Wave breaking in the BBM-equation

Anders M. Norevik

MASTER THESIS IN APPLIED AND COMPUTATIONAL MATHEMATICS

Department of Mathematics
University of Bergen
Norway
January 14, 2019

Preface

This thesis is my final work as a student at the University of Bergen. It has been hard work, and I would like to thank my supervisor Henrik Kalisch for all his invaluable help with everything from fluid mechanics theory to numerics, as well as all others who has given me helpful input during the work process. I would also like to thank my family for all their love and support during this time.

All numerical computations and figures in this thesis were done using Matlab R2015b and Microsoft Excel.

Contents

List of Figures

1 INTRODUCTION/BACKGROUND

This thesis has been written based mainly on the work of Henrik Kalisch and Magnar Bjørkavåg in their article "Wave breaking in Boussinesq models for undular bores" [1] from 2011.

A bore is a fluid phenomenon that can be described in rough terms as a flow of fluid moving along on top of "another" body of fluid. More formally described by quoting Kalisch and Bjørkavåg in [1] (p. 1570): "In its simplest description, a bore is a transition between two uniform free-surface flows with different flow depths". In Figure 1.1 below, a bore with height a_0 travelling onto initially undisturbed water with depth h_0 is shown.

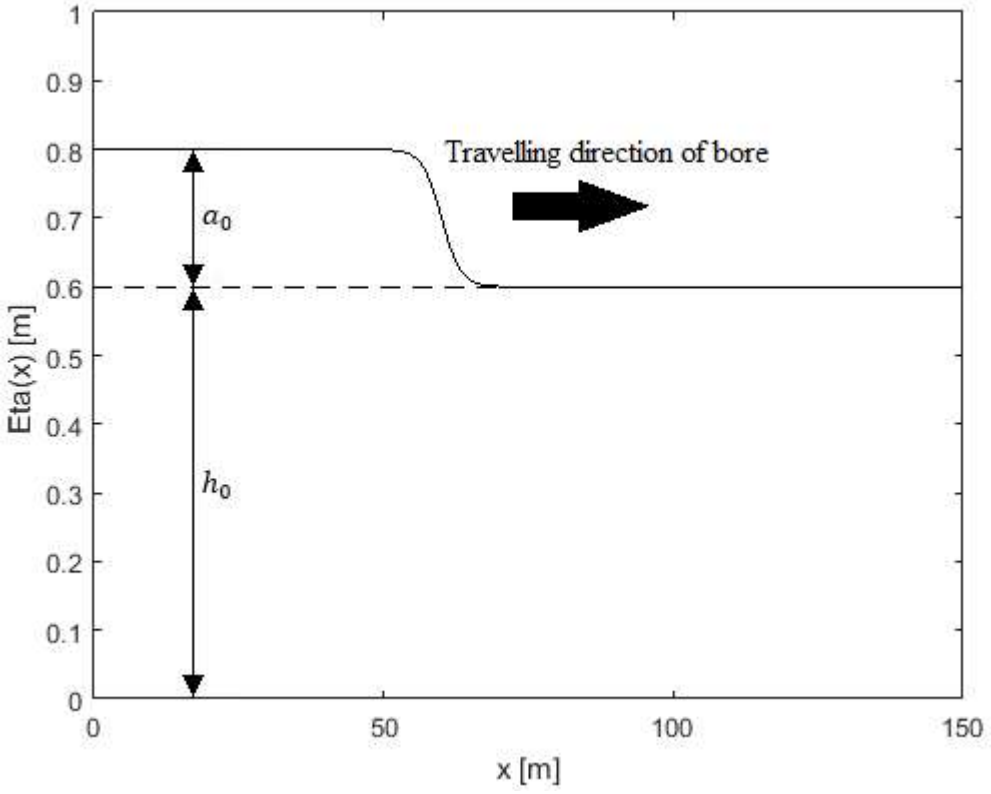

Figure 1.1 - A travelling bore

From this figure, we can define an important property for bores, the bore strength $\alpha = a_0/h_0$.

The background for the work in [1] was the experimental findings from 1935 by Favre [2], who studied travelling bores in an outdoor wave tank. Favre observed several interesting phenomena related to the above mentioned bore strength α. Quoting from Brun in [3] (p. 1):

"As found by Favre (…) by wave tank experiments, the strength of the bore can be determined by the ratio of the incident water level above the undisturbed water depth to the undisturbed water depth. Denoting this ratio by α, bores can occur in one of three categories: If α is less than 0.28 the bore is purely undular, and will feature oscillations downstream of the bore front. If α is between 0.28 and 0.75 the bore will continue to feature oscillations, but one or more waves behind the bore front will start to break. If α is greater than 0.75 the bore

is completely turbulent, and can no longer be described by the standard potential flow theory."

In other words, with regards to wave-breaking, Favre found a critical bore strength $\alpha_{critical}$ = 0.28, and for $\alpha \geq \alpha_{critical}$, he observed the onset of wave-breaking in waves behind the bore front.

In [1], Kalisch and Bjørkavåg wanted to investigate this value of $\alpha_{critical}$, by using a Boussinesq system to describe the bore flowing onto undisturbed water, creating a numerical scheme and simulating the time-evolution of the bore. The efforts of Kalisch and Bjørkavåg resulted in the finding of a critical bore strength $\alpha_{critical}$ = 0.379.

The main goal of this thesis is to study the onset of wave-breaking in undular bores, by investigating the value of $\alpha_{critical}$ found by Favre and Kalisch/Bjørkavåg. This is done by simulating numerically the time evolution of an undular bore expressed as an initial/boundary value problem using the non-linear dispersive Benjamin-Bona-Mahony equation from shallow water theory, more specificly the form of the BBM-equation (abbreviation used from this point on) found in [4].

The BBM-equation is a non-linear, third order partial difference equation whose solution $\eta(x,t)$ [m] describes the deflection of the free water surface relative to some zero-reference elevation.

As a numerical simulation of the bore initial/boundary value problem is finished, a numerical approximative solution for the surface profile $\eta(x,t)$ is available, and since then the positions of undulations/waves are known for all times, it is relatively straight-forward to calculate wave phase speed U [m/s] by simple speed = distance/time reasoning.

The theoretical derivation of the KdV- and BBM-equation found in [4] also includes equations for the horizontal fluid velocity u [m/s], which it should be emphasized is not identical to the wave phase speed U. By comparing the velocities U and u at the top of wave-crests at specific times, and introducing a wave-breaking criterion u > U, the onset of wave-breaking can then be studied.

The full MATLAB source code for "bbm_solver", used for simulating the undular bore, is included in appendix A. The full MATLAB source code of "bbm_error_checker", used for checking order/convergence speed of the numerical method, is included in appendix B. The idea is that for anyone interested, it should be possible to copy the entire text/code from the appendices into a MATLAB script or a script in any other program compatible with the programming language of MATLAB, and then run the programs.

2 WAVE BREAKING IN THE BBM-EQUATION

2.1 Derivation of the BBM-equation

2.1.1 Summary of the derivation done by Whitham

Chapter 2.1.2 follows and elaborates the steps in Whitham's formal theoretical derivations in [4], pages 464-466, leading ultimately to the Korteweg-deVries (KdV) equation and the BBM-equation.

Whitham's derivation assumes irrotational and incompressible flow, and viscosity is neglected. Further, it is based upon the assumption that the velocity potential $\varphi(x, z, t)$, with unit $[m^2/s]$ may be written as an asymptotic expansion in the "parameter" z, that is

$$\varphi = \sum_{n=0}^{\infty} z^n f_n(x, t) \tag{2.1}$$

Substituting this expansion for φ into the Laplace equation and applying the bottom boundary condition $\varphi_z = 0$ on $z = 0$ from the classical surface wave problem, we arrive at a new form of the expansion for $\varphi(x, z, t)$:

$$\varphi = \sum_{m=0}^{\infty} (-1)^m \cdot \frac{z^{2m}}{(2m)!} \cdot \frac{\partial^{2m} f}{\partial x^{2m}} \tag{2.2}$$

where with regards to the functions $f_n(x, t)$, it is now only dependent on the first function $f = f_0(x, t)$ from the original expansion.

At this point Whitham's derivation turns to non-dimensionalizing, which is an often used tool that makes important non-dimensional parameters explicitly present in the equations at hand. This in turn enables us to determine when different terms in an equation is small, and create simpler equations which approximate the original equation well under certain (physical) conditions, by dropping terms in higher order of the above mentioned dimensionless parameter groups that appears from the scaling between dimensional and non-dimensional variables. See e.g. [5] for a more detailed explanation of the method of non-dimensionalizing.

The expansion (2.2) above is non-dimensionalized, and we then get an expansion for the non-dimensional velocity potential $\tilde{\varphi}(\tilde{x}, \tilde{z}, \tilde{t})$. The latter is substituted for $\tilde{\varphi}$ into the free surface boundary conditions of the non-dimensional version of the surface wave problem. From this, and some justified simplication by dropping of terms in higher orders of parameters α and β arising from the non-dimensionalizing, we arrive at a variant of the non-dimensional Boussinesq equations. Then, by specializing to a right-travelling wave, the non-dimensional KdV-equation is obtained. By re-dimensionalizing the KdV-equation and then doing an approximation on one of the x-derivatives in the dispersive correction term, we are the finally at the dimensional BBM-equation as given in [4]:

$$\eta_t + c_0\left(1 + \frac{3}{2}\frac{\eta}{h_0}\right)\eta_x - \frac{\gamma}{c_0}\eta_{xxt} = 0 \tag{2.3}$$

where $c_0 = \sqrt{gh_0}$ is the shallow water approximation for the linear surface wave phase speed [4] [6] with units [m/s], h_0 is the height of the undisturbed water surface in [m], γ is denoted in [4] as the constant $\frac{1}{6}c_0h_0^2$, and finally the solution of the equation, $\eta = \eta(x,t)$, is the surface deflection in [m].

If we for simplicity set $g = 1$ and $h_0 = 1$, which also implies $c_0 = 1$, equation (2.3) becomes:

$$\eta_t + \eta_x + \frac{3}{2}\eta\eta_x - \frac{1}{6}\eta_{xxt} = 0 \tag{2.4}$$

2.1.2 The derivation in detail

We start out by pointing out that z is measured from the horizontal bottom. Note that we assume we have potential flow, so the velocity field is given by $\mathbf{u} = \nabla\varphi$, and further frictionless flow, so the no-slip bottom boundary condition $\varphi_x = 0$ on $z = 0$ does not apply. We need to find a solution to the Laplace equation.

$$\varphi_{xx} + \varphi_{zz} = 0 \tag{2.5}$$

with the bottom boundary condition $\varphi_z = 0$ on $z = 0$. This boundary condition is the "no through-flow"-condition, you cannot have water flowing *through* the physical bottom, so the vertical velocity component $\varphi_z = w$ [m/s] must be equal to zero at the bottom.

Quoting Whitham in [4] (p. 464): "The shallow water theory, with φ_x approximately independent of z, and the small total depth both suggest an expansion"

$$\varphi = \sum_{n=0}^{\infty} z^n f_n(x,t)$$

Writing out the first terms for the purpose of clarity, we have

$$\varphi = z^0 f_0(x,t) + z^1 f_1(x,t) + z^2 f_2(x,t) + z^3 f_3(x,t) + \cdots \tag{2.6}$$

and we recognize the shape of an asymptotic expansion, each term in the asymptotic sequence in z being dominated by the previous term as $z \to 0$.

Substituting the expansion into the Laplace equation, we then have

$$\frac{\partial^2}{\partial x^2}\left(\sum_{n=0}^{\infty} z^n f_n(x,t)\right) + \frac{\partial^2}{\partial z^2}\left(\sum_{n=0}^{\infty} z^n f_n(x,t)\right) = 0$$

At this point, it is convenient to calculate φ_z, which from the expansion for φ will be

$$\varphi_z = 0 \cdot f_0(x,t) + 1 \cdot z^{1-1} \cdot f_1(x,t) + 2 \cdot z^{2-1} \cdot f_2(x,t) + \cdots$$

corresponding to the expansion/sum

$$\varphi_z = \sum_{n=1}^{\infty} n \cdot z^{n-1} f_n(x,t) \tag{2.7}$$

which looks like

$$\varphi_z = f_1(x,t) + 2 \cdot z \cdot f_2(x,t) + 3 \cdot z^2 \cdot f_3(x,t) + \cdots$$

Now, applying the bottom boundary condition $\varphi_z = 0$ on $z = 0$, or $\varphi_z(x,0) = 0$, we get

$$\varphi_z(x,0) = f_1(x,t) + 2 \cdot 0 \cdot f_2(x,t) + 3 \cdot 0^2 \cdot f_3(x,t) + \cdots = 0$$

and all terms after the first one vanish, leaving us with

$$f_1(x,t) = 0$$

We need expressions for φ_{zz} and φ_{xx} to put into the Laplace equation. We find φ_{zz} by differentiation of equation (2.7) to be

$$\varphi_{zz} = \sum_{n=1}^{\infty} n \cdot (n-1) \cdot z^{n-2} f_n(x,t)$$

Writing out the terms of this expression, we have

$$\varphi_{zz} = 1 \cdot (1-1) \cdot z^{1-2} \cdot f_1(x,t) + 2 \cdot (2-1) \cdot z^{2-2} \cdot f_2(x,t) + 3 \cdot (3-1) \cdot z^{3-2} \cdot f_3(x,t) + \cdots$$

and since the first term (corresponding to n = 1) disappears, we have the expansion

$$\varphi_{zz} = \sum_{n=2}^{\infty} n \cdot (n-1) \cdot z^{n-2} f_n(x,t)$$

and rearranging the indices to start at n = 0, we get

$$\varphi_{zz} = \sum_{n=0}^{\infty} (n+2) \cdot (n+1) \cdot z^n f_{n+2}(x,t)$$

In finding φ_{xx}, we note that z is a constant when differentiating w.r.t x, and write it as

$$\varphi_{xx} = \sum_{n=0}^{\infty} z^n f_n(x,t)_{xx}$$

Substituting the expansions for φ_{zz} and φ_{xx} into the Laplace equation then gives

$$\sum_{n=0}^{\infty} z^n f_n(x,t)_{xx} + \sum_{n=0}^{\infty} (n+2) \cdot (n+1) \cdot z^n f_{n+2}(x,t) = 0$$

which can be rewritten as

$$\sum_{n=0}^{\infty} z^n \left[f_n(x,t)_{xx} + (n+2)\cdot(n+1)\cdot f_{n+2}(x,t) \right] = 0$$

Now since the above equation must hold for all z and n, which could obviously be non-zero and thus give non-zero values of z^n, we deduce that we must have

$$f_n(x,t)_{xx} + (n+2)\cdot(n+1)\cdot f_{n+2}(x,t) = 0$$

providing us with a recursive relationship for the $f_n(x,t)$-functions. Note that we already know that $f_1(x,t) = 0$, which leads to $f_1(x,t)_x = 0$ and $f_1(x,t)_{xx} = 0$. We want to investigate the $f_n(x,t)$-functions, and we write out the above equation for a few values of n, starting at n = 0.

$$n = 0: \quad f_0(x,t)_{xx} + 2\cdot 1\cdot f_2(x,t) = 0 \Rightarrow f_2(x,t) = -\frac{f_0(x,t)_{xx}}{2\cdot 1}$$

$$n = 1: \quad f_1(x,t)_{xx} + 3\cdot 2\cdot f_3(x,t) = 0 \Rightarrow f_3(x,t) = 0$$

$$n = 2: \quad f_2(x,t)_{xx} + 4\cdot 3\cdot f_4(x,t) = 0 \Rightarrow f_4(x,t) = -\frac{f_2(x,t)_{xx}}{4\cdot 3}$$

$$f_4(x,t) = -\frac{1}{4\cdot 3}\cdot\left(-\frac{f_0(x,t)_{xxxx}}{2\cdot 1}\right)$$

$$f_4(x,t) = \frac{f_0(x,t)_{xxxx}}{4\cdot 3\cdot 2\cdot 1}$$

$$n = 3: \quad f_3(x,t)_{xx} + 5\cdot 4\cdot f_5(x,t) = 0 \Rightarrow f_5(x,t) = 0$$

$$n = 4: \quad f_4(x,t)_{xx} + 6\cdot 5\cdot f_6(x,t) = 0 \Rightarrow f_6(x,t) = -\frac{f_4(x,t)_{xx}}{6\cdot 5}$$

$$f_6(x,t) = -\frac{1}{6\cdot 5}\cdot\left(\frac{f_0(x,t)_{xxxxxx}}{4\cdot 3\cdot 2\cdot 1}\right)$$

$$f_6(x,t) = -\frac{f_0(x,t)_{xxxxxx}}{6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}$$

We notice that all the odd-number functions vanish, and also that all the even-numbered functions can be written in terms of derivatives of $f_0(x,t)$.

Returning to our original expansion for φ

$$\varphi = z^0 f_0(x,t) + z^1 f_1(x,t) + z^2 f_2(x,t) + z^3 f_3(x,t) + \cdots$$

we write it up again, taking into account that all terms where an odd-numbered function is a factor vanish

$$\varphi = z^0 f_0(x,t) + z^2 f_2(x,t) + z^4 f_4(x,t) + z^6 f_6(x,t) + \cdots$$

Now we use the expressions in terms of derivatives of $f_0(x,t)$ found for the even-numbered functions, to get

$$\varphi = f_0(x,t) + z^2\left(-\frac{f_0(x,t)_{xx}}{2\cdot 1}\right) + z^4\left(\frac{f_0(x,t)_{xxxx}}{4\cdot 3\cdot 2\cdot 1}\right) + z^6\left(-\frac{f_0(x,t)_{xxxxxx}}{6\cdot 5\cdot 4\cdot 3\cdot 2\cdot 1}\right) + \cdots$$

which can be expressed by the general sum

$$\varphi = \sum_{m=0}^{\infty} (-1)^m \cdot \frac{z^{2m}}{(2m)!} \cdot \frac{\partial^{2m} f}{\partial x^{2m}} \tag{2.2}$$

where we have used the simplified notation $f_0(x,t) = f$.

The time has now come to introduce in detail the surface wave problem, a classical fluid mechanics boundary value problem. In the original dimensional form, it reads

$$\varphi_{xx} + \varphi_{zz} = 0 \text{ on } 0 < z < h_0 + \eta \text{ (the Laplace equation)}$$

$$\varphi_z = 0 \text{ on } z = 0 \text{ (bottom boundary condition)}$$

$$\eta_t + \varphi_x\eta_x - \varphi_z = 0 \text{ on } z = h_0 + \eta \text{ (first free surface boundary condition)}$$

$$\varphi_t + \frac{1}{2}|\nabla\varphi|^2 + gz = gh_0 \text{ on } z = h_0 + \eta \text{ (second free surface boundary condition)}$$

We will not go into the details of how to derive this problem from actual physical considerations, but a good explanation of the derivation for the case of linear waves can be found in e.g. [6], and the problem for non-linear waves is easily found by including non-linear terms in this derivation .

It is also convenient at this point to introduce two dimensionless parameters used by Whitham in [4], these are

$$\alpha = \frac{a}{h_0}$$

and

$$\beta = \frac{h_0^2}{l^2}$$

Note that α denotes the ratio of a typical wave amplitude to the undisturbed water depth, while β denotes a relationship for the ratio of the undisturbed water depth to a typical dominant wave length [1] [4]. We also remember the shallow water approximation for the linear surface wave phase speed, given in [4] and [6] as

$$c_0 = \sqrt{gh_0}$$

with units [m/s].

In the further derivation of the BBM-equation, we need the non-dimensional version of the surface wave problem, so we continue by doing the non-dimensionalizing.

