NLSE
2 %equation defined as shown
3 %
4 %i*u_t+sig1*u_xx+sig2*abs(u)^2 u=0
5 %
6 %
7 %res = verifySolutionNLSE(u,s)

```

```

8 %where
9 %u is the proposed solution with time running down the columns
10 %and space across the rows
11 %
12 %s is a structure with fields:
13 %s.dt - delta time
14 %s.dx - delta space
15 %s.sig1 - parameter of the equation
16 %s.sig2 - parameter of the equation
17 %
18 %
19 %and res is the residual from the numerical differentiation
20 %res should be identially zero for actual solutions and
21 %perfect differentiation
22
23
24
25 function respercentage = verifySolutionNLSE(u,s)
26
27 orderOfDifferentiation = 8; %2, 4, 6, or 8
28 oodp1 = orderOfDifferentiation+1;
29 oodo2 = orderOfDifferentiation/2;
30
31 [nt,nx] = size(u);
32
33 c{2} = [-1/2 0 1/2];
34 c{4} = [1/12 -2/3 0 2/3 -1/12];
35 c{6} = [-1/60 3/20 -3/4 0 3/4 -3/20 1/60];
36 c{8} = [1/280 -4/105 1/5 -4/5 0 4/5 -1/5 4/105 -1/280];
37
38 %O(6) central 1st derivative
39 u_t = zeros(nt-oodp1+1,nx);
40 for ind1 = 1:orderOfDifferentiation+1
41     u_t = u_t+c{orderOfDifferentiation}(ind1)*u(ind1:nt-oodp1+ind1,:);
42 end
43 u_t = u_t/s.dt;
44
45
46 %O(6) central 2nd derivative
47 c{2} = [1 -2 1];
48 c{4} = [-1/12 4/3 -5/2 4/3 -1/12];
49 c{6} = [1/90 -3/20 3/2 -49/18 3/2 -3/20 1/90];
50 c{8} = [-1/560 8/315 -1/5 8/5 -205/72 8/5 -1/5 8/315 -1/560];
51
52 u_xx = zeros(nt,nx-oodp1+1);
53 for ind1 = 1:orderOfDifferentiation+1
54     u_xx = u_xx+c{orderOfDifferentiation}(ind1)*u(:,ind1:nx-oodp1+ind1);
55 end
56 u_xx = u_xx/(s.dx^2);
57
58 %now reduce the size of each component to match
59 u = u(oodo2+1:nt-oodo2,oodo2+1:nx-oodo2);
60 u_t = u_t(:,oodo2+1:nx-oodo2);
61 u_xx = u_xx(oodo2+1:nt-oodo2,:);
62
63
64 res = li.*u_t+s.sig1*u_xx+s.sig2*(u.*conj(u)).*u;
65
66 sumsumRes = sum(sum(abs(res)))
67 sumsumU = sum(sum(abs(u)))
68 respercentage = sumsumRes/sumsumU*100;
69 fprintf(1,'sum of residuals is %5.5f\n',sumsumRes);
70 fprintf(1,'sum of solution is %5.5f\n',sumsumU);
71 fprintf(1,'percentage of res from total is %5.5f\n %',sumsumRes/sumsumU*100);

```

## Bibliography

- [1] N. N. Akhmediev and A. Ankiewicz. *Solitons: nonlinear pulses and beams*, volume 4. Chapman & Hall London, 1997.
- [2] N. N. Akhmediev, A. Ankiewicz, and J. M. Soto-Crespo. Rogue waves and rational solutions of the nonlinear schrödinger equation. *Physical Review E*, 80(2):026601, 2009.
- [3] N. N. Akhmediev, A. Ankiewicz, and M. Taki. Waves that appear from nowhere and disappear without a trace. *Physics Letters A*, 373(6):675–678, 2009.
- [4] N. N. Akhmediev and V. I. Korneev. Modulation instability and periodic solutions of the nonlinear schrödinger equation. *Theoretical and Mathematical Physics*, 69(2):1089–1093, 1986.
- [5] N. N. Akhmediev, J. M. Soto Crespo, A. Ankiewicz, et al. How to excite a rogue wave. *Physical Review A*, 80(043818), 2009.
- [6] B. Baschek and J. Imai. Rogue wave observations off the us west coast. *Oceanography*, 24, 2011.