First, we introduce the relationship between the dimensional and non-dimensional variables. They are:

$$x = l\tilde{x}$$

2 - **WAVE BREAKING IN THE BBM-EQUATION**

$$z = h_0\tilde{z}$$

$$t = \frac{l\tilde{t}}{c_0}$$

$$\eta = a\tilde{\eta}$$

$$\varphi = \frac{gla\tilde{\varphi}}{c_0}$$

So $\tilde{x},\ \tilde{z},\ \tilde{t}, \tilde{\eta}\ and\ \tilde{\varphi}$ are now are non-dimensional variables. In the above relations, we have

$$l = characteristic\ length\ scale\ in\ x-direction\ [m]$$

$$h_0 = characteristic\ length\ scale\ in\ z-direction\ [m]$$

$$\frac{gla}{c_0} = characteristic\ velocity\ potential\ \left[\frac{m^2}{s}\right]$$

$$a = characteristic\ surface\ displacement\ (equal\ to\ amplitude)[m]$$

$$\frac{l}{c_0} = characteristic\ time\ scale\ [s]$$

Further, from simple chain rule considerations, we get the following relationships between the dimensional and the non-dimensional differential operators:

$$\frac{\partial}{\partial x} = \frac{1}{l}\frac{\partial}{\partial \tilde{x}}$$

$$\frac{\partial}{\partial z} = \frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}$$

$$\frac{\partial}{\partial t} = \frac{c_0}{l}\frac{\partial}{\partial \tilde{t}}$$

Starting with the Laplace equation, we first rewrite it as

$$\frac{\partial^2}{\partial x^2}\left(\frac{gla\tilde{\varphi}}{c_0}\right) + \frac{\partial^2}{\partial z^2}\left(\frac{gla\tilde{\varphi}}{c_0}\right) = 0$$

and substituting in the non-dimensional differential operators we have

$$\frac{1}{l}\frac{\partial}{\partial \tilde{x}}\left[\frac{1}{l}\frac{\partial}{\partial \tilde{x}}\left(\frac{gla\tilde{\varphi}}{c_0}\right)\right] + \frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}\left[\frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}\left(\frac{gla\tilde{\varphi}}{c_0}\right)\right] = 0$$

We change the notation for derivatives, and rewrite the above as

$$\frac{1}{l^2}\cdot\frac{gla}{c_0}\cdot\tilde{\varphi}_{\tilde{x}\tilde{x}} + \frac{1}{h_0^2}\cdot\frac{gla}{c_0}\cdot\tilde{\varphi}_{\tilde{z}\tilde{z}} = 0$$

Note that the above equation has the units $[1/s]$. We non-dimensionalize it by multiplying it by the factor

$$\frac{c_0 h_0^2}{gla}$$

which has unit [s], and arrive at

$$\frac{h_0^2}{l^2}\tilde{\varphi}_{\tilde{x}\tilde{x}} + \tilde{\varphi}_{\tilde{z}\tilde{z}} = 0$$

And then by $\beta = \frac{h_0^2}{l^2}$, we have the non-dimensional Laplace equation:

$$\beta\tilde{\varphi}_{\tilde{x}\tilde{x}} + \tilde{\varphi}_{\tilde{z}\tilde{z}} = 0 \tag{2.8}$$

For the bottom boundary condition

$$\varphi_z = 0 \text{ on } z = 0$$

we rewrite to

$$\frac{\partial}{\partial z}(\varphi) = 0$$

and substitute for the differential operator and φ to get

$$\frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}\left(\frac{gla\tilde{\varphi}}{c_0}\right) = 0$$

which has unit [m/s]. Now we non-dimensionalize by multiplying the above equation with the factor

$$\frac{h_0 c_0}{gla}$$

with unit [s/m], and we get

$$\tilde{\varphi}_{\tilde{z}} = 0$$

Noting further that from the relationship

$$z = h_0\tilde{z}$$

we have that $z = 0$ implies $\tilde{z} = 0$, so we then have the non-dimensional bottom boundary condition

$$\tilde{\varphi}_{\tilde{z}} = 0 \text{ on } \tilde{z} = 0 \tag{2.9}$$

Continuing with the first free surface boundary condition

$$\eta_t + \varphi_x\eta_x - \varphi_z = 0 \text{ on } z = h_0 + \eta$$

we first rewrite for clarity to

$$\frac{\partial}{\partial t}(\eta) + \frac{\partial}{\partial x}(\varphi)\cdot\frac{\partial}{\partial x}(\eta) - \frac{\partial}{\partial z}(\varphi) = 0$$

And substitute for differential operators and variables to get

$$\frac{c_0}{l}\frac{\partial}{\partial \tilde{t}}(a\tilde{\eta}) + \frac{1}{l}\frac{\partial}{\partial \tilde{x}}\left(\frac{gla\tilde{\varphi}}{c_0}\right)\cdot\frac{1}{l}\frac{\partial}{\partial \tilde{x}}(a\tilde{\eta}) - \frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}\left(\frac{gla\tilde{\varphi}}{c_0}\right) = 0$$

Changing back the notation for derivatives, and noting and using that

$$\alpha = \frac{a}{h_0} \text{ gives } a = \alpha h_0$$

we have

$$\frac{\alpha h_0 c_0}{l}\cdot\tilde{\eta}_{\tilde{t}} + \frac{1}{l^2}\cdot\frac{gl\alpha^2 h_0^2}{c_0}\cdot\tilde{\varphi}_{\tilde{x}}\cdot\tilde{\eta}_{\tilde{x}} - \frac{gl\alpha}{c_0}\cdot\tilde{\varphi}_{\tilde{z}} = 0$$

which has the unit [m/s]. Now we non-dimensionalize by multiplying the above equation by the term

$$\frac{l}{\alpha h_0 c_0}$$

with unit [s/m], to get

$$\tilde{\eta}_{\tilde{t}} + \frac{g\alpha h_0}{c_0^2}\cdot\tilde{\varphi}_{\tilde{x}}\tilde{\eta}_{\tilde{x}} - \frac{gl^2}{h_0 c_0^2}\cdot\tilde{\varphi}_{\tilde{z}} = 0$$

and then by substituting for c_0 and β by using

$$c_0 = \sqrt{gh_0} \text{ and } \beta = \frac{h_0^2}{l^2}$$

we arrive at

$$\tilde{\eta}_{\tilde{t}} + \alpha\tilde{\varphi}_{\tilde{x}}\tilde{\eta}_{\tilde{x}} - \frac{1}{\beta}\tilde{\varphi}_{\tilde{z}} = 0$$

We must also find the non-dimensional expression for the z-value at the free surface. Remembering that in dimensional form, it reads

$$z = h_0 + \eta$$

We use the relationship $z = h_0\tilde{z}$ and $\eta = a\tilde{\eta}$ to get

$$h_0\tilde{z} = h_0 + a\tilde{\eta}$$

then dividing this equation by h_0 and using $\alpha = \frac{a}{h_0}$, it turns into

$$\tilde{z} = 1 + \alpha\tilde{\eta}$$

And so we now have the first free surface boundary condition in the non-dimensional version:

$$\tilde{\eta}_{\tilde{t}} + \alpha\tilde{\varphi}_{\tilde{x}}\tilde{\eta}_{\tilde{x}} - \frac{1}{\beta}\tilde{\varphi}_{\tilde{z}} = 0 \text{ on } \tilde{z} = 1 + \alpha\tilde{\eta} \tag{2.10}$$

Turning now to the second free surface boundary condition

$$\varphi_t + \frac{1}{2}|\nabla\varphi|^2 + gz = gh_0 \text{ on } z = h_0 + \eta$$

2 - **WAVE BREAKING IN THE BBM-EQUATION**

We rewrite the equation, expanding the gradient term, evaluating at $z = h_0 + \eta$ and changing notation for derivatives to get

$$\frac{\partial}{\partial t}(\varphi) + \frac{1}{2} \cdot \left(\frac{\partial}{\partial x}(\varphi)\right)^2 + \frac{1}{2} \cdot \left(\frac{\partial}{\partial z}(\varphi)\right)^2 + g\eta = 0$$

Now, substituting for differential operators and variables, we have

$$\frac{c_0}{l}\frac{\partial}{\partial \tilde{t}}\left(\frac{gla\tilde{\varphi}}{c_0}\right) + \frac{1}{2} \cdot \left(\frac{1}{l}\frac{\partial}{\partial \tilde{x}}\left(\frac{gla\tilde{\varphi}}{c_0}\right)\right)^2 + \frac{1}{2} \cdot \left(\frac{1}{h_0}\frac{\partial}{\partial \tilde{z}}\left(\frac{gla\tilde{\varphi}}{c_0}\right)\right)^2 + ga\tilde{\eta} = 0$$

which by simplifying, substituting in $c_0 = \sqrt{gh_0}$ and changing the derivative notation back turns into

$$ga\tilde{\varphi}_{\tilde{t}} + \frac{1}{2} \cdot \frac{ga^2}{h_0} \cdot \tilde{\varphi}_{\tilde{x}}^2 + \frac{1}{2} \cdot \frac{gl^2a^2}{h_0^3} \cdot \tilde{\varphi}_{\tilde{z}}^2 + ga\tilde{\eta} = 0$$

an equation having the unit $[m^2/s^2]$, so we nondimensionalize by multiplying the equation with the term

$$\frac{1}{ga}$$

which has the unit $[s^2/m^2]$, and we then have

$$\tilde{\varphi}_{\tilde{t}} + \frac{1}{2} \cdot \frac{a}{h_0} \cdot \tilde{\varphi}_{\tilde{x}}^2 + \frac{1}{2} \cdot \frac{l^2}{h_0^2} \cdot \frac{a}{h_0} \cdot \tilde{\varphi}_{\tilde{z}}^2 + \tilde{\eta} = 0$$

And finally, by expressing the equation above in terms of the parameters α and β, we find the second free surface boundary condition in the non-dimensional version to be

$$\tilde{\eta} + \tilde{\varphi}_{\tilde{t}} + \tfrac{1}{2}\alpha\tilde{\varphi}_{\tilde{x}}^2 + \tfrac{1}{2}\tfrac{\alpha}{\beta}\tilde{\varphi}_{\tilde{z}}^2 = 0 \text{ on } \tilde{z} = 1 + \alpha\tilde{\eta} \tag{2.11}$$

For the sake of clarity, let us now state the full non-dimensional version of the surface wave problem:

$$\beta\tilde{\varphi}_{\tilde{x}\tilde{x}} + \tilde{\varphi}_{\tilde{z}\tilde{z}} = 0 \text{ on } 0 < \tilde{z} < 1 + \alpha\tilde{\eta} \text{ (non-dim. Laplace equation)}$$

$$\tilde{\varphi}_{\tilde{z}} = 0 \text{ on } \tilde{z} = 0 \text{ (non-dim. bottom boundary condition)}$$

$$\tilde{\eta}_{\tilde{t}} + \alpha\tilde{\varphi}_{\tilde{x}}\tilde{\eta}_{\tilde{x}} - \frac{1}{\beta}\tilde{\varphi}_{\tilde{z}} = 0 \text{ on } \tilde{z} = 1 + \alpha\tilde{\eta} \text{ (first non-dim. free surface boundary condition)}$$

$$\tilde{\eta} + \tilde{\varphi}_{\tilde{t}} + \tfrac{1}{2}\alpha\tilde{\varphi}_{\tilde{x}}^2 + \tfrac{1}{2}\tfrac{\alpha}{\beta}\tilde{\varphi}_{\tilde{z}}^2 = 0 \text{ on } \tilde{z} = 1 + \alpha\tilde{\eta} \text{ (second non-dim. free surface boundary condition)}$$

Now, we need the non-dimensional version of the expansion (2.2). There are at least two ways to get this.

The first method is described by Kalisch and Bjørkavåg in [1], and it starts with assuming that the non-dimensional velocity potential $\tilde{\varphi}(\tilde{x}, \tilde{z}, \tilde{t})$ may be written as an asymptotic expansion in the "parameter" $\tilde{z}$, that is

$$\tilde{\varphi} = \sum_{n=0}^{\infty} \tilde{z}^n \tilde{f}_n(\tilde{x}, \tilde{t}) \tag{2.12}$$

Note that this is the non-dimensional version of (2.1). Then (following exactly the same procedure as we did earlier in this chapter for (2.1), substituting (2.12) into the non-dimensional Laplace equation (2.8) and applying the non-dimensional bottom boundary condition (2.9), the non-dimensional version of (2.2) is obtained.

The second method is to just start with (2.2) and non-dimensionalize it. Stating again (2.2) for clarity

$$\varphi = \sum_{m=0}^{\infty} (-1)^m \cdot \frac{z^{2m}}{(2m)!} \cdot \frac{\partial^{2m} f}{\partial x^{2m}} \tag{2.13}$$

where $f_0(x,t) = f$. Writing out the first few terms of the above expansion

$$\varphi = z^0 f - \frac{z^2}{2} f_{xx} + \frac{z^4}{24} f_{xxxx} - \frac{z^6}{720} f_{xxxxxx} + \cdots \tag{2.14}$$

and noting that the dimensional velocity potential φ has the unit $[m^2/s]$, we observe that for (2.14) to be homogenous w.r.t units, the function $f_0(x,t) = f$ also must have the unit $[m^2/s]$, as z^0 just has the unit [1]. Knowing this, we introduce the non-dimensional function $\tilde{f} = \tilde{f}_0(\tilde{x}, \tilde{t})$ through the relationship

$$f = \frac{gla\tilde{f}}{c_0}$$

and we are now ready to non-dimensionalize (2.2). As usual, substituting for variables and differential operators, we get

$$\frac{gla\tilde{\varphi}}{c_0} = \sum_{m=0}^{\infty} (-1)^m \cdot \frac{(h_0\tilde{z})^{2m}}{(2m)!} \cdot \frac{1}{l^{2m}} \cdot \frac{\partial^{2m}}{\partial \tilde{x}^{2m}} \left(\frac{gla\tilde{f}}{c_0} \right)$$

This can be rewritten to

$$\frac{gla\tilde{\varphi}}{c_0} = \frac{gla}{c_0} \cdot \sum_{m=0}^{\infty} (-1)^m \cdot \frac{\tilde{z}^{2m}}{(2m)!} \cdot \frac{(h_0^2)^m}{(l^2)^m} \cdot \frac{\partial^{2m} \tilde{f}}{\partial \tilde{x}^{2m}}$$

which is non-dimensionalized by multiplying by the factor

$$\frac{c_0}{gla}$$

and then after recognizing the parameter β, we find the non-dimensionalized version of (2.2) to be:

$$\tilde{\varphi} = \sum_{m=0}^{\infty} (-1)^m \cdot \frac{\tilde{z}^{2m}}{(2m)!} \cdot \frac{\partial^{2m} \tilde{f}}{\partial \tilde{x}^{2m}} \cdot \beta^m \tag{2.15}$$

The derivation continues by substituting (2.15) and its derivatives into the free surface boundary conditions in the non-dimensional Laplace equation (2.8). We write out the first terms of the series (2.15) as well as its derivatives:

$$\tilde{\varphi} = \tilde{f} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}} \cdot \beta + \frac{\tilde{z}^4}{24} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} \cdot \beta^2 + O(\beta^3) \tag{2.16}$$

$$\tilde{\varphi}_{\tilde{x}} = \tilde{f}_{\tilde{x}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \cdot \beta + \frac{\tilde{z}^4}{24} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}\tilde{x}} \cdot \beta^2 + O(\beta^3) \tag{2.17}$$

$$\tilde{\varphi}_{\tilde{z}} = -\tilde{z} \cdot \tilde{f}_{\tilde{x}\tilde{x}} \cdot \beta + \frac{1}{6} \cdot \tilde{z}^3 \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} \cdot \beta^2 + O(\beta^3) \tag{2.18}$$

$$\tilde{\varphi}_{\tilde{t}} = \tilde{f}_{\tilde{t}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} \cdot \beta + \frac{\tilde{z}^4}{24} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}\tilde{t}} \cdot \beta^2 + O(\beta^3) \tag{2.19}$$

Substituting for $\tilde{\varphi}_{\tilde{x}}$ and $\tilde{\varphi}_{\tilde{z}}$ in the first free surface boundary condition (2.10) of (2.8), we get

$$\tilde{\eta}_{\tilde{t}} + \alpha \left[\tilde{f}_{\tilde{x}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \cdot \beta + O(\beta^2) \right] \tilde{\eta}_{\tilde{x}} - \frac{1}{\beta} \left[-\tilde{z} \cdot \tilde{f}_{\tilde{x}\tilde{x}} \cdot \beta + \frac{1}{6} \cdot \tilde{z}^3 \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} \cdot \beta^2 + O(\beta^3) \right] = 0$$

which after a little rearrangement and evaluating at $\tilde{z} = 1 + \alpha\tilde{\eta}$ turns into

$$\tilde{\eta}_{\tilde{t}} + \alpha \tilde{f}_{\tilde{x}} \tilde{\eta}_{\tilde{x}} + (1 + \alpha\tilde{\eta}) \tilde{f}_{\tilde{x}\tilde{x}} - \left[\frac{1}{6} (1 + \alpha\tilde{\eta})^3 \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + \frac{1}{2} \alpha (1 + \alpha\tilde{\eta})^2 \cdot \tilde{\eta}_{\tilde{x}} \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \right] \beta + O(\beta^2) = 0$$

The second and third terms can be rewritten as

$$\left[(1 + \alpha\tilde{\eta}) \tilde{f}_{\tilde{x}} \right]_{\tilde{x}} = \alpha \tilde{f}_{\tilde{x}} \tilde{\eta}_{\tilde{x}} + (1 + \alpha\tilde{\eta}) \tilde{f}_{\tilde{x}\tilde{x}}$$

and taking this into the equation, we have

$$\begin{aligned} \tilde{\eta}_{\tilde{t}} + \left[(1 + \alpha\tilde{\eta}) \tilde{f}_{\tilde{x}} \right]_{\tilde{x}} - \left[\frac{1}{6} (1 + \alpha\tilde{\eta})^3 \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + \frac{1}{2} \alpha (1 + \alpha\tilde{\eta})^2 \cdot \tilde{\eta}_{\tilde{x}} \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \right] \beta \\ + O(\beta^2) = 0 \end{aligned} \tag{2.20}$$

Substituting for $\tilde{\varphi}_{\tilde{x}}$, $\tilde{\varphi}_{\tilde{z}}$ and $\tilde{\varphi}_{\tilde{t}}$ in the second free surface boundary condition (2.11) of (2.8), we get

$$\begin{aligned} \tilde{\eta} + \left[\tilde{f}_{\tilde{t}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} \cdot \beta + O(\beta^2) \right] + \frac{1}{2} \alpha \left[\tilde{f}_{\tilde{x}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \cdot \beta + O(\beta^2) \right]^2 \\ + \frac{1}{2} \frac{\alpha}{\beta} \left[-\tilde{z} \cdot \tilde{f}_{\tilde{x}\tilde{x}} \cdot \beta + O(\beta^2) \right]^2 = 0 \end{aligned}$$

which after multiplying out the bracket parenthesis and ordering terms in powers of β turns into

$$\tilde{\eta} + \tilde{f}_{\tilde{t}} + \frac{1}{2} \alpha \tilde{f}_{\tilde{x}}^2 - \frac{1}{2} \tilde{z}^2 \cdot \left[\tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} + \alpha \tilde{f}_{\tilde{x}} \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} - \alpha \tilde{f}_{\tilde{x}\tilde{x}}^2 \right] \beta + O(\beta^2) = 0$$

And by evaluating at $\tilde{z} = 1 + \alpha\tilde{\eta}$ we then have

$$\tilde{\eta} + \tilde{f}_{\tilde{t}} + \frac{1}{2}\alpha\tilde{f}_{\tilde{x}}^2 - \frac{1}{2}(1+\alpha\tilde{\eta})^2 \cdot \left[\tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} + \alpha\tilde{f}_{\tilde{x}}\tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} - \alpha\tilde{f}_{\tilde{x}\tilde{x}}^2\right]\beta + O(\beta^2) = 0 \qquad (2.21)$$

Equations (2.20) and (2.21) can be manipulated into several new equations. As a little side-step: As stated by Whitham in [4], by dropping all terms of $O(\beta)$ and differentiating (2.21) with respect to $\tilde{x}$, we get a non-dimensional version of the shallow water equations:

$$\tilde{\eta}_{\tilde{t}} + [(1+\alpha\tilde{\eta})\tilde{w}]_{\tilde{x}} = 0$$

$$\tilde{w}_{\tilde{t}} + \alpha\tilde{w}\tilde{w}_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} = 0$$

We continue our progress towards the BBM-equation by keeping terms "purely" in $O(\beta)$, but dropping all term in $O(\alpha\beta)$ in (2.20) and (2.21).

For (2.20) we get by first using $\tilde{w} = \tilde{f}_{\tilde{x}}$:

$$\tilde{\eta}_{\tilde{t}} + [(1+\alpha\tilde{\eta})\tilde{w}]_{\tilde{x}} - \left[\frac{1}{6}(1+\alpha\tilde{\eta})^3 \cdot \tilde{w}_{\tilde{x}\tilde{x}\tilde{x}} + \frac{1}{2}\alpha(1+\alpha\tilde{\eta})^2 \cdot \tilde{\eta}_{\tilde{x}}\tilde{w}_{\tilde{x}\tilde{x}}\right]\beta + O(\beta^2) = 0$$

Then by expanding the two parentheses inside the square brackets, we get

$$\begin{aligned}\tilde{\eta}_{\tilde{t}} + [(1+\alpha\tilde{\eta})\tilde{w}]_{\tilde{x}} \\ - \left[\frac{1}{6}(1+3\alpha\tilde{\eta}+3\alpha^2\tilde{\eta}^2+\alpha^3\tilde{\eta}^3) \cdot \tilde{w}_{\tilde{x}\tilde{x}\tilde{x}} + \frac{1}{2}\alpha(1+2\alpha\tilde{\eta}+\alpha^2\tilde{\eta}^2) \cdot \tilde{\eta}_{\tilde{x}}\tilde{w}_{\tilde{x}\tilde{x}}\right]\beta \\ + O(\beta^2) = 0\end{aligned}$$

and then by dropping all terms in $O(\alpha\beta)$ we get

$$\tilde{\eta}_{\tilde{t}} + [(1+\alpha\tilde{\eta})\tilde{w}]_{\tilde{x}} - \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) = 0 \qquad (2.22)$$

Turning to (2.21), we multiply out the first parenthesis to get

$$\tilde{\eta} + \tilde{f}_{\tilde{t}} + \frac{1}{2}\alpha\tilde{f}_{\tilde{x}}^2 - \frac{1}{2}(1+2\alpha\tilde{\eta}+\alpha^2\tilde{\eta}^2) \cdot \left[\tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} + \alpha\tilde{f}_{\tilde{x}}\tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} - \alpha\tilde{f}_{\tilde{x}\tilde{x}}^2\right]\beta + O(\beta^2) = 0$$

and we drop all terms in $O(\beta^2)$ and $O(\alpha\beta)$ to have

$$\tilde{\eta} + \tilde{f}_{\tilde{t}} + \frac{1}{2}\alpha\tilde{f}_{\tilde{x}}^2 - \frac{1}{2}\beta\tilde{f}_{\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0$$

Differentiating the equation w.r.t. $\tilde{x}$ yields

$$\tilde{\eta}_{\tilde{x}} + \tilde{f}_{\tilde{x}\tilde{t}} + \alpha\tilde{f}_{\tilde{x}}\tilde{f}_{\tilde{x}\tilde{x}} - \frac{1}{2}\beta\tilde{f}_{\tilde{x}\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0$$

and finally by doing the substitution $\tilde{w} = \tilde{f}_{\tilde{x}}$, we have

$$\tilde{w}_{\tilde{t}} + \alpha\tilde{w}\tilde{w}_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} - \frac{1}{2}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0 \qquad (2.23)$$

Equations number (2.22) and (2.23) are, according to Whitham in [4], a variant of the Boussinesq equations.

Whitham further notes in [4] that $\tilde{w}$ is the first term in the expansion of the non-dimensional horizontal fluid velocity $\tilde{\varphi}_{\tilde{x}} = \tilde{u}$, which is easily verified by using $\tilde{w} = \tilde{f}_{\tilde{x}}$ in equation (2.17) to find

$$\tilde{\varphi}_{\tilde{x}} = \tilde{u} = \tilde{w} - \beta \frac{\tilde{z}^2}{2} \cdot \tilde{w}_{\tilde{x}\tilde{x}} + O(\beta^2)$$

It is easily observed from this equation that the non-dimensional horizontal fluid velocity depends on the elevation $\tilde{z}$. Now we want to continue by finding the averaged value of $\tilde{u}$ over the depth, so we set up an integral based on the following figure:

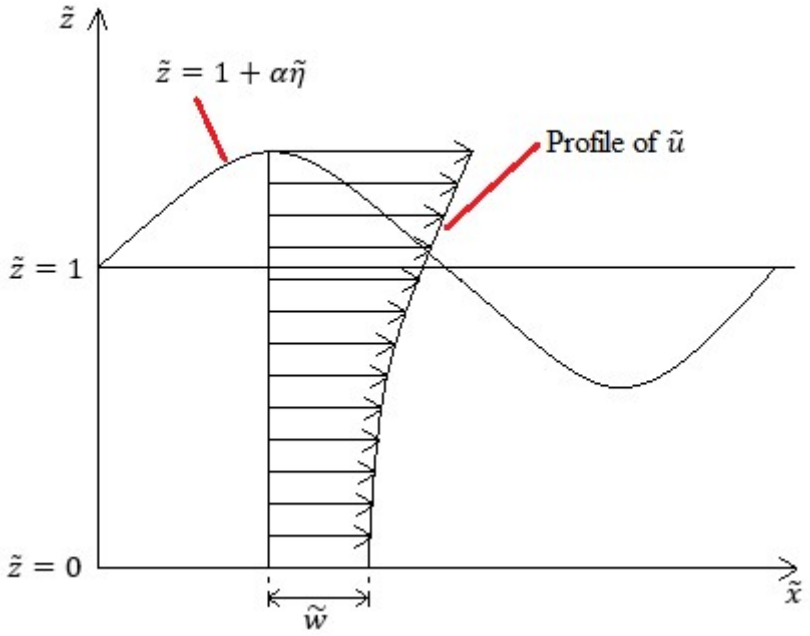

Figure 2.1 – Non-dimensional horizontal fluid velocity profile

The integral will be

$$\bar{\tilde{u}} = \frac{1}{1 + \alpha\tilde{\eta}} \int_{\tilde{z}=0}^{\tilde{z}=1+\alpha\tilde{\eta}} \tilde{u}\, d\tilde{z}$$

which when substituting for $\tilde{u}$ reads

$$\bar{\tilde{u}} = \frac{1}{1 + \alpha\tilde{\eta}} \int_{\tilde{z}=0}^{\tilde{z}=1+\alpha\tilde{\eta}} \tilde{w} - \beta \frac{\tilde{z}^2}{2} \cdot \tilde{w}_{\tilde{x}\tilde{x}} + O(\beta^2)\, d\tilde{z}$$

After doing the integration, we have

$$\bar{\tilde{u}} = \frac{1}{1 + \alpha\tilde{\eta}} \left[\tilde{w}\tilde{z} - \frac{1}{6}\beta \tilde{w}_{\tilde{x}\tilde{x}} \tilde{z}^3 + O(\beta^2) \right]_{\tilde{z}=0}^{\tilde{z}=1+\alpha\tilde{\eta}}$$

which after doing the evaluation of the bracket parenthesis reads

$$\bar{\tilde{u}} = \frac{1}{1 + \alpha\tilde{\eta}} \left[(1 + \alpha\tilde{\eta})\tilde{w} - \frac{1}{6}\beta \tilde{w}_{\tilde{x}\tilde{x}} \cdot (1 + \alpha\tilde{\eta}) \cdot (1 + 2\alpha\tilde{\eta} + \alpha^2\tilde{\eta}^2) + O(\beta^2) \right]$$

and the by dropping terms in $O(\alpha\beta)$ we find the averaged non-dimensional horizontal fluid velocity as

$$\bar{\tilde{u}} = \tilde{w} - \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) \tag{2.24}$$

Now we note that we have an expression for $\bar{\tilde{u}} = \bar{\tilde{u}}(\tilde{w})$, but we also need the inverse $\tilde{w} = \tilde{w}(\bar{\tilde{u}})$. Rearranging (2.24), we have

$$\tilde{w} = \bar{\tilde{u}} + \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) \tag{2.25}$$

Differentiating (2.24) twice w.r.t. $\tilde{x}$, we get

$$\bar{\tilde{u}}_{\tilde{x}\tilde{x}} = \tilde{w}_{\tilde{x}\tilde{x}} - \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2)$$

which we use to express $\tilde{w}_{\tilde{x}\tilde{x}}$ as

$$\tilde{w}_{\tilde{x}\tilde{x}} = \bar{\tilde{u}}_{\tilde{x}\tilde{x}} + \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2)$$

Which we substitute into (2.25) to find

$$\tilde{w} = \bar{\tilde{u}} + \frac{1}{6}\beta\left(\bar{\tilde{u}}_{\tilde{x}\tilde{x}} + \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2)\right) + O(\alpha\beta, \beta^2)$$

and then by dropping all terms in $O(\alpha\beta)$ and $O(\beta^2)$ we arrive at the inverse of (2.24):

$$\tilde{w} = \bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) \tag{2.26}$$

Equations (2.22) and (2.23) are central in the following, so we write them up again together

$$\tilde{\eta}_{\tilde{t}} + [(1 + \alpha\tilde{\eta})\tilde{w}]_{\tilde{x}} - \frac{1}{6}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) = 0 \tag{2.22}$$

$$\tilde{w}_{\tilde{t}} + \alpha\tilde{w}\tilde{w}_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} - \frac{1}{2}\beta\tilde{w}_{\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0 \tag{2.23}$$

As a little detour, substituting (2.26) for $\tilde{w}$ into (2.22) and (2.23), turning the equations into

$$\tilde{\eta}_{\tilde{t}} + \left[(1 + \alpha\tilde{\eta}) \cdot \left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right)\right]_{\tilde{x}} - \frac{1}{6}\beta\left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right)_{\tilde{x}\tilde{x}\tilde{x}} + O(\alpha\beta, \beta^2) = 0$$

$$\left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right)_{\tilde{t}} + \alpha\left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right) \cdot \left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right)_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} - \frac{1}{2}\beta\left(\bar{\tilde{u}} + \frac{1}{6}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}}\right)_{\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0$$

and dropping all terms in $O(\alpha\beta)$ and $O(\beta^2)$ to get

$$\tilde{\eta}_{\tilde{t}} + [(1 + \alpha\tilde{\eta}) \cdot \bar{\tilde{u}}]_{\tilde{x}} + O(\alpha\beta, \beta^2) = 0$$

$$\bar{\tilde{u}}_{\tilde{t}} + \alpha\bar{\tilde{u}}\bar{\tilde{u}}_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} - \frac{1}{3}\beta\bar{\tilde{u}}_{\tilde{x}\tilde{x}\tilde{t}} + O(\alpha\beta, \beta^2) = 0$$

we can use the first of these two equations, rearrange it and drop all terms of $O(\alpha, \beta)$ to find $\bar{\tilde{u}}_{\tilde{x}} = -\tilde{\eta}_{\tilde{t}} + O(\alpha, \beta)$, which is then substituted into the $\bar{\tilde{u}}_{\tilde{x}\tilde{x}\tilde{t}}$-term of the second equation, giving us now the set of equations

$$\tilde{\eta}_{\tilde{t}} + [(1 + \alpha\tilde{\eta}) \cdot \bar{\tilde{u}}]_{\tilde{x}} + O(\alpha\beta, \beta^2) = 0$$

$$\bar{\tilde{u}}_{\tilde{t}} + \alpha\bar{\tilde{u}}\bar{\tilde{u}}_{\tilde{x}} + \tilde{\eta}_{\tilde{x}} + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{t}\tilde{t}} + O(\alpha\beta, \beta^2) = 0$$

which is the non-dimensional version of the Boussinesq equations, introduced by Whitham in [4], p. 462.

However, in our search for the BBM-equation, we turn again to equations (2.22) and (2.23), and follow Whitham's statement that the KdV-equation is derived from (for example) this set of equations, by specializing to a right-travelling wave.

Neglecting terms of $O(\alpha)$ and $O(\beta)$ in (2.22) and (2.23), we get a system of PDEs consisting of the zeroth-order (with respect to parameters α and β) equations

$$\tilde{\eta}_{\tilde{t}} + \widetilde{w}_{\tilde{x}} = 0 \tag{2.27}$$

$$\widetilde{w}_{\tilde{t}} + \tilde{\eta}_{\tilde{x}} = 0 \tag{2.28}$$

Differentiating the first equation w.r.t. $\tilde{t}$ and the second equation w.r.t. $\tilde{x}$ gives us

$$\tilde{\eta}_{\tilde{t}\tilde{t}} + \widetilde{w}_{\tilde{x}\tilde{t}} = 0$$

$$\widetilde{w}_{\tilde{x}\tilde{t}} + \tilde{\eta}_{\tilde{x}\tilde{x}} = 0$$

and then by subtracting the second one of these from the first one, we have the homogenous wave equation [7] for $\tilde{\eta}$. Observe that in equations (2.27) and (2.28) we can also differentiate the first equation w.r.t. $\tilde{x}$ and the second one w.r.t. $\tilde{t}$, and then from subtracting the resulting first equation from the second one, we will get the homogenous wave equation for $\widetilde{w}$. So we then have the wave equations

$$\tilde{\eta}_{\tilde{t}\tilde{t}} - \tilde{\eta}_{\tilde{x}\tilde{x}} = 0$$

$$\widetilde{w}_{\tilde{t}\tilde{t}} - \widetilde{w}_{\tilde{x}\tilde{x}} = 0$$

The general solutions to these equations are [7] respectively

$$\tilde{\eta}(\tilde{x}, \tilde{t}) = F(\tilde{x} + \tilde{t}) + G(\tilde{x} - \tilde{t})$$

$$\widetilde{w}(\tilde{x}, \tilde{t}) = H(\tilde{x} + \tilde{t}) + I(\tilde{x} - \tilde{t})$$

where functions F and H represent a left-travelling wave, while the functions G and I represent a right-travelling wave. At this point, we want to specialize to right-travelling waves, so we drop the F- and H-functions from the solutions for $\tilde{\eta}$ and $\widetilde{w}$, so that we now have

$$\tilde{\eta}(\tilde{x}, \tilde{t}) = G(\tilde{x} - \tilde{t})$$

$$\widetilde{w}(\tilde{x}, \tilde{t}) = I(\tilde{x} - \tilde{t})$$

Now, substituting these expressions for $\tilde{\eta}$ and $\widetilde{w}$ in equation (2.27) and (2.28) yields

$$[G(\tilde{x}-\tilde{t})]_{\tilde{t}}+[I(\tilde{x}-\tilde{t})]_{\tilde{x}}=0$$

$$[I(\tilde{x}-\tilde{t})]_{\tilde{t}}+[G(\tilde{x}-\tilde{t})]_{\tilde{x}}=0$$

Performing the derivations according to the chain rule will turn this into

$$G'^{\tilde{t}}(\tilde{x}-\tilde{t})\cdot(-1)+I'^{\tilde{x}}(\tilde{x}-\tilde{t})\cdot 1=0$$

$$I'^{\tilde{t}}(\tilde{x}-\tilde{t})\cdot(-1)+G'^{\tilde{x}}(\tilde{x}-\tilde{t})\cdot 1=0$$

and since the derivatives of the *outer functions*, e.g. $G'^{\tilde{t}}(\tilde{x}-\tilde{t})$ and $I'^{\tilde{x}}(\tilde{x}-\tilde{t})$ does not depend on whether we differentiate w.r.t $\tilde{x}$ or $\tilde{t}$, we observe that the only way that the last set of equations above will hold is if

$$G(\tilde{x}-\tilde{t})=\ I(\tilde{x}-\tilde{t})=J(\tilde{x}-\tilde{t})$$

So then we have

$$\widetilde{w}=\ \tilde{\eta} \tag{2.29}$$

and then by substituting this for $\widetilde{w}$ into either of the equations (2.27) and (2.28), it follows that

$$\tilde{\eta}_{\tilde{t}}+\tilde{\eta}_{\tilde{x}}=0 \tag{2.30}$$

Equation (2.29) is the zeroth order solution for $\widetilde{w}$ as a function of $\tilde{\eta}$ with regards to the parameters α and β. Now we want to look for a solution that is corrected to the first order of α and β, on the form

$$\widetilde{w}=\ \tilde{\eta}+\ \alpha A+\ \beta B+O(\alpha^2+\beta^2) \tag{2.31}$$

the new variables A and B being functions of $\tilde{\eta}$ and $\tilde{x}$-derivatives of $\tilde{\eta}$. Substituting the proposed form of the solution for $\widetilde{w}$ (2.31) into equations (2.22) and (2.23) we have

$$\tilde{\eta}_{\tilde{t}}+[(1+\alpha\tilde{\eta})\cdot(\tilde{\eta}+\ \alpha A+\ \beta B)]_{\tilde{x}}-\frac{1}{6}\beta(\tilde{\eta}+\ \alpha A+\ \beta B)_{\tilde{x}\tilde{x}\tilde{x}}+O(\alpha\beta,\beta^2)=0$$

$$(\tilde{\eta}+\ \alpha A+\ \beta B)_{\tilde{t}}+\alpha(\tilde{\eta}+\ \alpha A+\ \beta B)\cdot(\tilde{\eta}+\ \alpha A+\ \beta B)_{\tilde{x}}+\tilde{\eta}_{\tilde{x}}-\frac{1}{2}\beta(\tilde{\eta}+\ \alpha A+\ \beta B)_{\tilde{x}\tilde{x}\tilde{t}}+O(\alpha\beta,\beta^2)=0$$

which after some sorting out of terms turn into

$$\tilde{\eta}_{\tilde{t}}+\tilde{\eta}_{\tilde{x}}+\alpha(A_{\tilde{x}}+2\tilde{\eta}\tilde{\eta}_{\tilde{x}})+\beta\left(B_{\tilde{x}}-\frac{1}{6}\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}}\right)+O(\alpha^2+\beta^2)=0 \tag{2.32}$$

$$\tilde{\eta}_{\tilde{t}}+\tilde{\eta}_{\tilde{x}}+\alpha(A_{\tilde{t}}+\tilde{\eta}\tilde{\eta}_{\tilde{x}})+\beta\left(B_{\tilde{t}}-\frac{1}{2}\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{t}}\right)+O(\alpha^2+\beta^2)=0 \tag{2.33}$$

From equation (2.30) or any of the two above equations, it follows that $\tilde{\eta}_{\tilde{t}}=-\tilde{\eta}_{\tilde{x}}+O(\alpha,\beta)$, and from this, we can replace all $\tilde{t}$-derivatives in the terms in first order of α and β (and these terms only!) in the above equation (2.33) by minus the $\tilde{x}$-derivatives. To give an illustration, we will have for example

$$\beta B_{\tilde{t}}=\beta\cdot\left(-B_{\tilde{x}}+O(\alpha,\beta)\right)$$

$$\beta B_{\tilde{t}} = -\beta B_{\tilde{x}} + O(\alpha\beta, \beta^2)$$

where the second term on the RHS in the latter equation will just be thrown into the "$O(\alpha^2 + \beta^2)$-sack" as we proceed. After doing the derivative replacement, we have

$$\tilde{\eta}_{\tilde{t}} + \tilde{\eta}_{\tilde{x}} + \alpha(A_{\tilde{x}} + 2\tilde{\eta}\tilde{\eta}_{\tilde{x}}) + \beta\left(B_{\tilde{x}} - \frac{1}{6}\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}}\right) + O(\alpha^2 + \beta^2) = 0 \tag{2.32}$$

$$\tilde{\eta}_{\tilde{t}} + \tilde{\eta}_{\tilde{x}} + \alpha(-A_{\tilde{x}} + \tilde{\eta}\tilde{\eta}_{\tilde{x}}) + \beta\left(-B_{\tilde{x}} + \frac{1}{2}\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}}\right) + O(\alpha^2 + \beta^2) = 0 \tag{2.34}$$

It can easily be verified (by substituting for A and B into the equations) that the two equations (2.32) and (2.34) above are consistent (equal) if

$$A = -\tfrac{1}{4}\tilde{\eta}^2 \text{ and } B = \tfrac{1}{3}\tilde{\eta}_{\tilde{x}\tilde{x}}$$

which gives us the two equations

$$\tilde{w} = \tilde{\eta} - \frac{1}{4}\alpha\tilde{\eta}^2 + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}} + O(\alpha^2 + \beta^2) \tag{2.35}$$

$$\tilde{\eta}_{\tilde{t}} + \tilde{\eta}_{\tilde{x}} + \frac{3}{2}\alpha\tilde{\eta}\tilde{\eta}_{\tilde{x}} + \frac{1}{6}\beta\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}} + O(\alpha^2 + \beta^2) = 0 \tag{2.36}$$

The first equation (2.35) is similar to a Riemann invariant, according to Whitham in [4]. It is also the non-dimensional horizontal fluid velocity at the bottom ($\tilde{z} = 0$) for both the KdV-equation and the BBM-equation. The second equation (2.36) is the non-dimensional version of the KdV-equation given by Whitham in [4], p. 463.

To be thorough, we will do the steps of the re-dimensionalizing of the KdV-equation here, using the same relationships between dimensional and non-dimensional variables and differential operators as we have done earlier. Obviously, this is just doing the steps of the non-dimensionalizing backwards. Starting out with the non-dimensional KdV-equation

$$\tilde{\eta}_{\tilde{t}} + \tilde{\eta}_{\tilde{x}} + \frac{3}{2}\alpha\tilde{\eta}\tilde{\eta}_{\tilde{x}} + \frac{1}{6}\beta\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}} + O(\alpha^2 + \beta^2) = 0 \tag{2.36}$$

we divide the equation by the term

$$\frac{l}{c_0\alpha h_0}$$

which has the unit [s/m], thus getting a new equation

$$\frac{c_0\alpha h_0}{l}\tilde{\eta}_{\tilde{t}} + \frac{c_0\alpha h_0}{l}\tilde{\eta}_{\tilde{x}} + \frac{3c_0\alpha^2 h_0}{2l}\tilde{\eta}\tilde{\eta}_{\tilde{x}} + \frac{c_0\alpha h_0}{6l}\beta\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}} = 0$$

having the unit [m/s], which is correct. Now by some rearranging and using

$$\alpha = \frac{a}{h_0} \text{ and } \beta = \frac{h_0^2}{l^2}$$

we arrive at

$$\frac{c_0}{l}\cdot(a\tilde{\eta})_{\tilde{t}}+c_0\cdot\left(1+\frac{3}{2h_0}\cdot a\tilde{\eta}\right)\cdot\frac{1}{l}\cdot(a\tilde{\eta})_{\tilde{x}}+\gamma\cdot\frac{1}{l^3}\cdot(a\tilde{\eta})_{\tilde{x}\tilde{x}\tilde{x}}=0$$

and by substituting dimensional variables and differential operators for the non-dimensional ones, we finally get the KdV-equation as given by Whitham in [4]:

$$\eta_t+c_0\left(1+\frac{3}{2}\frac{\eta}{h_0}\right)\eta_x+\gamma\eta_{xxx}=0 \tag{2.37}$$

Now, we recall the relation $\tilde{\eta}_{\tilde{t}}=-\tilde{\eta}_{\tilde{x}}+O(\alpha,\beta)$ which we employed earlier, we can rewrite this as $\tilde{\eta}_{\tilde{t}}\approx-\tilde{\eta}_{\tilde{x}}$. Utilizing our dimensional/non-dimensional relationships, we can rewrite to

$$\frac{l}{c_0}\cdot\frac{\partial}{\partial t}\left(\frac{1}{\alpha h_0}\eta\right)\approx-l\cdot\frac{\partial}{\partial x}\left(\frac{1}{\alpha h_0}\eta\right)$$

which is equivalent to

$$\frac{l}{c_0\alpha h_0}\eta_t\approx-\frac{l}{\alpha h_0}\eta_x$$

which still is a dimensionless equation. Now we redimensionalize the above equation by multiplying with the factor

$$\frac{c_0\alpha h_0}{l}$$

having the units [m/s], to arrive at a dimensional approximation

$$\eta_t=-c_0\eta_x$$

with units [m/s]. The next thing to do is to substitute this approximation for one of the x-derivatives in the last term of the KdV-equation (2.37), and arrive at last at the end of this chapter and the BBM-equation as given in [4]:

$$\eta_t+c_0\left(1+\frac{3}{2}\frac{\eta}{h_0}\right)\eta_x-\frac{\gamma}{c_0}\eta_{xxt}=0 \tag{2.3}$$

As previously described in chapter (2.1.1), we can for simplicity set $g=1$ and $h_0=1$, implying $c_0=1$, turning equation (2.3) into:

$$\eta_t+\eta_x+\frac{3}{2}\eta\eta_x-\frac{1}{6}\eta_{xxt}=0 \tag{2.4}$$

which is a simpler form of the BBM-equation.