- [7] G. K. Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.
- [8] T. B. Benjamin and J. E. Feir. The disintegration of wave trains on deep water. *J. Fluid Mech*, 27(3):417–430, 1967.
- [9] C. Chabalko and B. Balachandran. GPU based simulation of physical system characterized by mobile discrete interactions. *B. H. V. Topping and P. Ivanyi (Editor), “Developements in Parallel, Distributed, Grid and Cloud Computing for Engineering”, Saxe-Coburg Publications, Stirlingshire, UK, Chapter 5*, pages 95–124, 2013.
- [10] A. Chabchoub, N. Hoffmann, M. Onorato, and N. N. Akhmediev. Super rogue waves: Observation of a higher-order breather in water waves. *Phys. Rev. X*, 2:011015, Mar 2012.

- [11] A. Chabchoub, N. P. Hoffmann, and N.N. Akhmediev. Rogue wave observation in a water wave tank. *Phys. Rev. Lett.*, 106:204502, May 2011.
- [12] R. A. Dalrymple and R. G. Dean. *Water wave mechanics for engineers and scientists*. Prentice-Hall, 1991.
- [13] M. H. Dao, H. Xu, E. S. Chan, and P. Tkalich. Numerical modelling of extreme waves by smoothed particle hydrodynamics. *Natural Hazards and Earth System Science*, 11(2):419–429, 2011.
- [14] L. Draper. Freak ocean waves. *Weather*, 21(1):2–4, 1966.
- [15] A. I. Dyachenko and V. E. Zakharov. Modulation instability of stokes wave- $\zeta$  freak wave. *Journal of Experimental and Theoretical Physics Letters*, 81(6):255–259, 2005.
- [16] K. B. Dysthe. Note on a modification to the nonlinear schrodinger equation for application to deep water waves. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 369(1736):105–114, 1979.
- [17] K. B. Dysthe, H.E. Krogstad, and P. Müller. Oceanic rogue waves. *Annu. Rev. Fluid Mech.*, 40:287–310, 2008.
- [18] C. Fochesato, S. Grilli, and F. Dias. Numerical modeling of extreme rogue waves generated by directional energy focusing. *Wave Motion*, 44(5):395–416, 2007.
- [19] R. Grimshaw. *Nonlinear ordinary differential equations*, volume 2. CRC Press, 1991.
- [20] O. R. Its and V. P. Kotliarov. Explicit formulas for the solutions of a nonlinear schrodinger equation. *Akademiia Nauk Ukraini koi RSR Dopovidi Seriia Fiziko Matematichni ta Tekhnichni Nauki*, 1:965–968, 1976.
- [21] R. S. Johnson. *A modern introduction to the mathematical theory of water waves*, volume 19. Cambridge University Press, 1997.
- [22] C. Kharif and E. Pelinovsky. Physical mechanisms of the rogue wave phenomenon. *European Journal of Mechanics-B/Fluids*, 22(6):603–634, 2003.
- [23] C. Kharif, E. Pelinovsky, and A. Slunyaev. *Rogue waves in the ocean. Advances in Geophysical and Environmental Mechanics and Mathematics*. Springer-Verlag, Berlin-Heidelberg, 2009.
- [24] C. Kharif, E. Pelinovsky, T. Talipova, and A. Slunyaev. Focusing of nonlinear wave groups in deep water. *Journal of Experimental and Theoretical Physics Letters*, 73(4):170–175, 2001.

- [25] B. Kibler, J. Fatome, C. Finot, G. Millot, F. Dias, G. Genty, N. N. Akhmediev, and J. M. Dudley. The peregrine soliton in nonlinear fibre optics. *Nature Physics*, 6(10):790–795, 2010.
- [26] G. R. Liu and M. B. Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003.
- [27] E. Y. Lo and S. Shao. Simulation of near-shore solitary wave mechanics by an incompressible sph method. *Applied Ocean Research*, 24(5):275–286, 2002.
- [28] Y. C. Ma and M. J. Ablowitz. The periodic cubic schrödinger equation. *Studies in Applied Mathematics*, 65:113–158, 1981.
- [29] J. C. Martin and W. J. Moyce. Part iv. an experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 244(882):312–324, 1952.
- [30] J. J. Monaghan. On the problem of penetration in particle methods. *Journal of Computational physics*, 82(1):1–15, 1989.