2.2 Derivation of the equation for the horizontal fluid velocity

For studying the wave breaking, we will also need an equation for the horizontal fluid velocity $u(x,z,t)$ [m/s]. We look to equation (2.35), which gives the non-dimensional horizontal fluid velocity at the bottom ($\tilde{z}=0$):

$$\tilde{w} = \tilde{\eta} - \frac{1}{4}\alpha\tilde{\eta}^2 + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}} + O(\alpha^2+\beta^2) \tag{2.35}$$

Differentiating this equation twice w.r.t. $\tilde{x}$ yields

$$\tilde{w}_{\tilde{x}\tilde{x}} = \tilde{\eta}_{\tilde{x}\tilde{x}} - \frac{1}{2}\alpha(\tilde{\eta}\tilde{\eta}_{\tilde{x}})_{\tilde{x}} + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + O(\alpha^2+\beta^2) \tag{2.38}$$

Returning to the expression for the non-dimensional horizontal fluid velocity $\tilde{u}$, given in equation (2.17) and dropping higher order terms we have

$$\tilde{\varphi}_{\tilde{x}} = \tilde{u} = \tilde{w} - \beta\frac{\tilde{z}^2}{2}\cdot\tilde{w}_{\tilde{x}\tilde{x}} + O(\beta^2)$$

Substituting expressions in (2.35) and (2.38) for $\tilde{w}$ and for $\tilde{w}_{\tilde{x}\tilde{x}}$ in the equation above, we get

$$\tilde{\varphi}_{\tilde{x}} = \tilde{u} = \left[\tilde{\eta} - \frac{1}{4}\alpha\tilde{\eta}^2 + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}} + O(\alpha^2+\beta^2)\right] - \beta\frac{\tilde{z}^2}{2} \cdot\left[\tilde{\eta}_{\tilde{x}\tilde{x}} - \frac{1}{2}\alpha(\tilde{\eta}\tilde{\eta}_{\tilde{x}})_{\tilde{x}} + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}\tilde{x}\tilde{x}} + O(\alpha^2+\beta^2)\right] + O(\beta^2)$$

the bracket parentheses indicating the substitutions. After some rearrangement and sorting out of terms in higher orders of the parameters, we have

$$\tilde{\varphi}_{\tilde{x}} = \tilde{u} = \tilde{\eta} - \frac{1}{4}\alpha\tilde{\eta}^2 + \frac{1}{3}\beta\tilde{\eta}_{\tilde{x}\tilde{x}} - \frac{1}{2}\beta\tilde{z}^2\tilde{\eta}_{\tilde{x}\tilde{x}} + O(\alpha\beta,\alpha^2,\beta^2)$$

And with a final small rearrangement, to get

$$\tilde{u} = \tilde{\eta} - \frac{1}{4}\alpha\tilde{\eta}^2 + \beta\tilde{\eta}_{\tilde{x}\tilde{x}}\left(\frac{1}{3} - \frac{\tilde{z}^2}{2}\right) + O(\alpha\beta,\alpha^2,\beta^2) \tag{2.39}$$

the equation for the non-dimensional horizontal fluid velocity $\tilde{u}(\tilde{x},\tilde{z},\tilde{t})$.

However, we need the dimensional form of the last equation, so once again we have to do a re-dimensionalization. We use the same relationships for variables and differential operators as given in chapter 2.1. In addition, we need to introduce a characteristic horizontal fluid velocity u_{CH} [m/s], as well as the new dimensional/non-dimensional relationship

$$u = \tilde{u}u_{CH}$$

We already know $u = \varphi_x$ and $\tilde{u} = \tilde{\varphi}_{\tilde{x}}$, so we can use this to find the correct u_{CH} from the relation above. Including just the first two terms (the number of terms included is, logically, irrelevant for finding u_{CH}) of the expansions, we have

$$u = \varphi_x = f_x - \frac{z^2}{2} f_{xxx}$$

$$\tilde{u} = \tilde{\varphi}_{\tilde{x}} = \tilde{f}_{\tilde{x}} - \frac{\tilde{z}^2}{2} \cdot \tilde{f}_{\tilde{x}\tilde{x}\tilde{x}} \cdot \beta$$

Taking the latter and substituting into dimensional variables and differential operators, it reads

$$l \cdot \frac{\partial}{\partial x}\left(\frac{c_0}{gla}\varphi\right) = l \cdot \frac{\partial}{\partial x}\left(\frac{c_0}{gla} f\right) - \frac{h_0^2}{l^2} \cdot \frac{\left(\frac{z}{2}\right)^2}{2} \cdot l^3 \cdot \frac{\partial^3}{\partial x^3}\left(\frac{c_0}{gla} f\right)$$

which by some simplification and changing back to short-hand derivative notation turns into

$$\frac{c_0}{ga} \cdot \varphi_x = \frac{c_0}{ga} \cdot f_x - \frac{c_0}{ga} \cdot \frac{z^2}{2} \cdot f_{xxx}$$

Note that this is still a non-dimensional equation, it is still $\tilde{u}$! We observe that by multiplying by the factor

$$\frac{ga}{c_0}$$

with units [m/s], we are using the relationship $u = \tilde{u} u_{CH}$ to re-dimensionalize back to

$$u = \varphi_x = f_x - \frac{z^2}{2} f_{xxx}$$

obviously then with units [m/s], as wanted. So we have then found that we have

$$u_{CH} = \frac{ga}{c_0}$$

Investigating further, we see that

$$u_{CH} = \frac{ga}{c_0} = \frac{g\alpha h_0}{\sqrt{gh_0}} = \frac{\alpha\sqrt{gh_0} \cdot \sqrt{gh_0}}{\sqrt{gh_0}} = \alpha\sqrt{gh_0} = \alpha c_0$$

so we conclude that our characteristic horizontal fluid velocity is

$$u_{CH} = \alpha c_0$$

Having all this in place, and also remembering the expressions for parameters α and β, we do the substitutions to dimensional variables and differential operators and get

$$\frac{u}{u_{CH}} = \frac{1}{\alpha h_0}\eta - \frac{1}{4}\alpha \cdot \left(\frac{1}{\alpha h_0}\eta\right)^2 + \frac{h_0^2}{l^2} \cdot l^2 \cdot \frac{\partial}{\partial x^2}\left(\frac{1}{\alpha h_0}\eta\right) \cdot \left[\frac{1}{3} - \frac{\left(\frac{z}{h_0}\right)^2}{2}\right]$$

after some steps of tidying up, this becomes

$$\frac{u}{u_{CH}} = \frac{1}{\alpha h_0}\eta - \frac{1}{4} \cdot \frac{\eta^2}{\alpha h_0^2} + \frac{h_0}{\alpha}\eta_{xx}\left(\frac{1}{3} - \frac{z^2}{2h_0^2}\right)$$

Now, we substitute $u_{CH} = \alpha c_0$, and then multiply the equation by αc_0 to find

$$u = c_0 \cdot \left[\frac{1}{h_0}\eta - \frac{1}{4h_0^2}\eta^2 + h_0\eta_{xx}\left(\frac{1}{3} - \frac{z^2}{2h_0^2}\right)\right] \tag{2.40}$$

To get an expression for the horizontal fluid velocity which is consistent with the form of the BBM-equation given in (2.4), we set $g = 1$ and $h_0 = 1$, resulting in $c_0 = 1$, to get:

$$u = \eta - \frac{1}{4}\eta^2 + \eta_{xx}\left(\frac{1}{3} - \frac{z^2}{2}\right) \tag{2.41}$$

Note that by setting $z = 0$ in (2.40) we will get exactly the same expression for $u(z = 0) = w$ as we would get from re-dimensionalizing equation (2.35) for the bottom velocity directly, so this should give us some confidence in our derivations.

Note also that when simulating the physical bore problem, with initial value and boundary conditions as shown in Figure 4.2, the boundary condition on the RHS of the spatial domain will ensure that we have far-field surface deflection and horizontal fluid velocity equal to zero on the right, exactly as in [1].

2.3 The wave breaking criterion and the method to investigate wave breaking

At this point, it is useful to get a little ahead of ourselves and introduce a little sneak-peak into the results presented in chapter 5. When simulating the bore initial/boundary value problem with the BBM-equation numerically, the typical observation will be an emerging train of waves at the right-propagating bore front, in which the number of waves and the wave amplitudes will increase with time and apparently stabilize at a maximum, the latter in the case when breaking is not occuring. The leading wave (the right-most wave) will generally have the highest amplitude (maximum value of η), and it is at the crest of this leading wave that we anticipate that wave-breaking will occur, if it does. This is because the phase speed is approximately the same for all waves in the wave-train, and the horizontal fluid velocity is dependent on and increasing with the elevation z in the fluid, see eq. (2.40), so it should reach its maximum value in the domain at the top of the leading wave.

The approximative solution $\eta(x, t)$ of the initial/boundary value problem using the BBM-equation, obtained from the numerical scheme described in chapter 3, will provide us with the position of the leading wave at all time-steps. From this, we can calculate the phase speed of the leading wave by a simple velocity = distance/time consideration. This phase speed can be observed through simulations to be fairly constant with time, this will also be more thoroughly explained in chapter 5. However, the discrete nature of numerical solving will give a stuttering graph when trying to plot the phase speed as a function of time, so in order to get a smoother function, the phase speed is averaged over several time-steps backwards in time to give U [m/s], which is the value used as the leading wave phase speed in all calculations.

As mentioned above, from equation (2.40) it can be seen that the horizontal fluid velocity u increases with z, when we remember that η_{xx} is typically negative at the very top of a wave. For any x-value directly beneath a wave-crest, the horizontal fluid velocity will have its highest value when evaluated at the very surface of the wave. From this, it is at the top of the

leading wave we will find the horizontal fluid velocity of interest, that is equation (2.40) for u, evaluated at $z = h_0 + \eta(x,t)$ for the leading wave.

With this in mind, we will for this study use a similar breaking criterion as the one used by Kalisch and Bjørkavåg in [1]:

A wave solution $\eta(x,t)$ of (2.3) travelling with phase speed U starts to break if

$$c_0 \cdot \left[\frac{1}{h_0}\eta - \frac{1}{4h_0^2}\eta^2 + h_0\eta_{xx}\left(\frac{1}{3} - \frac{(h_0+\eta)^2}{2h_0^2}\right)\right] > U \tag{2.42}$$

If the form (2.4) of the BBM-equation is used, the horizontal fluid velocity is given by (2.41), and then the breaking criterion (2.42) turns into:

$$\eta - \frac{1}{4}\eta^2 + \eta_{xx}\left(\frac{1}{3} - \frac{(1+\eta)^2}{2}\right) > U \tag{2.43}$$

It should be stated clearly that in the event that wave-breaking occurs according to the above criterion, the BBM-equations (2.3) and (2.4) are no longer valid as descriptions of the physical water surface, as the flow then according to the breaking criterion will feature fluid particles leaving the wave-shape, and the surface will then no longer satisfy the demands to a mathematical function, that one and only one function value of $\eta(x,t)$ should exist for all points (x,t) in the domain. Note however that we cannot observe wave breaking in the BBM-equation alone, i.e. without using the breaking criterion. A simulation of the initial/boundary value problem for the bore using the BBM-equation to describe the free surface will continue to produce a solution for the surface after the breaking criterion has been met. This is due to a violation of the kinematic (the first) free surface boundary condition in the surface wave problem, arising from simplifications in the derivation of the equation. Thus, we must use the breaking criterion manually to check for wave breaking.

As a side-note, it should also be mentioned that the validity of the BBM-equation could possibly in some circumstances disappear somewhat before wave-breaking occurs, as the underlying theory has assumptions on wave-height and the steepness of the wave-profile, though we will not pursue this issue any further here.

As stressed by Kalisch and Bjørkavåg in [1], the goal is to study the onset of wave-breaking, not to provide a description of breaking waves.

3 SOLITARY WAVE SOLUTION OF THE BBM-EQUATION

3.1 Derivation of the analytical solitary wave solution

Like the related KdV-equation, the BBM-equation has two types of possible analytical solutions: A solitary wave solution and a periodical cnoidal wave solution. We choose here to focus on the solitary wave solution, as it is the most relevant for the task at hand.

We start out by looking at the BBM-equation (2.4):

$$\eta_t + \eta_x + \frac{3}{2}\eta\eta_x - \frac{1}{6}\eta_{xxt} = 0$$

Now, under the assumption that the solitary wave solution is a constant-shape wave travelling with constant phase speed c, we state that

$$\eta(x,t) = \Phi(s) = \Phi(x - ct)$$

If the above is substituted for η in (2.4), the equation turns into

$$-c\Phi' + \Phi' + \frac{3}{2}\Phi\Phi' + \frac{1}{6}c\Phi''' = 0 \tag{3.1}$$

and we note that we have now transformed the PDE (2.4) into an ODE. We integrate the above equation once, to arrive at

$$-c\Phi + \Phi + \frac{3}{4}\Phi^2 + \frac{1}{6}c\Phi'' = K$$

Now, it follows from the definition of the solitary wave that Φ and Φ'' must go to zero as s goes towards $\pm\infty$. That means that the integration constant K is zero, and with a little rearranging, we get

$$(1-c)\Phi + \frac{3}{4}\Phi^2 + \frac{1}{6}c\Phi'' = 0 \tag{3.2}$$

Next, we assume that the solution is on the form

$$\Phi = A\,\mathrm{sech}^2(Bs) \tag{3.3}$$

where A and B are constants. This means that we have

$$\Phi' = -2AB\,\tanh(Bs)\cdot sech^2(Bs) \tag{3.4}$$

$$\Phi'' = -2AB^2\,sech^4(Bs) + 4AB^2\,sech^2(Bs) - 4AB^2\,sech^4(Bs) \tag{3.5}$$

Now, we substitute the above expressions for Φ and Φ'' into equation (3.2), to find

$$\left[(1-c)\cdot A+\frac{2}{3}AB^2c\right]\cdot sech^2(Bs)+\left[\frac{3}{4}A^2-\frac{1}{3}AB^2c-\frac{2}{3}AB^2c\right]\cdot sech^4(Bs)=0 \quad (3.6)$$

where we have collected terms in powers of $sech$. We observe that the only way the above expression can be zero, is if both the terms inside the square brackets are zero. From

$$(1-c)\cdot A+\frac{2}{3}AB^2c=0$$

we get

$$B=\sqrt{\frac{3}{2}}\cdot\sqrt{\frac{c-1}{c}} \quad (3.7)$$

and further, from

$$\frac{3}{4}A^2-\frac{1}{3}AB^2c-\frac{2}{3}AB^2c=0$$

using expression (3.7) for B, we find

$$A=2(c-1) \quad (3.8)$$

Now, substituting the expressions for A and B into the assumed form (3.3) of the solitary wave solution, we arrive at

$$\eta(x,t)=2(c-1)\cdot sech^2\left(\sqrt{\frac{3}{2}}\cdot\sqrt{\frac{c-1}{c}}\cdot(x-ct)\right) \quad (3.9)$$

where as noted before, c is the phase speed of the solitary wave, and further $H=2(c-1)$ is the height of the solitary wave.

We note also that the form of the solution (3.9) is identical to the one found in [8], where the solitary wave solution has been found for a BBM-equation with different coefficients, indicating that we have performed the solution derivation correctly.

3.2 Maximum wave height for the solitary wave solution

In [3], a theoretical maximum wave height for solitary wave solutions of the KdV-equation is found by employing the analytical solitary wave solution and the breaking criterion. Here, we follow the same procedure to find a maximum wave height for the solitary wave solution of the BBM-equation.

It is convenient to first rewrite the solitary wave solution in terms of the wave height H. From $H = 2(c-1)$ we have

$$c = \frac{H}{2} + 1$$

which by substitution into (3.9) gives

$$\eta(x,t) = H \cdot sech^2\left(\sqrt{\frac{3}{2}} \cdot \sqrt{\frac{H}{H+2}} \cdot (x - ct)\right) \tag{3.10}$$

Differentiating (3.10) two times wrt. x, we get

$$\eta_x = -2 \cdot \sqrt{\frac{3}{2}} \cdot \sqrt{\frac{H}{H+2}} \cdot H \cdot tanh(\dots) \cdot sech^2(\dots) \tag{3.11}$$

$$\eta_{xx} = \frac{6H^2}{H+2} \cdot sech^2(\dots) \cdot tanh^2(\dots) - \frac{3H^2}{H+2} \cdot sech^4(\dots) \tag{3.12}$$

where for simplicity and compactness, (…) denotes the big parenthesis in (3.10).

The solitary wave has a constant shape for all time, meaning that the values of η, η_x and η_{xx} is the same everywhere in the domain if we follow a given point on the wave-form. Therefore, we choose for simplicity to evaluate at $(x,t) = (0,0)$, which gives

$$\eta(0,0) = H$$

$$\eta_{xx}(0,0) = -\frac{3H^2}{H+2}$$

Now, recall the breaking criterion (2.43):

$$\eta - \frac{1}{4}\eta^2 + \eta_{xx}\left(\frac{1}{3} - \frac{(1+\eta)^2}{2}\right) > U$$

Substituting the expressions for η and η_{xx} found above into the breaking criterion, as well as noting that the phase speed of the solitary wave is given by

$$c = U = \frac{H}{2} + 1$$

the breaking criterion becomes

$$H - \frac{1}{4}H^2 - \frac{3H^2}{H+2} \cdot \left(\frac{1}{3} - \frac{(1+H)^2}{2} \right) > \frac{H}{2} + 1 \tag{3.13}$$

Collecting all terms on the left hand side and doing some rearranging, we find the fourth order polynomial

$$\frac{3}{2}H^4 + \frac{11}{4}H^3 + \frac{1}{2}H^2 - 2 > 0 \tag{3.14}$$

meaning that solving

$$P(H) = \frac{3}{2}H^4 + \frac{11}{4}H^3 + \frac{1}{2}H^2 - 2 = 0 \tag{3.15}$$

should give the maximum theoretical height that a solitary wave can have without wave-breaking starting to occur.

The derivative $P'(H)$ is positive for all $H \geq 0$ and we are looking for a positive value of H, so the maximum height that we are looking for is the zero in the figure below:

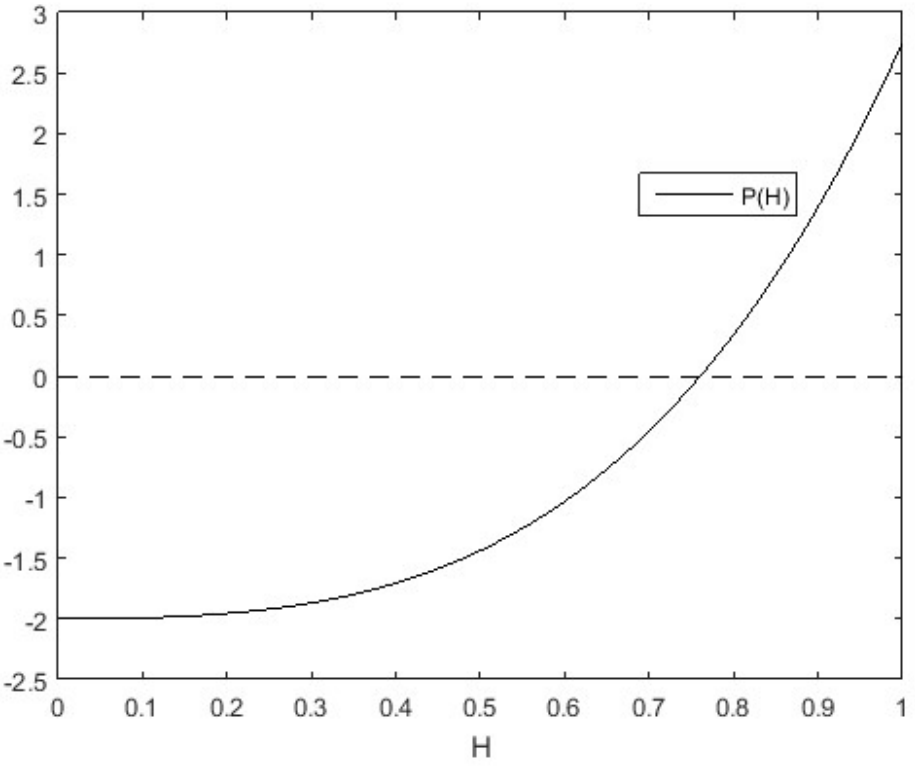

Figure 3.1 – The maximum height polynomial and its root

Using a bisection algorithm for root-finding with an error tolerance of 10^{-7}, the maximum height for the solitary wave is found to be

$$H_{MAX} = 0.7604389 \tag{3.16}$$

4 NUMERICAL STUDY OF THE BBM-EQUATION

4.1 Derivation of the numerical scheme for solving the initial/boundary value problem for the bore in the BBM-equation

4.1.1 Introduction

The steps in deriving the numerical scheme used for studying the bore initial/boundary value problem in this thesis are the same as the steps carried out in [1], though here, we will do a more thorough review.

The starting point is the BBM-equation given in [4], p. 463:

$$\eta_t + c_0\left(1 + \frac{3}{2}\frac{\eta}{h_0}\right)\eta_x - \frac{\gamma}{c_0}\eta_{xxt} = 0 \tag{4.1}$$

Writing out the parenthesis, we have:

$$\eta_t + c_0\eta_x + \frac{3}{2}\frac{c_0}{h_0}\eta\eta_x - \frac{\gamma}{c_0}\eta_{xxt} = 0$$

And introducing the constants $b = \frac{3}{2}\frac{c_0}{h_0}$ and $d = \frac{\gamma}{c_0}$ for convenience, we have:

$$\eta_t + c_0\eta_x + b\eta\eta_x - d\eta_{xxt} = 0$$

Collecting all terms involving derivatives with respect to t on the left-hand side, factoring out respectively η_t on the LHS and $-\partial_x$ on the RHS, we have:

$$(1 - d\partial_x^2)\eta_t = -\partial_x\left(\frac{b}{2}\eta\eta + c_0\eta\right)$$

Now by multiplying both sides by $(1 - d\partial_x^2)^{-1}$, and introducing simplified notation for the operators $\boldsymbol{A}_d^{-1} = (1 - d\partial_x^2)^{-1}$ and $\boldsymbol{B} = -\partial_x$we have

$$\eta_t = \boldsymbol{A}_d^{-1}\boldsymbol{B}\left(\frac{b}{2}\eta\eta + c_0\eta\right)$$

providing us with a convenient expression for the time derivative of η.

In [1], the trapezoidal method is used to advance the numerical solution forward in time. The trapezoidal rule is introduced in [9], p. 257 as a method of numerical integration based on simple geometrical considerations and given as

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{2}(y_0 + y_1) - \frac{h^3}{12}f''(c)$$

where $h = x_1 - x_0$ denotes the step length and c is a value between x_0 and x_1. So the *approximation* for one step is the $O(h^3)$ method

$$\int_{x_0}^{x_1} f(x)\,dx \approx \frac{h}{2}(y_0 + y_1)$$

estimating the area underneath the graph of $f(x)$ from x_0 to x_1. Now by substituting derivatives of the function (with respect to the independent variable) for the actual function values in the trapezoidal rule, we can get a method that finds the change in function value between x_0 and x_1 instead of the area under the graph, thus giving us an ODE solver:

$$f(x_1) - f(x_0) = \int_{x_0}^{x_1} f_x\,dx = h \cdot \frac{(f_x(x_1) + f_x(x_0)}{2} + O(h^3)$$

Rewriting the above formula in a notation more suitable for the problem at hand, we have

$$\eta^{n+1} - \eta^n = \int_{n\Delta t}^{(n+1)\Delta t} \eta_t\,dt = \frac{\eta_t^n + \eta_t^{n+1}}{2} \cdot \Delta t + O((\Delta t)^3)$$

And rearranging, we find the the $O(h^3)$ numerical scheme we want to employ for advancing the solution forward in time

$$\eta^{n+1} = \eta^n + \frac{\Delta t}{2} \cdot (\eta_t^n + \eta_t^{n+1})$$

Substituting our previously derived expression for η_t into the above equation, as well as introducing vector notation (in bold), we get

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^n + \frac{\Delta t}{2} \cdot \boldsymbol{A}_d^{-1}\boldsymbol{B}\left[\left(\frac{b}{2}\boldsymbol{\eta}^n\boldsymbol{\eta}^n + c_0\boldsymbol{\eta}^n\right) + \left(\frac{b}{2}\boldsymbol{\eta}^{n+1}\boldsymbol{\eta}^{n+1} + c_0\boldsymbol{\eta}^{n+1}\right)\right] \tag{4.2}$$

It should be stated that this as this is a numerical method with a *local error* of $O((\Delta t)^3)$ *for one step*, the method is then $O((\Delta t)^2)$ according to standard theoretical considerations, when applied to a domain with more than one step. This is the reason the time-stepping method is referred to as "second order" in [1], and the method (as derived so far) is actually what is referred to in [9], p. 260, as the *composite* trapezoidal method, applied as an ODE solver.

Now looking at the last formula (4.2), the numerical scheme is clearly implicit, as $\boldsymbol{\eta}^{n+1}$ is included in the RHS increment function. As is done in [1], we will make the numerical scheme explicit by substituting Euler predictions for the $\boldsymbol{\eta}^{n+1}$-values in the increment function in (4.2). From the standard Euler forward (explicit) formula (see for example [9], p. 284), we have

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}^n + \Delta t \cdot \boldsymbol{A}_d^{-1}\boldsymbol{B}\left(\frac{b}{2}\boldsymbol{\eta}^n\boldsymbol{\eta}^n + c_0\boldsymbol{\eta}^n\right) \tag{4.3}$$

And our numerical scheme after substituting in the Euler predictions is

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^n + \frac{\Delta t}{2} \cdot \boldsymbol{A}_d^{-1}\boldsymbol{B}\left[\left(\frac{b}{2}\boldsymbol{\eta}^n\boldsymbol{\eta}^n + c_0\boldsymbol{\eta}^n\right) + \left(\frac{b}{2}\boldsymbol{\eta}^*\boldsymbol{\eta}^* + c_0\boldsymbol{\eta}^*\right)\right] \tag{4.4}$$

As is noted in [1], this method is known as Heun's method (see e.g. [10], p. 483), and it is a second-order, explicit Runge-Kutta method. This is the method that has been used to advance the approximate solution of $\eta(x,t)$ forward in time in the simulations.

4.1.2 Deriving linear system of equations for Euler predictions $\boldsymbol{\eta}^*$

Rearranging the equation (4.3) for the Euler predictions, we have

$$\frac{\boldsymbol{\eta}^* - \boldsymbol{\eta}^n}{\Delta t} = \boldsymbol{A}_d^{-1}\boldsymbol{B}\left(\frac{b}{2}\boldsymbol{\eta}^n\boldsymbol{\eta}^n + c_0\boldsymbol{\eta}^n\right) \tag{4.5}$$

And multiplying on both sides by $\boldsymbol{A}_d = (1 - d\partial_x^2)$, we arrive at

$$\boldsymbol{A}_d\left(\frac{\boldsymbol{\eta}^* - \boldsymbol{\eta}^n}{\Delta t}\right) = \boldsymbol{B}\left(\frac{b}{2}\boldsymbol{\eta}^n\boldsymbol{\eta}^n + c_0\boldsymbol{\eta}^n\right)$$

Writing out the expression above we have

$$\frac{\boldsymbol{\eta}^* - \boldsymbol{\eta}^n}{\Delta t} - \left[\frac{d}{\Delta t}\cdot(\partial_x^2\boldsymbol{\eta}^* - \partial_x^2\boldsymbol{\eta}^n)\right] = -\frac{b}{2}\cdot\partial_x(\boldsymbol{\eta}^n\boldsymbol{\eta}^n) - c_0\partial_x(\boldsymbol{\eta}^n)$$

Now, we discretize the differential operators with standard central differences formulae for the operators ∂_x^2 and ∂_x, as given in [9], p. 246-247. The example below is given for the function η, when we are in position x_j of the discretizised spatial grid:

$$\partial_x^2\eta = \frac{\eta_{j-1} - 2\cdot\eta_j + \eta_{j+1}}{(\Delta x)^2} \tag{4.6}$$

$$\partial_x\eta = \frac{\eta_{j+1} - \eta_{j-1}}{2\Delta x} \tag{4.7}$$

Note that the above formulae are both approximations of $O((\Delta x)^2)$.

Substituting (4.6) and (4.7) for the differential operators in (4.5), as well as introducing indicial notation, we have for $1 \le j \le N$

$$\frac{\eta_j^* - \eta_j^n}{\Delta t} - \frac{d}{(\Delta x)^2}\cdot\frac{\left(\eta_{j-1}^* - 2\eta_j^* + \eta_{j+1}^*\right) - \left(\eta_{j-1}^n - 2\eta_j^n + \eta_{j+1}^n\right)}{\Delta t} = -\frac{b}{2}\cdot\frac{\left(\eta_{j+1}^n\eta_{j+1}^n - \eta_{j-1}^n\eta_{j-1}^n\right)}{2\Delta x} - c_0\cdot\frac{\left(\eta_{j+1}^n - \eta_{j-1}^n\right)}{2\Delta x}$$

And designating the RHS as $r_j^{\eta^n}$ we have

$$\frac{\eta_j^* - \eta_j^n}{\Delta t} - \frac{d}{(\Delta x)^2}\cdot\frac{\left(\eta_{j-1}^* - 2\eta_j^* + \eta_{j+1}^*\right) - \left(\eta_{j-1}^n - 2\eta_j^n + \eta_{j+1}^n\right)}{\Delta t} = r_j^{\eta^n} \tag{4.8}$$

At this point, it could be helpful to illustrate with a figure how we are going to solve the BBM-equation numerically on an (x,t)-grid.

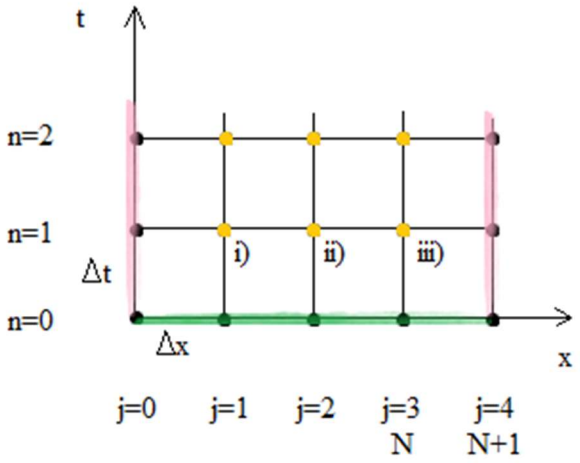

Figure 4.1 - The discretized (x,t)-domain

In Figure 4.1 we see how the letter j indicates x-positions, and the letter n is the time-step index. Pink color (j = 0 and j = N+1) indicates the positions on the grid where all values for the solution variable are known (for all time) from the boundary conditions. The green color (n = 0) indicates the positions on the grid where all values for the solution variable are known from the initial conditions, which are given by the function

$$\eta(x,0) = \frac{1}{2}a_0(1 - \tanh(\kappa x - x_0)) \tag{4.9}$$

where κ is a parameter controlling the steepness of the initial bore front, and x_0 is a term used for translating the initial position of the bore front to the desired point in the spatial interval.

Consequently, it is the "internal" grid points, shown in orange color in the figure and marked i) – iii) for the first three, in which we must use the numerical scheme to calculate for the solution variable. Here, we have chosen $N = 3$ as it is the smallest spatial grid discretization that will give us the opportunity to explain some important points about the calculation, as will be shown in the following.

Writing out equation (4.8) for the x-indices j = 1, 2 and 3, we have

$$i)\ For\ j = 1\text{:}\ \frac{\eta_1^* - \eta_1^n}{\Delta t} - \frac{d}{(\Delta x)^2}\cdot\frac{(\eta_0^* - 2\eta_1^* + \eta_2^*) - (\eta_0^n - 2\eta_1^n + \eta_2^n)}{\Delta t} = -\frac{b}{2}\cdot\frac{(\eta_2^n\eta_2^n - \eta_0^n\eta_0^n)}{2\Delta x} - c_0\cdot\frac{(\eta_2^n - \eta_0^n)}{2\Delta x}$$

$$ii)\ For\ j = 2\text{:}\ \frac{\eta_2^* - \eta_2^n}{\Delta t} - \frac{d}{(\Delta x)^2}\cdot\frac{(\eta_1^* - 2\eta_2^* + \eta_3^*) - (\eta_1^n - 2\eta_2^n + \eta_3^n)}{\Delta t} = -\frac{b}{2}\cdot\frac{(\eta_3^n\eta_3^n - \eta_1^n\eta_1^n)}{2\Delta x} - c_0\cdot\frac{(\eta_3^n - \eta_1^n)}{2\Delta x}$$

$$iii)\ For\ j=3:\ \frac{\eta_3^*-\eta_3^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_2^*-2\eta_3^*+\eta_4^*)-(\eta_2^n-2\eta_3^n+\eta_4^n)}{\Delta t}$$
$$=-\frac{b}{2}\cdot\frac{(\eta_4^n\eta_4^n-\eta_2^n\eta_2^n)}{2\Delta x}-c_0\cdot\frac{(\eta_4^n-\eta_2^n)}{2\Delta x}$$

where the right-hand sides are $r_1^{\eta^n}, r_2^{\eta^n}\ and\ r_3^{\eta^n}$, respectively.

Note that this is as system of 3 linear equations with 3 unknowns η_1^*, η_2^* and η_3^*, so it should be solvable. In general, we will have a linear system of N equations in N unknowns.

Now in order to be able to easily write these equations into a computer program for numerical solving, it is convenient to have them in vector form. We try to rewrite equations i) – iii) above as

$$\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^*-\eta_1^n\\ \eta_2^*-\eta_2^n\\ \eta_3^*-\eta_3^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2 & 1 & 0\\ 1 & -2 & 1\\ 0 & 1 & -2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^*-\eta_1^n\\ \eta_2^*-\eta_2^n\\ \eta_3^*-\eta_3^n\end{bmatrix}=\begin{bmatrix}r_1^{\eta^n}\\ r_2^{\eta^n}\\ r_3^{\eta^n}\end{bmatrix} \qquad (4.10)$$

Writing out the vector equation (4.10), we get

$$a)\ \frac{\eta_1^*-\eta_1^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(-2\eta_1^*+\eta_2^*)-(-2\eta_1^n+\eta_2^n)}{\Delta t}=r_1^{\eta^n}$$

$$b)\ \frac{\eta_2^*-\eta_2^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_1^*-2\eta_2^*+\eta_3^*)-(\eta_1^n-2\eta_2^n+\eta_3^n)}{\Delta t}=r_2^{\eta^n}$$

$$c)\ \frac{\eta_3^*-\eta_3^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_2^*-2\eta_3^*)-(\eta_2^n-2\eta_3^n)}{\Delta t}=r_3^{\eta^n}$$

We observe that on the LHS in a), we are missing the term

$$-\frac{d}{(\Delta x)^2}\cdot\frac{\eta_0^*-\eta_0^n}{\Delta t}$$

in order for a) to equal i), and similarly on the LHS in c), we are missing the term

$$-\frac{d}{(\Delta x)^2}\cdot\frac{\eta_4^*-\eta_4^n}{\Delta t}$$

in order for c) to equal iii). This would of course correspond to the addition of the terms above with a *positive* sign to the LHS in equations i) and iii), respectively. So we sort the problem out by updating the RHS in equations a) and c) by adding the the terms with positive signs:

$$r_1^{\eta^n}\xrightarrow{Update} r_1^{\eta^n}+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_0^*-\eta_0^n}{\Delta t}$$

$$r_3^{\eta^n} \xrightarrow{Update} r_3^{\eta^n} + \frac{d}{(\Delta x)^2} \cdot \frac{\eta_4^* - \eta_4^n}{\Delta t}$$

Note that equation b) came out equal to equation ii) directly from the vector equation. In general, we will always need to do the RHS update explained above for the equations corresponding to x-grid positions with j-indices 1 and N, the ones next to the boundary grid point with j-indices 0 and N+1. All "internal" equations (1 < j < N) will come out fine directly from the vector form. This is just a result of multiplying by the tridiagonal matrix in the vector form of the equations.

After the updates, we now have the linear system of equations

$$\frac{1}{\Delta t} \cdot \begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \eta_3^* - \eta_3^n \end{bmatrix} - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \eta_3^* - \eta_3^n \end{bmatrix} = \begin{bmatrix} r_1^{\eta^n} + \frac{d}{(\Delta x)^2} \cdot \frac{\eta_0^* - \eta_0^n}{\Delta t} \\ r_2^{\eta^n} \\ r_3^{\eta^n} + \frac{d}{(\Delta x)^2} \cdot \frac{\eta_4^* - \eta_4^n}{\Delta t} \end{bmatrix} \tag{4.11}$$

Again referring to Figure 4.1,we note that the values η_0^* and η_4^* (the latter will generally be η_{N+1}^*) are Euler predictions at the time $t = (n+1)\Delta t$, so the Dirichlet boundary conditions at time $t^{n+1} = (n+1)\Delta t$, that is (respectively) $\eta_l(t^{n+1})$ for η_0^*, and $\eta_r(t^{n+1})$ for η_{N+1}^*, must be substituted in their place.

Similarly, we must also substitute for the values η_0^n and η_4^n (the latter will generally be η_{N+1}^n) with the Dirichlet boundary conditions at time $t^n = n\Delta t$, that is (respectively) $\eta_l(t^n)$ for η_0^n, and $\eta_r(t^n)$ for η_{N+1}^n.

Now, having incorporated the boundary conditions, our linear system of equations is

$$\frac{1}{\Delta t} \cdot \begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \eta_3^* - \eta_3^n \end{bmatrix} - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \eta_3^* - \eta_3^n \end{bmatrix} = \begin{bmatrix} r_1^{\eta^n} + \frac{d}{(\Delta x)^2} \cdot \frac{\eta_l(t^{n+1}) - \eta_l(t^n)}{\Delta t} \\ r_2^{\eta^n} \\ r_3^{\eta^n} + \frac{d}{(\Delta x)^2} \cdot \frac{\eta_r(t^{n+1}) - \eta_r(t^n)}{\Delta t} \end{bmatrix} \tag{4.12}$$

Note that we have taken boundary conditions into the equations, but we have still not said anything about the actual *values* of the boundary conditions. For the bore problem at hand, we will assume constant values for all time for η at the boundaries, $\eta_l(t) = a_0$ (the height of the bore), and $\eta_r(t) = 0$, both measured from the height h_0 of the surface of undisturbed water, refer to Figure 4.2 below.

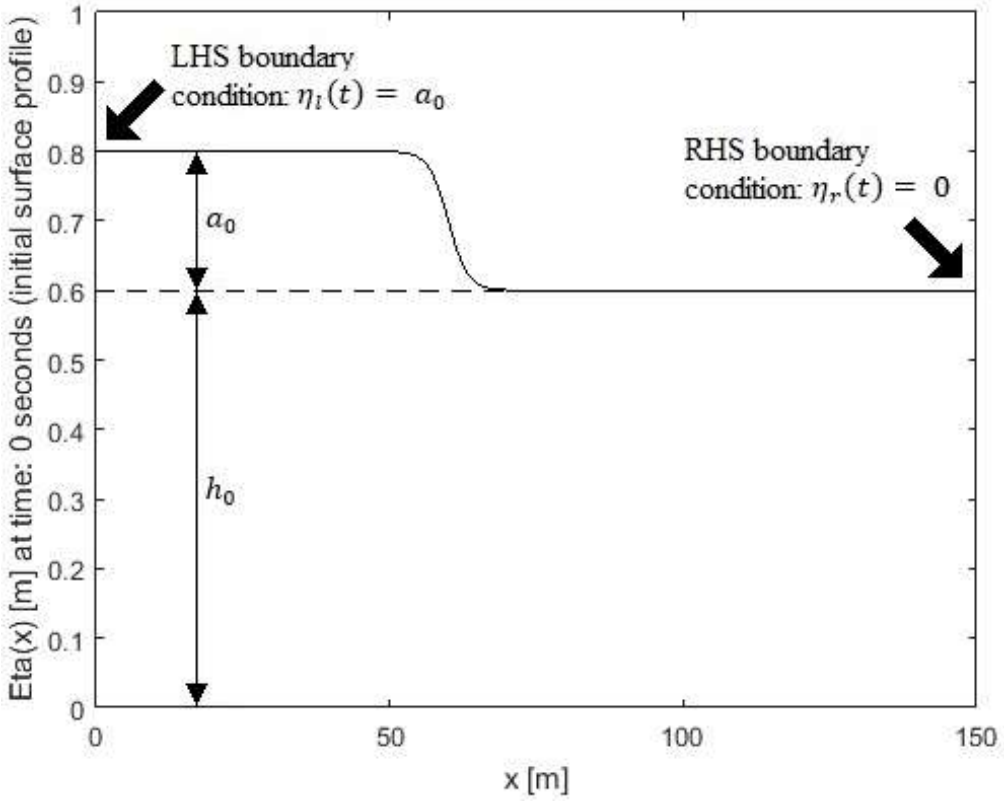

Figure 4.2 – LHS and RHS boundary conditions are constant

Having constant boundary conditions will of course imply that for our problem, we will always have $\eta_l(t^{n+1}) = \eta_l(t^n)$ and $\eta_r(t^{n+1}) = \eta_r(t^n)$, and so in our case the "update terms" in the RHS of equations for x-positions with j-indices 1 and N disappears, and then in the general case, we are left with

$$\frac{1}{\Delta t}\cdot\begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \vdots \\ \eta_{N-1}^* - \eta_{N-1}^n \\ \eta_N^* - \eta_N^n \end{bmatrix} - \frac{d}{(\Delta x)^2 \cdot \Delta t}\cdot\begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix}$$
$$\cdot\begin{bmatrix} \eta_1^* - \eta_1^n \\ \eta_2^* - \eta_2^n \\ \vdots \\ \eta_{N-1}^* - \eta_{N-1}^n \\ \eta_N^* - \eta_N^n \end{bmatrix} = \begin{bmatrix} r_1^{\eta^n} \\ r_2^{\eta^n} \\ \vdots \\ r_{N-1}^{\eta^n} \\ r_N^{\eta^n} \end{bmatrix} \tag{4.13}$$

which is the linear system of N equations to be solved for the N unknowns $\eta_1^*, \eta_2^*, \ldots, \eta_N^*$.