- [31] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.
- [32] J. J. Monaghan. Simulating free surface flows with sph. *Journal of computational physics*, 110(2):399–406, 1994.
- [33] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.
- [34] J. J. Monaghan and A. Kos. Solitary waves on a cretan beach. *Journal of waterway, port, coastal, and ocean engineering*, 125(3):145–155, 1999.
- [35] J. P. Morris, P. J. Fox, and Y. Zhu. Modeling low reynolds number incompressible flows using sph. *Journal of computational physics*, 136(1):214–226, 1997.
- [36] W. M. Moslem, P. K. Shukla, and B. Eliasson. Surface plasma rogue waves. *EPL (Europhysics Letters)*, 96(2):25002, 2011.
- [37] M. Muller, D. Charypar, and M. Gross. Particle based fluid simulation for interactive applications. *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2003.
- [38] A. R. Osborne. The hyperelliptic inverse scattering transform for the periodic, defocusing nonlinear schroedinger equation. *Journal of Computational Physics*, 109(1):93–107, 1993.
- [39] A. R. Osborne. The random and deterministic dynamics of Šrogue wavesŠ in unidirectional, deep-water wave trains. *Marine structures*, 14(3):275–293, 2001.

- [40] A. R. Osborne. *Nonlinear Ocean Waves & the Inverse Scattering Transform*, volume 97. Access Online via Elsevier, 2010.
- [41] A. R. Osborne, M. Onorato, and M. Serio. The nonlinear dynamics of rogue waves and holes in deep-water gravity wave trains. *Physics Letters A*, 275(5):386–393, 2000.
- [42] E. Pelinovsky, T. Talipova, and C. Kharif. Nonlinear-dispersive mechanism of the freak wave formation in shallow water. *Physica D: Nonlinear Phenomena*, 147(1):83–94, 2000.
- [43] M. Remoissenet. *Waves called solitons: concepts and experiments*. Springer, 1999.
- [44] V. Ruban, Y. Kodama, M. Ruderman, J. Dudley, R. Grimshaw, P. V. E. McClintock, M. Onorato, C. Kharif, E. Pelinovsky, T. Soomere, et al. Rogue waves—towards a unifying concept?: Discussions and debates. *The European Physical Journal-Special Topics*, 185(1):5–15, 2010.
- [45] M. Rudman, P. Cleary, J. Leontini, M. Sinnott, and M. Prakash. Rogue wave impact on a semi-submersible offshore platform. In *ASME 2008 27th International Conference on Offshore Mechanics and Arctic Engineering*, pages 887–894. American Society of Mechanical Engineers, 2008.
- [46] A. Shabat and V. E. Zakharov. Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media. *Soviet Physics JETP*, 34:62–69, 1972.
- [47] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM, 1968.
- [48] A. Slunyaev, C. Kharif, E. Pelinovsky, and T. Talipova. Nonlinear wave focusing on water of finite depth. *Physica D: Nonlinear Phenomena*, 173(1):77–96, 2002.
- [49] A. O. Smirnov. Solution of a nonlinear schrödinger equation in the form of two-phase freak waves. *Theoretical and Mathematical Physics*, 173(1):1403–1416, 2012.
- [50] D. R. Solli, C. Ropers, P. Koonath, and B. Jalali. Optical rogue waves. *Nature*, 450(7172):1054–1057, 2007.
- [51] E. R. Tracy. *Topics in nonlinear wave theory with applications*. dissertation, University of Maryland, College Park, 1984.
- [52] E. R. Tracy and H. H. Chen. Nonlinear self-modulation: An exactly solvable model. *Phys. Rev. A*, 37:815–839, Feb 1988.

- [53] N. K. Vitanov, A. Chabchoub, and N. Hoffmann. Deep-water waves: On the nonlinear schrödinger equation and its solutions. *arXiv preprint arXiv:1301.0990*, 2013.
- [54] A. Vorobyev. *A Smoothed Particle Hydrodynamics Method for the Simulation of Centralized Sloshing Experiments*. KIT Scientific Publishing, 2012.
- [55] A. Vorobyev and V. Kriventsev. Particle method for liquid-in-liquid interaction simulation. 2009.
- [56] H. C. Yuen and B. M. Lake. Instabilities of waves on deep water. *Annual Review of Fluid Mechanics*, 12(1):303–334, 1980.