4.1.3 Deriving linear system of equations for $\boldsymbol{\eta}^{n+1}$

Returning now to the numerical scheme in (4.4)

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^{n} + \frac{\Delta t}{2} \cdot \boldsymbol{A}_d^{-1}\boldsymbol{B}\left[\left(\frac{b}{2}\boldsymbol{\eta}^{n}\boldsymbol{\eta}^{n} + c_0\boldsymbol{\eta}^{n}\right) + \left(\frac{b}{2}\boldsymbol{\eta}^{*}\boldsymbol{\eta}^{*} + c_0\boldsymbol{\eta}^{*}\right)\right] \tag{4.4}$$

We will proceed by exactly the same procedure as for the Euler predictions in chapter 4.1.2. Rearranging the equation above, we have

$$\frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t} = \frac{\mathbf{1}}{\mathbf{2}} \cdot \boldsymbol{A}_d^{-1}\boldsymbol{B}\left[\left(\frac{b}{2}\boldsymbol{\eta}^{n}\boldsymbol{\eta}^{n} + c_0\boldsymbol{\eta}^{n}\right) + \left(\frac{b}{2}\boldsymbol{\eta}^{*}\boldsymbol{\eta}^{*} + c_0\boldsymbol{\eta}^{*}\right)\right]$$

And multiplying on both sides by $\boldsymbol{A}_d = (1 - d\partial_x^2)$, we arrive at

$$\boldsymbol{A}_d\left(\frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t}\right) = \frac{\mathbf{1}}{\mathbf{2}} \cdot \boldsymbol{B}\left[\left(\frac{b}{2}\boldsymbol{\eta}^{n}\boldsymbol{\eta}^{n} + c_0\boldsymbol{\eta}^{n}\right) + \left(\frac{b}{2}\boldsymbol{\eta}^{*}\boldsymbol{\eta}^{*} + c_0\boldsymbol{\eta}^{*}\right)\right]$$

If we substitute for the operators $\boldsymbol{A}_d$ and $\boldsymbol{B}$, we get

$$\frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t} - \frac{d}{\Delta t} \cdot (\partial_x^2\boldsymbol{\eta}^{n+1} - \partial_x^2\boldsymbol{\eta}^{n}) = -\frac{1}{2} \cdot \left[\left(\frac{b}{2} \cdot \partial_x(\boldsymbol{\eta}^{n}\boldsymbol{\eta}^{n}) + c_0\partial_x(\boldsymbol{\eta}^{n})\right) + \left(\frac{b}{2} \cdot \partial_x(\boldsymbol{\eta}^{*}\boldsymbol{\eta}^{*}) + c_0\partial_x(\boldsymbol{\eta}^{*})\right)\right]$$

And applying again the central differences formulae (4.6) and (4.7) described in chapter 3.1.2, as well as rewriting to indicial form, we have for $1 \leq j \leq N$

$$\begin{aligned}\frac{\eta_j^{n+1} - \eta_j^{n}}{\Delta t} - \frac{d}{(\Delta x)^2} \cdot \frac{\left(\eta_{j-1}^{n+1} - 2\eta_j^{n+1} + \eta_{j+1}^{n+1}\right) - \left(\eta_{j-1}^{n} - 2\eta_j^{n} + \eta_{j+1}^{n}\right)}{\Delta t} \\ = \frac{1}{2}\left[\left(-\frac{b}{2} \cdot \frac{\left(\eta_{j+1}^{n}\eta_{j+1}^{n} - \eta_{j-1}^{n}\eta_{j-1}^{n}\right)}{2\Delta x} - c_0 \cdot \frac{\left(\eta_{j+1}^{n} - \eta_{j-1}^{n}\right)}{2\Delta x}\right)\right. \\ \left. + \left(-\frac{b}{2} \cdot \frac{\left(\eta_{j+1}^{*}\eta_{j+1}^{*} - \eta_{j-1}^{*}\eta_{j-1}^{*}\right)}{2\Delta x} - c_0 \cdot \frac{\left(\eta_{j+1}^{*} - \eta_{j-1}^{*}\right)}{2\Delta x}\right)\right]\end{aligned} \tag{4.14}$$

Inside the bracket parenthesis in equation (4.14), we have two expressions in the main round parenthesis. We recognize the first one as $r_j^{\eta^n}$, and we designate the second one as $r_j^{\eta^*}$, so then we have

$$\begin{aligned}\frac{\eta_j^{n+1} - \eta_j^{n}}{\Delta t} - \frac{d}{(\Delta x)^2} \cdot \frac{\left(\eta_{j-1}^{n+1} - 2\eta_j^{n+1} + \eta_{j+1}^{n+1}\right) - \left(\eta_{j-1}^{n} - 2\eta_j^{n} + \eta_{j+1}^{n}\right)}{\Delta t} \\ = \frac{1}{2}\left(r_j^{\eta^n} + r_j^{\eta^*}\right)\end{aligned} \tag{4.15}$$

Now again as in chapter 3.1.2, choosing N = 3 and writing out equation (4.15) for the x-indices j = 1, 2 and 3, we have

$$i)\ For\ j=1\text{:}\ \frac{\eta_1^{n+1}-\eta_1^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_0^{n+1}-2\eta_1^{n+1}+\eta_2^{n+1})-(\eta_0^n-2\eta_1^n+\eta_2^n)}{\Delta t}=\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)$$

$$ii)\ For\ j=2\text{:}\ \frac{\eta_2^{n+1}-\eta_2^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_1^{n+1}-2\eta_2^{n+1}+\eta_3^{n+1})-(\eta_1^n-2\eta_2^n+\eta_3^n)}{\Delta t}=\frac{1}{2}\left(r_2^{\eta^n}+r_2^{\eta^*}\right)$$

$$iii)\ For\ j=3\text{:}\ \frac{\eta_3^{n+1}-\eta_3^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_2^{n+1}-2\eta_3^{n+1}+\eta_4^{n+1})-(\eta_2^n-2\eta_3^n+\eta_4^n)}{\Delta t}=\frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)$$

As for the equations for the Euler predictions, we want to have the equations above in vector form to make programming easier. We try to rewrite equations i) – iii) above as

$$\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2&1&0\\1&-2&1\\0&1&-2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}=\frac{1}{2}\begin{bmatrix}r_1^{\eta^n}+r_1^{\eta^*}\\ r_2^{\eta^n}+r_2^{\eta^*}\\ r_3^{\eta^n}+r_3^{\eta^*}\end{bmatrix}\tag{4.16}$$

And writing out the vector equation (4.16) above, we get

$$a)\ \frac{\eta_1^{n+1}-\eta_1^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(-2\eta_1^{n+1}+\eta_2^{n+1})-(-2\eta_1^n+\eta_2^n)}{\Delta t}=\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)$$

$$b)\ \frac{\eta_2^{n+1}-\eta_2^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_1^{n+1}-2\eta_2^{n+1}+\eta_3^{n+1})-(\eta_1^n-2\eta_2^n+\eta_3^n)}{\Delta t}=\frac{1}{2}\left(r_2^{\eta^n}+r_2^{\eta^*}\right)$$

$$c)\ \frac{\eta_3^{n+1}-\eta_3^n}{\Delta t}-\frac{d}{(\Delta x)^2}\cdot\frac{(\eta_2^{n+1}-2\eta_3^{n+1})-(\eta_2^n-2\eta_3^n)}{\Delta t}=\frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)$$

And again, precisely as for the equations for the Euler predictions, we observe that in a), we are missing the term

$$-\frac{d}{(\Delta x)^2}\cdot\frac{\eta_0^{n+1}-\eta_0^n}{\Delta t}$$

in order for a) to equal i), and similarly on the LHS in c), we are missing the term

$$-\frac{d}{(\Delta x)^2}\cdot\frac{\eta_4^{n+1}-\eta_4^n}{\Delta t}$$

in order for c) to equal iii). As before, this would of course correspond to the addition of the terms above with a *positive* sign to the LHS in equations i) and iii), respectively. So once again we sort the problem out by updating the RHS in equations a) and c) by adding the the terms with positive signs:

$$\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)\xrightarrow{Update}\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_0^{n+1}-\eta_0^n}{\Delta t}$$

$$\frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)\xrightarrow{Update}\frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_4^{n+1}-\eta_4^n}{\Delta t}$$

After the updates, we now have the linear system of equations

$$\begin{aligned}\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2&1&0\\1&-2&1\\0&1&-2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}\\ =\begin{bmatrix}\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_0^{n+1}-\eta_0^n}{\Delta t}\\ \frac{1}{2}\left(r_2^{\eta^n}+r_2^{\eta^*}\right)\\ \frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_4^{n+1}-\eta_4^n}{\Delta t}\end{bmatrix}\end{aligned}\tag{4.17}$$

And we note that the values η_0^{n+1}, η_0^n, η_4^{n+1} (generally η_{N+1}^{n+1}) and η_4^n (generally η_{N+1}^n) are given by the boundary conditions as follows:

$$\eta_0^{n+1}=\eta_l(t^{n+1})$$

$$\eta_0^n=\eta_l(t^n)$$

$$\eta_{N+1}^{n+1}=\eta_r(t^{n+1})$$

$$\eta_{N+1}^n=\eta_r(t^n)$$

By utilizing the above, our linear system of equations is now

$$\begin{aligned}\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2&1&0\\1&-2&1\\0&1&-2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \eta_3^{n+1}-\eta_3^n\end{bmatrix}\\ =\begin{bmatrix}\frac{1}{2}\left(r_1^{\eta^n}+r_1^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_l(t^{n+1})-\eta_l(t^n)}{\Delta t}\\ \frac{1}{2}\left(r_2^{\eta^n}+r_2^{\eta^*}\right)\\ \frac{1}{2}\left(r_3^{\eta^n}+r_3^{\eta^*}\right)+\frac{d}{(\Delta x)^2}\cdot\frac{\eta_r(t^{n+1})-\eta_r(t^n)}{\Delta t}\end{bmatrix}\end{aligned}\tag{4.18}$$

But as before, the bore problem with *constant* boundary conditions $\eta_l(t) = a_0$ and $\eta_r(t) = 0$ will imply $\eta_l(t^{n+1}) = \eta_l(t^n)$ and $\eta_r(t^{n+1}) = \eta_r(t^n)$ for all times, so that the "update terms" in the RHS of equations for x-positions with j-indices 1 and N disappears, and then in the general case, we are left with

$$\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \vdots\\ \eta_{N-1}^{n+1}-\eta_{N-1}^n\\ \eta_N^{n+1}-\eta_N^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2&1&0&\cdots&\cdots&0\\ 1&-2&1&0&\cdots&0\\ 0&\ddots&\ddots&\ddots&\ddots&\vdots\\ \vdots&\ddots&\ddots&\ddots&\ddots&0\\ 0&\cdots&0&1&-2&1\\ 0&\cdots&\cdots&0&1&-2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^{n+1}-\eta_1^n\\ \eta_2^{n+1}-\eta_2^n\\ \vdots\\ \eta_{N-1}^{n+1}-\eta_{N-1}^n\\ \eta_N^{n+1}-\eta_N^n\end{bmatrix}=\frac{1}{2}\begin{bmatrix}r_1^{\eta^n}+r_1^{\eta^*}\\ r_2^{\eta^n}+r_2^{\eta^*}\\ \vdots\\ r_{N-1}^{\eta^n}+r_{N-1}^{\eta^*}\\ r_N^{\eta^n}+r_N^{\eta^*}\end{bmatrix} \tag{4.19}$$

which is the linear system of N equations to be solved for the N unknowns $\eta_1^{n+1}, \eta_2^{n+1}$, …, η_N^{n+1}.

4.1.4 Rearranging linear systems of equations for programming purposes

For solving the linear system of equations introduced in chapters 4.1.2 and 4.1.3, Matlabs built-in "backslash-solver" for linear systems of equations has been employed. This is not an "inexpensive" solver in terms of calculation costs, for example, it utilizes matrix invertion. More inexpensive solvers could have been used, e.g. a Gaussian elimination algorithm. But for the numerical simulations of the problem at hand, the "backslash-solver" has worked well. Also, the actual solving of linear systems of equations is not the main focus of this thesis, making the "backslash-solver" an easy and acceptable choice.

To use the "backslash-solver", the system of equations must be provided for Matlab in the form $A\boldsymbol{x} = \boldsymbol{b}$, which the software then solves for vector **x**. We must therefore rearrange the systems of equations to this form.

Starting with equations (4.13) for the Euler predictions $\boldsymbol{\eta}^*$

$$\frac{1}{\Delta t}\cdot\begin{bmatrix}\eta_1^*-\eta_1^n\\ \eta_2^*-\eta_2^n\\ \vdots\\ \eta_{N-1}^*-\eta_{N-1}^n\\ \eta_N^*-\eta_N^n\end{bmatrix}-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot\begin{bmatrix}-2&1&0&\cdots&\cdots&0\\ 1&-2&1&0&\cdots&0\\ 0&\ddots&\ddots&\ddots&\ddots&\vdots\\ \vdots&\ddots&\ddots&\ddots&\ddots&0\\ 0&\cdots&0&1&-2&1\\ 0&\cdots&\cdots&0&1&-2\end{bmatrix}\cdot\begin{bmatrix}\eta_1^*-\eta_1^n\\ \eta_2^*-\eta_2^n\\ \vdots\\ \eta_{N-1}^*-\eta_{N-1}^n\\ \eta_N^*-\eta_N^n\end{bmatrix}=\begin{bmatrix}r_1^{\eta^n}\\ r_2^{\eta^n}\\ \vdots\\ r_{N-1}^{\eta^n}\\ r_N^{\eta^n}\end{bmatrix}$$

we name the tridiagonal matrix T, and rewrite to a more compact form:

$$\frac{1}{\Delta t}\cdot(\boldsymbol{\eta}^*-\boldsymbol{\eta}^n)-\frac{d}{(\Delta x)^2\cdot\Delta t}\cdot T\cdot(\boldsymbol{\eta}^*-\boldsymbol{\eta}^n)=\boldsymbol{r}^{\eta^n}$$

After a little rearrangement, and recognizing that for the identity matrix I, we have $I \cdot \boldsymbol{x} = \boldsymbol{x}$ for any vector **x**, we have

$$\frac{1}{\Delta t} \cdot I \cdot \boldsymbol{\eta}^* - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T \cdot \boldsymbol{\eta}^* = \boldsymbol{r}^{\eta^n} + \frac{1}{\Delta t} \cdot \boldsymbol{\eta}^n - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T \cdot \boldsymbol{\eta}^n$$

Factoring out $\boldsymbol{\eta}^*$on the LHS and naming the RHS $\boldsymbol{b}_1$, we have

$$\left[\frac{1}{\Delta t} \cdot I - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T\right] \boldsymbol{\eta}^* = \boldsymbol{b}_1 \tag{4.20}$$

and even more compactly by taking the entire matrix in the bracket parenthesis as A

$$A\boldsymbol{\eta}^* = \boldsymbol{b}_1 \tag{4.21}$$

and we now have the correct form to give as input for Matlab.

For the linear system of equations (4.19) for $\boldsymbol{\eta}^{n+1}$

$$\frac{1}{\Delta t} \cdot \begin{bmatrix} \eta_1^{n+1} - \eta_1^n \\ \eta_2^{n+1} - \eta_2^n \\ \vdots \\ \eta_{N-1}^{n+1} - \eta_{N-1}^n \\ \eta_N^{n+1} - \eta_N^n \end{bmatrix} - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot \begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} \eta_1^{n+1} - \eta_1^n \\ \eta_2^{n+1} - \eta_2^n \\ \vdots \\ \eta_{N-1}^{n+1} - \eta_{N-1}^n \\ \eta_N^{n+1} - \eta_N^n \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} r_1^{\eta^n} + r_1^{\eta^*} \\ r_2^{\eta^n} + r_2^{\eta^*} \\ \vdots \\ r_{N-1}^{\eta^n} + r_{N-1}^{\eta^*} \\ r_N^{\eta^n} + r_N^{\eta^*} \end{bmatrix}$$

Again, we name the tridiagonal matrix as T, and rearrange into a more compact form

$$\frac{1}{\Delta t} \cdot (\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n) - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T \cdot (\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n) = \frac{1}{2}\left(\boldsymbol{r}^{\eta^n} + \boldsymbol{r}^{\eta^*}\right)$$

After some rearranging and once again employing the identity matrix I, we arrive at

$$\frac{1}{\Delta t} \cdot I \cdot \boldsymbol{\eta}^{n+1} - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T \cdot \boldsymbol{\eta}^{n+1} = \frac{1}{2}\left(\boldsymbol{r}^{\eta^n} + \boldsymbol{r}^{\eta^*}\right) + \frac{1}{\Delta t} \cdot \boldsymbol{\eta}^n - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T \cdot \boldsymbol{\eta}^n$$

Factoring out $\boldsymbol{\eta}^{n+1}$ on the LHS and naming the RHS $\boldsymbol{b}_2$, we have

$$\left[\frac{1}{\Delta t} \cdot I - \frac{d}{(\Delta x)^2 \cdot \Delta t} \cdot T\right] \boldsymbol{\eta}^{n+1} = \boldsymbol{b}_2 \tag{4.22}$$

And recognizing that the matrix inside the square brackets is the same as we had for the Euler prediction equations, we have

$$A\boldsymbol{\eta}^{n+1} = \boldsymbol{b}_2 \tag{4.23}$$

which is the form that Matlab needs.

4.1.5 Summary

To sum up, we state the "recipe" for solving the bore initial value/boundary condition problem for the BBM-equation numerically, by using the linear systems of equations introduced in this chapter:

(Starting at time t = 0 and time-step index n = 0)

1) Solve equation (4.21) $A\boldsymbol{\eta}^{*} = \boldsymbol{b}_1$ for $\boldsymbol{\eta}^{*}$.
2) Solve equation (4.23) $A\boldsymbol{\eta}^{n+1} = \boldsymbol{b}_2$ for $\boldsymbol{\eta}^{n+1}$.
3) Repeat steps 1) and 2) for the desired number of time-steps.

4.2 The bore front/leading wave propagation velocity

As mentioned in chapter 2, after having solved the BBM-equation numerically with the method described in chapter 4, we know the approximative solution of $\eta(x,t)$ in all of the domain, which means we know where the leading wave is located for all times, with the accuracy given by the numerical scheme. We can then calculate U by the simple reasoning

$$U = \frac{Distance\ travelled}{Time\ elapsed}$$

At this point, some notes about the phase speed U in a discrete model like this one could be mentioned:

We can never observe any momentanous phase speed U. If we for a moment disregard the possibility of calculating the average phase speed U over several time-steps, we can only logically calculate the average speed U of the leading wave during one single time-step. Of course, as the time-step Δt approaches zero, any single-time-step average speed approaches a momentanous phase speed U for the leading wave.

As noted in chapter 2.3, the bore-front/leading wave crest propagation speed U [m/s] will typically be fairly constant, but calculating U from every single time-step and then trying to plot $U(t)$ will result in a stuttering graph due to the discrete nature of the numerical scheme.

To solve this "problem", we calculate the average of the propagation speed over a number of previous (or future) time steps, and then designate this calculated value as our bore-front/leading wave propagation speed at the current time step.

Of course, if the "true" value of U varies a lot over short time periods, it would be a bad idea to take the average over a large number of previous (or future) time-steps and then assigning the calculated U to the current time-step. However, we do expect from theory and previous works that U is anticipated to be fairly constant over time, which we also will see in chapter 5. We

conclude from this that the averaging of U in our calculations is "safe", and will provide a good approximation of the true value of the leading wave phase speed.

The formula for U at the time $n\Delta t$ that is used in the Matlab simulation program is equivalent to

$$U(n\Delta t) = \frac{x\ position(n\Delta t) - \ x\ position[(n-s)\Delta t]}{s\Delta t}$$

where s is the number of time-steps backward in time that we want to average over to find U.

Of course, the manual averaging of the propagation speed over s steps requires that we actually have taken s steps before the averaging calculations are initiated, so also built into the Matlab program is code doing a logical checked to ensure that this is taken care of.

After the above described "main" averaging has been carried out, there might still be small stuttering part on the plot of the time series U(t). Therefore, Matlabs built-in smooth-function, which is a moving-average calculation, has been applied on top of this.

Several checks on the calculation of the phase speed U has been done, using the simple method of looking directly at the solution $\eta(x,t)$ and observing the distance travelled by the leading wave between two arbitrary time-steps, and then comparing this to the distance given by multiplying an approximate average of the phase speed from the time series $U(t)$ by the time difference between the time steps. This shows that the calculation method for the phase speed U is seemingly very accurate.

4.3 Discretization of the equation for the horizontal fluid velocity

Recall equation (2.40) from chapter 2.2

$$u = c_0 \cdot \left[\frac{1}{h_0}\eta - \frac{1}{4h_0^2}\eta^2 + h_0\eta_{xx}\left(\frac{1}{3} - \frac{z^2}{2h_0^2}\right)\right]$$

The equation for the horizontal fluid velocity $u(x,z,t)$. In order to implement this into a computer program, we need to discretize the term η_{xx}. Using the central differences formula for the second derivative as given in chapter 3.1.2:

$$\partial_x^2\eta = \frac{\eta_{j-1} - 2\cdot\eta_j + \eta_{j+1}}{(\Delta x)^2} \tag{4.6}$$

we get that the expression for the horizontal velocity profile for any chosen x-position on the spatial grid, with index j, will be

$$u_j = c_0 \cdot \left[\frac{1}{h_0}\eta_j - \frac{1}{4h_0^2}{\eta_j}^2 + h_0 \cdot \frac{\eta_{j-1} - 2\cdot\eta_j + \eta_{j+1}}{(\Delta x)^2} \cdot \left(\frac{1}{3} - \frac{z^2}{2h_0^2}\right)\right] \tag{4.24}$$

where $u_j = u_j\big(\eta_j(t), z\big)$ since the x-position is chosen.

Remember also from chapter 2.3 that the horizontal fluid velocity of importance will be found at the very top ($z = h_0 + \eta(x,t)$) of the leading wave. If, at a given time, the center of the leading wave is located at the x-grid position with index j, and we denote by η_j the height of the leading wave from the top of the undisturbed surface, then the expression for the horizontal fluid velocity at the top of the leading wave will be

$$u_j = c_0 \cdot \left[\frac{1}{h_0}\eta_j - \frac{1}{4h_0^2}{\eta_j}^2 + h_0 \cdot \frac{\eta_{j-1} - 2\cdot\eta_j + \eta_{j+1}}{(\Delta x)^2} \cdot \left(\frac{1}{3} - \frac{\left(h_0 + \eta_j\right)^2}{2h_0^2} \right) \right] \quad (4.25)$$

If we choose coefficients $g = 1$ and $h_0 = 1$, giving $c_0 = 1$, eq. (4.25) turns into

$$\eta_j - \frac{1}{4}{\eta_j}^2 + \frac{\eta_{j-1} - 2\cdot\eta_j + \eta_{j+1}}{(\Delta x)^2} \cdot \left(\frac{1}{3} - \frac{\left(1 + \eta_j\right)^2}{2} \right) \quad (4.26)$$

As for the time series of the leading wave phase velocity, some stuttering in the graph of the time series for u(t) may be observed when calculated as described above, although to a much smaller extent than for the phase speed. Therefore, Matlabs built-in smooth function, a moving-average calculation, has been applied to smooth out the curve of the horizontal fluid velocity u(t).

4.4 Summary of numerical study

The numerical study of the wave-breaking is best described by a brief summary of chapters 4.1 – 4.3:

1) The bore initial/boundary condition problem with the BBM-equation is solved by the methods of chapter 4.1.
2) As the approximative solution of $\eta(x,t)$ becomes gradually available, the bore front/leading wave phase speed U is calculated at all time-steps as described in chapter 4.2.
3) The horizontal fluid velocity $u(x, z = h_0 + \eta, t)$ at the top of the leading wave is calculated for each time-step as shown in chapter 4.3.
4) For each time-step, u is compared to U. If we have u > U at any time-step, we assume wave-breaking is starting to occur at that time-step.

4.5 Numerical considerations

4.5.1 Check of order/convergence speed

A convergence study has been done to confirm that the numerical scheme does indeed show second order convergence speed. The error is given by the L^2-error defined as:

$$Error = \left(\frac{L}{N+1} \cdot \max_n \sum_{j=0}^{N} \left| \eta_j^n - \eta(j\Delta x, n\Delta t) \right|^2 \right)^{\frac{1}{2}} \tag{4.27}$$

Like in [1], we choose a spatial domain $[0, L]$ where $L = 150$. We set $\Delta x = L/2^k$ and $\Delta t = \Delta x$, and increase the exponent k to half the steplengths. The results are shown in the figure below:

k	Error	Ratio
11	1.81E-03	
12	3.64E-04	4.96
13	8.03E-05	4.53
14	1.87E-05	4.29
15	4.52E-06	4.15
16	1.08E-06	4.19

Figure 4.3 – Results of convergence study for numerical scheme

We observe from Figure 4.3 that the error reduces by a factor of four when the spatial and temporal step sizes are halved, and conclude from this that the numerical scheme shows second order accuracy.

Note that the Matlab program "bbm_error_checker" used for the convergence study is included in Appendix B.

4.5.2 Stability considerations

Ideally, a formal stability analysis for the numerical scheme should have been done. Unfortunately, the time was not there to do this during the current work, other tasks being prioritized.

However, a practical demonstration of stability has been conducted. The assumption regarding stability is as usual for an explicit scheme for a PDE that

$$\Delta t_{max,stable} = f(\Delta x) \tag{4.28}$$

That is: The maximum stable temporal step-length Δt is dependent upon the chosen spatial step-length Δx, and so to ensure stability, for a chosen Δx we must choose for the simulations

$$\Delta t \leq \Delta t_{max,stable} \tag{4.29}$$

A series of simulations has been run, where the spatial step-length Δx has been kept constant while the time-step Δt has been gradually decreased. The results are presented in the figure below:

Case 1	Δx [m]	Δt [s]	Δt/Δx	
	0.04	0.01	1/4	
	U [m/s]	u [m/s]	η_{max} [m]	η_{max} x-pos. [m]
t = 20 s	1.236	0.608	0.474	81.58
t = 40 s	1.263	0.723	0.526	106.60

Case 2	Δx [m]	Δt [s]	Δt/Δx	
	0.04	0.001	1/40	
	U [m/s]	u [m/s]	η_{max} [m]	η_{max} x-pos. [m]
t = 20 s	1.236	0.609	0.474	81.58
t = 40 s	1.265	0.723	0.526	106.60

Case 3	Δx [m]	Δt [s]	Δt/Δx	
	0.04	0.0005	1/80	
	U [m/s]	u [m/s]	η_{max} [m]	η_{max} x-pos. [m]
t = 20 s	1.236	0.609	0.474	81.60
t = 40 s	1.264	0.723	0.526	106.60

Case 4	Δx [m]	Δt [s]	Δt/Δx	
	0.04	0.00025	1/160	
	U [m/s]	u [m/s]	η_{max} [m]	η_{max} x-pos. [m]
t = 20 s	1.236	0.609	0.475	81.58
t = 40 s	1.263	0.723	0.526	106.65

Case 5	Δx [m]	Δt [s]	Δt/Δx	
	0.04	0.0002	1/200	
	U [m/s]	u [m/s]	η_{max} [m]	η_{max} x-pos. [m]
t = 20 s	1.236	0.609	0.475	81.58
t = 40 s	1.263	0.723	0.526	106.65

Figure 4.4 - Practical demonstration of stability for numerical scheme. Here L = 150 m, h_0 = 1 m and a_0 = 0.3 m has been used, using a simulation time of t_{max} = 42 s for all runs.

We observe that the measured data are practically identical for all cases, and conclude from this that the highest time-step $\Delta t = 0.01$ combined with $\Delta x = 0.04$ is within the stability region of the numerical scheme.

5 RESULTS

5.1 The form of the BBM-equation used in simulations

In the simulations the form (2.4) of the BBM-equation has been used, where we have set $g = 1$ and $h_0 = 1$, also implying $c_0 = 1$, ref. below:

$$\eta_t + \eta_x + \frac{3}{2}\eta\eta_x - \frac{1}{6}\eta_{xxt} = 0$$

This has been done to be able to compare simulation results with the theoretical findings in chapter 3, where the same form of the equation was used for simplicity. Note however that the numerical simulation model allows for other choices of g and h_0 in equation (2.3).

5.2 General observations from simulations

5.2.1 Appearance of wave train with increasing number of waves with time

During a simulation run, the initial wave profile will always transform into a train of waves, travelling to the right, as shown in Figure 5.1 below. During a run, new waves will continuously form at the back of the wave train, so that the total number of waves is continuously increasing.

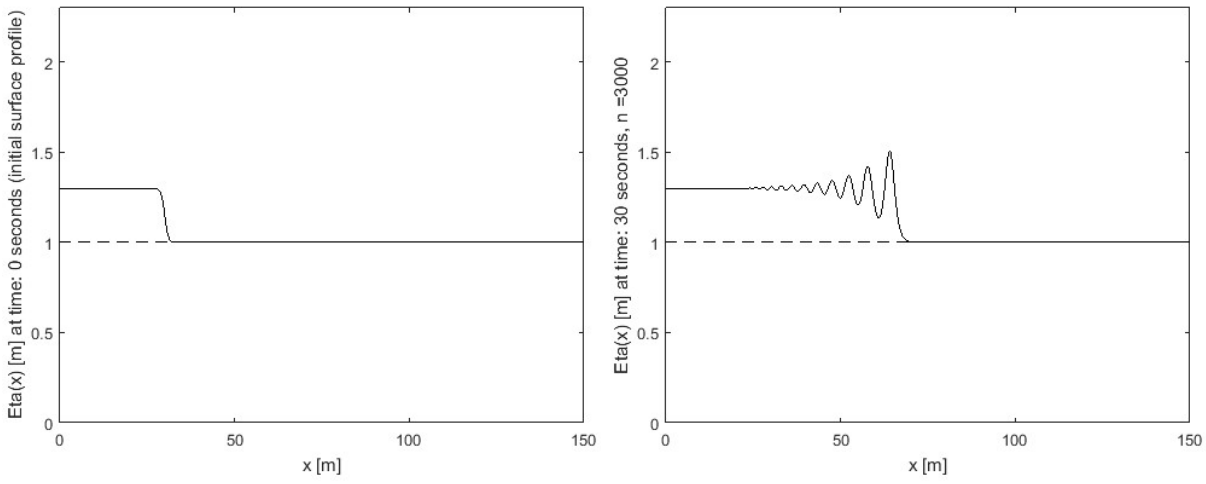

Figure 5.1 - Initial profile transforming into a right-travelling train of waves

5.2.2 Amplitudes increase with time and position to the right in wave train

The amplitudes of the waves in the wave train increase with both time (up to a certain point) and position to the right in the wave train, so the leading wave (the right-most) generally have the highest amplitude. An example of this is shown in Figure 5.2 below.

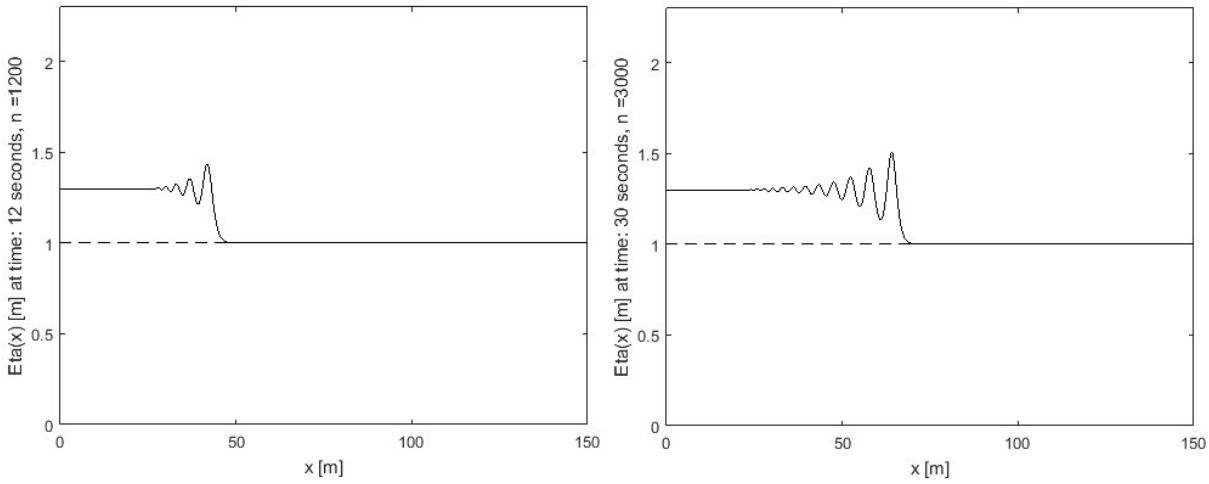

Figure 5.2 - Amplitudes of waves increase with time and to the right in the wave train

5.2.3 Amplitudes stabilize at a maximum and limit horizontal fluid velocity

As time passes during a simulation, giving that the bore strength α is such that wave breaking does not occur, the amplitude of the leading wave (which is the largest overall amplitude) stabilizes towards a maximum. An example is shown in Figure 5.3 below.

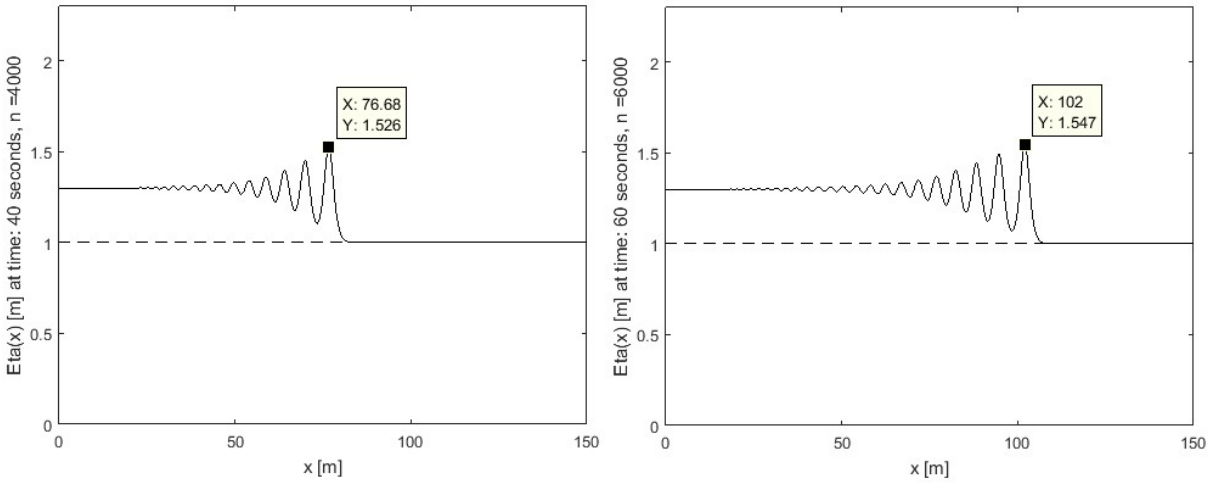

Figure 5.3 - Maximum amplitude stabilizing

By studying time series of the maximum amplitude versus the calculated horizontal fluid velocity at the top of the leading wave, one easily observes that the horizontal fluid velocity apparently is bounded by the amplitude of the wave, which should not come as any surprise when studying equations (4.24), (4.25) and (4.26), which clearly show the dependence of the horizontal fluid velocity on the elevation z.

Below is an example of time series for the maximum amplitude eta [m] (left) and the horizontal fluid velocity u [m/s] at the top of the leading wave for a simulation run (right). Observe how closely the (dotted) graph of u follows the graph of the maximum amplitude eta. Note also the calculated leading wave phase speed U [m/s] which as mentioned in chapters 2 and 4 is fairly constant. Further, U is also larger than u for all times in this series, showing that there is no wave-breaking occurring in this simulation run.

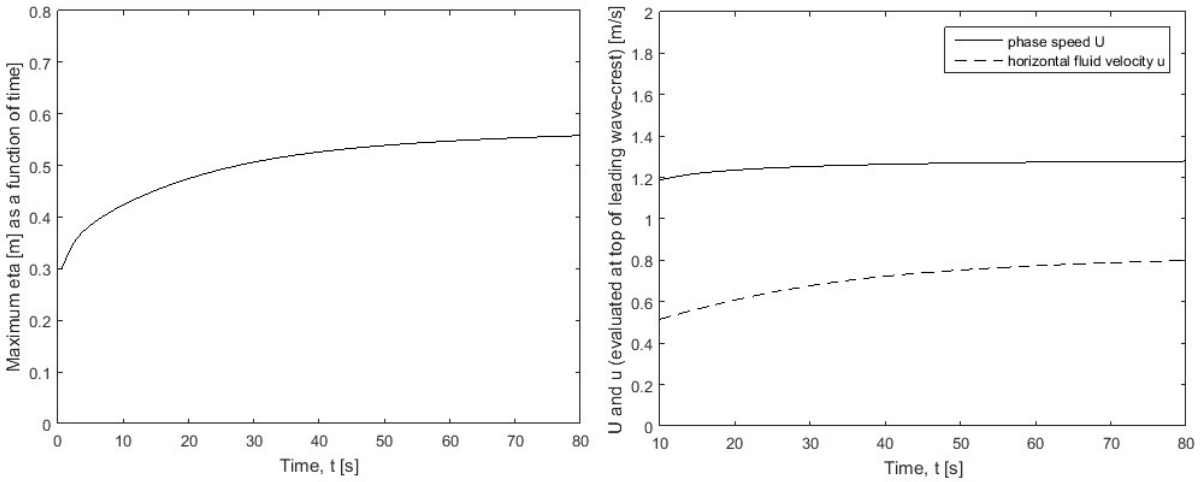

Figure 5.4 - Horizontal fluid velocity dependence on maximum wave amplitude

From the above, we can note an observation: If the horizontal fluid velocity u evaluated at the top of the leading wave has not exceeded the phase speed U of the leading wave before the leading wave reaches its maximum amplitude, then no wave-breaking will occur.

5.2.4 Wave-breaking is observed to depend on chosen value of alpha

As expected, the value $\alpha = a_0/h_0$ clearly influences whether wave-breaking is observed during a simulation. After some simulation runs with the finished program, it quickly became clear that there is some kind of wave-breaking threshold for α-values in the area of approximately 0.39-0.40. This will be elaborated in chapter 5.3. In the figure below, two simulation runs are shown, differing only in the value for a_0. In the left plot, a_0 is set to 0.35 m, while for the right we have a_0 set to 0.45. With h_0 = 1 m in both cases, this corresponds of course to α = 0.35 and α = 0.45, respectively. As has been noted earlier in chapter 2.3, after wave-breaking occurs, the the BBM-equation (2.3) is no longer valid as a description of the physical water surface.

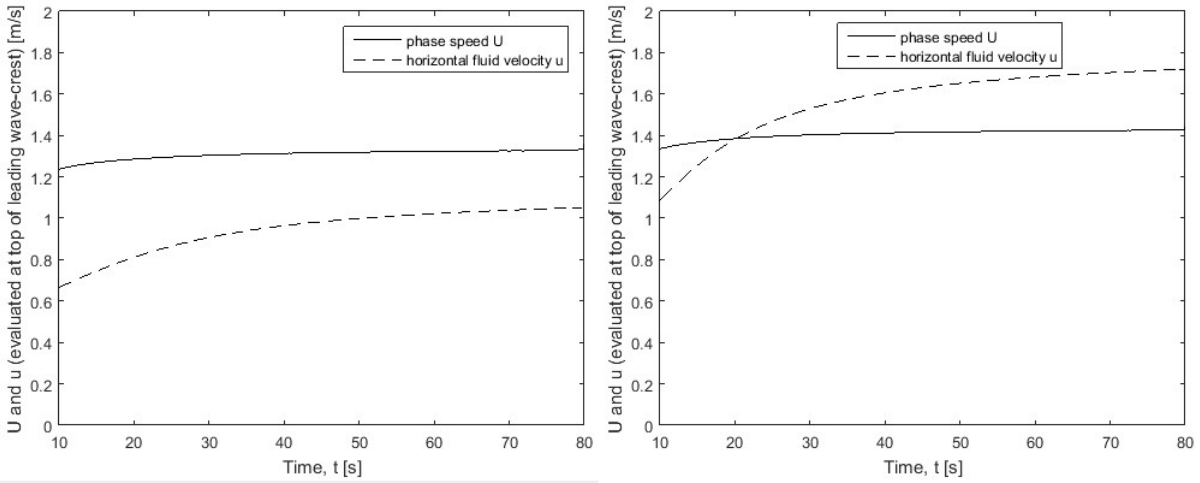

Figure 5.5 - Two simulation runs, with no wave-breaking and wave-breaking. To the left: Alpha = 0.35, no wave-breaking is observed. To the right: Alpha = 0.45, wave-breaking occurs after around 20 seconds.

5.3 Observed critical alpha for wave-breaking

The value of $\alpha_{critical}$ for which the onset of wave-breaking is observed for $\alpha > \alpha_{critical}$ was found by Favre in [2] to be 0.28, by Kalisch/Bjørkavåg in [1] to be 0.379, by Brun in [3] to be 0.358, and finally by Kalisch/Brun to be 0.3238 in [11].

In [3], Brun chooses to start out with the value $\alpha_{critical}$ = 0.28 found by Favre, and to gradually increase the bore strength to find $\alpha_{critical}$ for the KdV-system. However, during the work on this report, it quickly became clear from simulation runs that a possible $\alpha_{critical}$ in this case would be somewhat higher than 0.28. Therefore the starting point in the search for $\alpha_{critical}$ was set to α = 0.39.

In [3], the choice is also made to use a combination of temporal step-length Δt and a maximum number of time steps which results in a final time of 120 seconds for the simulation runs, on the grounds of keeping the accumulating error from becoming too big. Although this seems somewhat arbitrary, for the ease of comparison a final time of 120 seconds for simulation runs was chosen also in the current work. Note that allowing a longer time period for simulation would most likely result in a slightly lower value for $\alpha_{critical}$.

Using step-lengths Δx = 0.04 and Δt = 0.01, the case with h_0 = 1 m and a_0 = 0.39 m giving the desired starting value of α = 0.39 was chosen as the starting point for investigation. The figure below shows that no wave-breaking was found for this case.

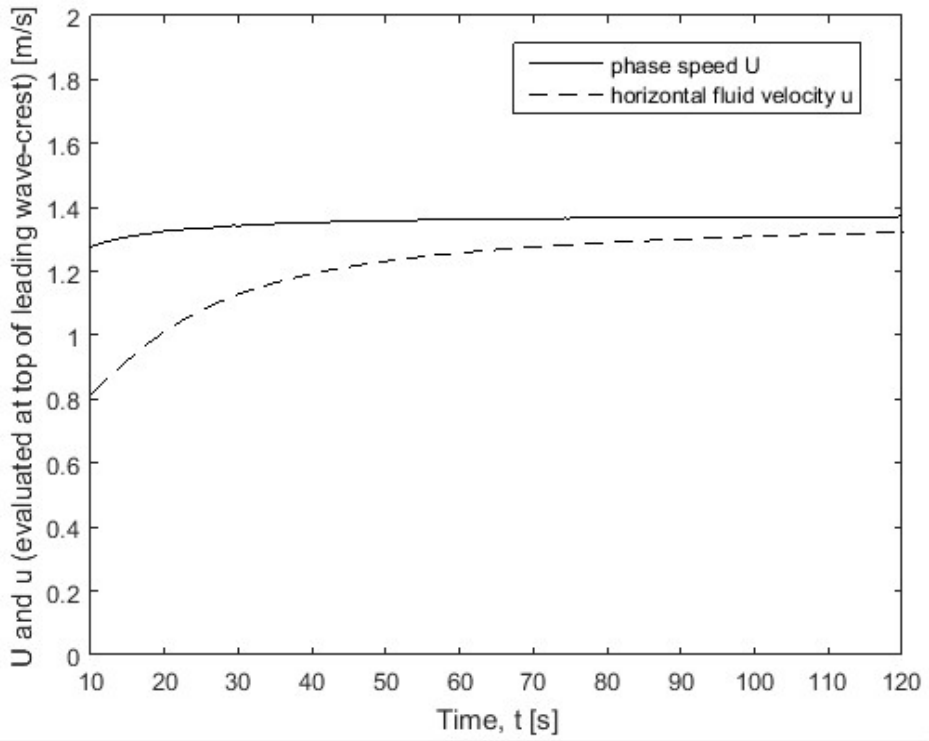

Figure 5.6 – Starting point alpha = 0.39 in the search for critical alpha. No wave-breaking observed for this case.

Running several simulations, increasing a_0 and thereby also α between each run, the general value-range of α for the appearance of wave-breaking was identified. Applying an increment of 0.001 for a_0 in this area, the onset of wave-breaking was first observed for α = 0.399, as shown in the figure below.

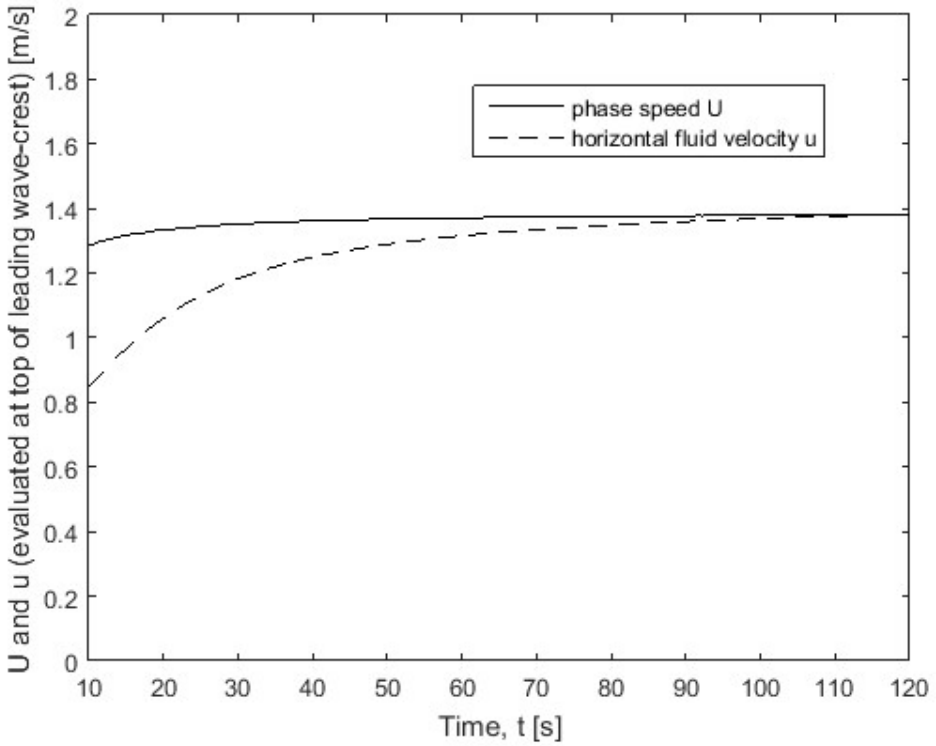

Figure 5.7 – Critical alpha for wave-breaking. Wave breaking occuring for alpha = 0.399

The main finding in the current work is then a critical bore-strength $\alpha_{critical}$ = 0.399 for the onset of wave-breaking in the initial/boundary value bore-problem with the BBM-equation.

It is about twice the experimental value found by Favre in [2], but we must keep in mind that Favre did his experiments in an outdoor wave tank, where all kinds of real-world physical effects would have affected the results. The critical value for α found in the current work does not differ much from values found in other theoretical works, especially it should be noted how close the value found here is to the value $\alpha_{critical}$ = 0.379 obtained by Kalisch/Bjørkavåg in [1].

5.4 Bore strength α versus maximum wave height and breaking times

By running simulations with bore strength α higher than the found critical value, we can observe the dependence of the maximum achieved wave height and the time of wave breaking during a simulation, upon the bore strength α. Using the same calculation method as before, a number of simulations have been run until breaking is occurring, and then the height of the leading wave at that point as well as the time of breaking has been noted. The data is presented in the following figures.

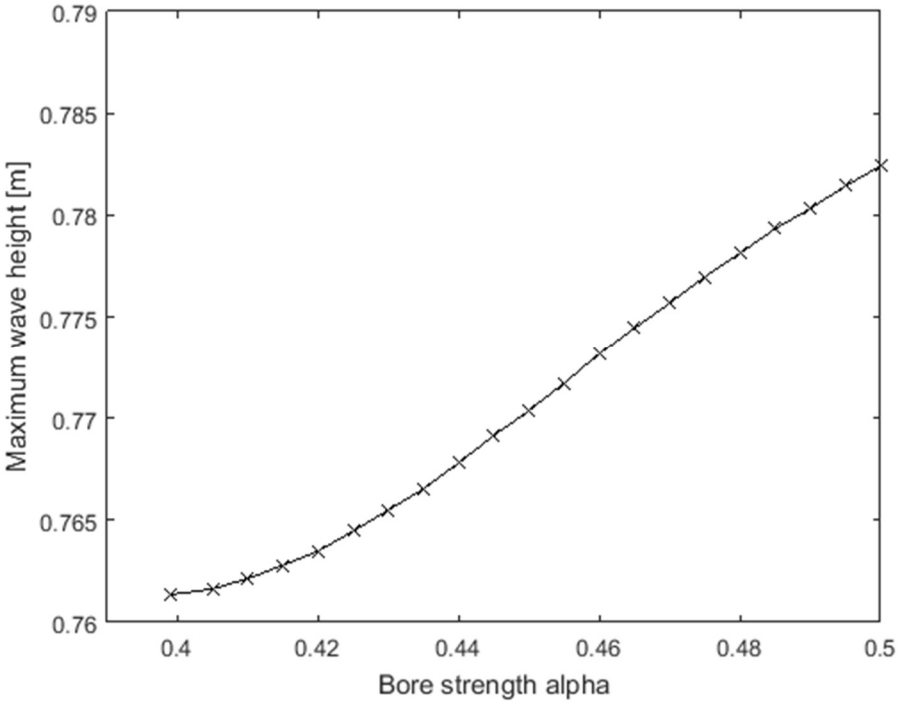

Figure 5.8 - Maximum wave height at breaking point as a function of bore strength alpha

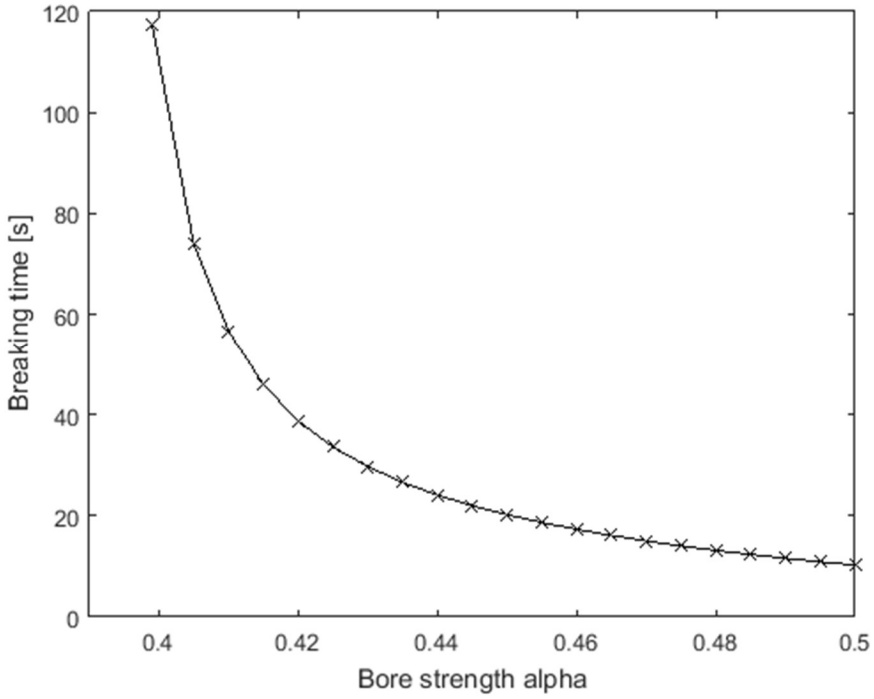

Figure 5.9 - Breaking time as a function of bore strength alpha

We observe that a higher bore strength will result in a higher achieved maximum wave height, as well as shorter times before breaking occurs.

Interesting to note here is the apparent connection to the result found for the theoretical maximum wave height of the solitary wave found in chapter 3.2, that is

$$H_{MAX} = 0.7604389$$

For simulations with bore strength α close to the critical value $\alpha_{critical} = 0.399$, it seems that the maximum wave height at the time of breaking will approach this value. This result is similar to what was found in [3], and we expect also here that allowing for a longer time period in the search for the critical value of alpha could give a slightly lower $\alpha_{critical}$, resulting in a wave height even closer to the theoretical maximum at the point of breaking.

In [3], theoretical work given in [4] is used to explain this result. The idea is that over time, the leading wave of the bore will approach a solitary wave. In [1], the authors point to a general taking in the literature in which it is assumed that bore of the undular type will transform into a train of solitary waves in models where viscosity is not included. The results of the current work do indeed point strongly in this direction. From simulations using the bore strength $\alpha_{critical} = 0.399$, it can be seen that at the point of wave-breaking we do not only have a leading wave height very close to the theoretical maximum, but we also have a phase speed for the leading wave that comes very close to the speed obtained by inserting the maximum wave height into the wave height-phase speed relation given by eq. (3.9).

Apparently, the bore strength $\alpha_{critical} = 0.399$ represents a sort of threshold, for which higher values of α results in the breaking of waves before solitary waves have had the time to fully develop.

6 CONCLUSIONS AND FURTHER WORK

The travelling bore is a physical fluid phenomenon described in a simplified way as a fluid flow moving along on top of another body of fluid beneath.

In the 1930s, experimental studies of the travelling bore were conducted in an outdoor wave tank by Favre [2]. He found that the physical property bore strength, given as $\alpha = a_0/h_0$, the ratio of the height of the incoming water flow to the height of the undisturbed water, had an important impact on the onset of wave-breaking for waves in the bore. Most relevant for the current work, he found that a bore strength of $\alpha_{critical}$ = 0.28 constituted a threshold for wave-breaking, for which higher values of α would result in the occurrence of wave-breaking in otherwise purely undular bores.

Several theoretical studies [1] [3] [11] have been done to investigate the experimental results found by Favre. Using a Boussinesq system to model the surface waves, Kalisch and Bjørkavåg [1] found the wave-breaking threshold to be $\alpha_{critical}$ = 0.379. Brun used the KdV equation and found an $\alpha_{critical}$ = 0.358 in [3], and also with the KdV equation Kalisch and Brun found $\alpha_{critical}$ = 0.3238 in [11].

In the current work, the BBM equation as given in [4] has been used to model the surface waves in the bore.

Based on the work in [1], a second order numerical scheme using finite differences for spatial discretization and Heuns method for time-stepping has been developed. Using appropriate boundary conditions, the numerical scheme has been used to solve the initial/boundary value problem for an approximative solution of the surface profile $\eta(x,t)$. A study on convergence speed confirms that the numerical scheme displays second order behaviour, and a practical demonstration of stability for commonly used spatial and temporal step-size combinations has been done.

After the solution of $\eta(x,t)$ is available, the phase speed U of the leading wave can be found from simple calculations, as its position is then known for all time. Then, by calculating the horizontal fluid velocity u at the top of the leading wave, the onset of wave-breaking has been investigating using the breaking criterion u > U.

The threshold for the onset of wave-breaking has in the current work been found to be $\alpha_{critical}$ = 0.399. This somewhat higher than the value found by Favre, but it corresponds well to the values found in other theoretical works. However, it should be noted that Favre conducted his experiments in an outdoor wave tank where all kinds of real world physical effects surely influenced the result, while the current work involves a theoretical model with a lot of simplifications.

The results also indicate that for bore-strengths in the area of the critical value, the leading wave seems to develop into a solitary wave with a maximum height close to the theoretical maximum value found by using the breaking criterion on the analytical solitary wave solution. Seemingly, the critical value of α represents an upper threshold where for lower bore-strengths the undular bore develops into a train of solitary waves, while for higher bore-strengths wave breaking occurs before this can happen.

6 - **CONCLUSIONS AND FURTHER WORK**

There are several possible actions to take which may result in a better match between theoretical and experimental results. In [12] the effect of an imposed background vorticity on a flow described by the KdV equation is investigated, showing a clear influence on wave breaking. Another option could be to use higher order Boussinesq systems to describe the surface waves more accurately, e.g. one of the systems described in [13].

With the above in mind, it would be interesting to see the results of possible further work which could include studies with more advanced theoretical models where physical effects are accounted for. Another task for future study could be to look more closely at the difference in the results found in the current work for wave breaking in the BBM equation and the results found for the related KdV equation in [3].

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
% BBM Solver
% Program to solve the BBM equation (as given in "Linear and nonlinear
% waves" by G.B Whitham, p. 463) numerically with Heuns method for
% time-stepping, applied to the physical problem of a bore with height a0
% flowing over undisturbed water with height h0.

% The program also calculates the bore front speed/speed of the
% leading wave, U [m/s], which is calculated from the approximative
% solution for eta(x,t), and also calculates the horizontal fluid velocity
% u(x, z = h0 + eta_max, t) [m/s] at the top of the leading wave-crest at
% all time-steps, to be able to observe the onset of wave breaking
% according to the breaking criterion u > U.

% Output:
% i)   Approximative solution of the surface profile eta(x,t) on an
%      (x,t)-grid (so dimensions are 1 spatial + 1 temporal) for all
%      time-steps.
% ii)  Calculated values of bore front/leading wave-crest
%      phase speed U [m/s] for all time steps.
% iii) Calculated values of horizontal fluid velocity
%      u(x, z = h0 + eta_max, t) [m/s] at the top of the leading wave-crest
%      for all time steps.

% Results displayed in plots:
% i)   Surface-profile eta(x) for a user-controlled set of t-values
% ii)  Horizontal fluid velocity profile underneath leading wave-crest
%      for a user-controlled set of t-values.
% iii) Time series of U and u in the same plot for visual control
%      of eventual onset of wave-breaking.
% iv)  Time series of eta_max, the maximum wave-height in the domain

%=========================================================================

% NB! Remember to clear the memory between each run of the program

% Use command "close all" to close all figures from a simulation

% The program contains a lot of "greened-out" code pieces used during the
% development of the program, they are intentionally left in the program.
% They can be useful in many ways, e.g. if you want the value of a variable
% printed to the screen at a certain time/point in the program

% Elaborate explanations of code pieces are intentionally left in the
% program, hopefully to easy the use of the program for any new users.

% Last review/update of code: 20.12.18

%=========================================================================
% MISCELLANOUS
%=========================================================================

clear  % Clearing workspace, freeing up system memory

tic  % Starts timer for program run
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
% Below: Set plot outline color to white, as it works better when copying
% plots into documents. Code from
% https://www.mathworks.com/help/matlab/creating_plots/
% default-property-values.html, checked 18.10.18, 11:32

set(groot,'defaultFigureColor','w')

%=========================================================================
% INPUT PARAMETERS
%=========================================================================

a0 = 0.399;      % Height of bore (ABOVE undisturbed water depth) [m]

h0 = 1;          % Undisturbed water depth [m]

L = 300;         % Length of spatial domain (in x-direction) [m]
                 % Recommended to set this to an even number

b_start_pos = 5;  % Divisor controlling the starting position of the
                  % bore-front in the spatial interval. bore_start_pos = 2
                  % corresponds to bore-front initial position in the
                  % middle of the spatial interval

n_max = 13000;   % The desired maximum number of steps to take forward in
                 % time (Preferably this number should be divisible by
                 % rem-control number for plots of surface profile)

kappa = 1;       % Controls steepness of the initial surface profile in the
                 % transition from a0 to 0.
                 % Higher value = steeper bore front slope, and vice versa
                 % Note: Changing this value will affect the result for the
                 % approximative solution of eta(x,t) somewhat, but will
                 % apparently not have an effect on the occurence of
                 % wave-breaking depending on alpha = a0/h0

U_avg_steps = 100;% Averaging to smooth out "main" stuttering in the graph
                  % of the time series U(t) for the leading wave
                  % phase speed U [m/s]. U_avg_steps is the number
                  % of time steps to average leading wave phase speed U
                  % [m/s] over. U must be averaged in order to get a smooth
                  % curve U(t), as it will be "stuttering" when calculated
                  % over individual time steps, due to the discrete grid.
                  % NB! Set U_avg_steps = 1 to calculate U over individual
                  % time-steps

smooth_span_U = 501; % Secondary smoothing of the graph of the time series
                     % U(t) for the leading wave-crest phase speed U [m/s].
                     % Sets the span of the moving average calculation
                     % employed in Matlabs "smooth"-function, see Matlab
                     % documentation for details

smooth_span_u = 501; % Smoothing of the graph of the time series of
                     % the horizontal fluid velocity u [m/s] evaluated
                     % at the top of the leading wave.

%=========================================================================
% CONSTANTS GIVEN BY INPUT PARAMETERS
%=========================================================================
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
g = 1;            % Gravitational acceleration [m/(s^2)]

c0 = sqrt(g*h0); % Phase speed for linear surface waves in the
                 % shallow water approximation [m/s].
                 % See Whitham p. 454/KCD eq. (8.49) p. 362

gamma = (1/6)*c0*(h0^2); % Constant given in eq. (13.94),
                         % Whitham p. 462 [m^3/s]

b = (3/2)*(c0/h0);       % Constant for simplification

d  = gamma/c0;           % Constant for simplification

%=========================================================================
% SPATIAL AND TEMPORAL STEP SIZES, X-VECTOR
%=========================================================================

% NB! As per 20.12.18: LARGEST KNOWN stable delta_x and delta_t:

% delta_x = = 0.04 delta_t = delta_x/4 = 0.01

% UPDATE here if larger stable step-sizes are found
%=========================================================================

delta_x = 0.04;           % Steplength in spatial direction (x-dir.)
                          % NB! Choose delta_x so that L is divisible
                          % by delta_x (required for an integer number of
                          % uniform steps delta_x to exactly fill all of L)
                          % Easily checked by L/delta_x in command window

%delta_x = L/(2^k);       % Alternate formula for spatial steplength.
                          % Note that this method ENSURES that
                          % L is divisible by delta_x, as then we have
                          % L/delta_x = 2^k, which is always an integer

delta_t = 0.01;           % Temporal steplength (timestep)

N = (L/delta_x) - 1;      % The number of "internal" spatial grid points.
                          % Controls the "inner" size of vectors and the
                          % size of matrices,
                          % which will be N and N x N, respectively
                          % Follows logically, and also from
                          % delta_x = L/(2^k) set equal delta_x = L/(1+N)
                          % Note that x_l and x_r come in addition,
                          % so the total number of grid points
                          % on the spatial domain is N + 2

x = zeros(N+2,1);         % Creates a (N+2) x 1 vector x sized to hold all
                          % x-points (all spatial grid points)

for j = 1:length(x)       % Updates vector x with the correct
    x(j) = (j-1)*delta_x; % x-values for the grid points
end

x;
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
%=========================================================
% ETA-VECTOR
%=========================================================

eta = zeros(N+2,1);     % Creates a (N+2 x 1) vector sized to hold
                        % eta-values for all x-points

%=========================================================================
% ETA STAR-VECTOR
%=========================================================================

% Creates a N x 1 vector sized to hold eta_star-values for all
% "internal" x-points (excluding x_l and x_r).
% eta_star values are the Euler predictions (when at the current time
% t = n*delta_t) for the eta-values at time t = (n+1)*delta_t

eta_star = zeros(N+2,1);

eta_star(1) = a0;       % First element of eta_star vector should be
eta_star;               % the LHS boundary condition for eta

%=========================================================================
% ETA (N+1)-VECTOR
%=========================================================================

% Creates a N x 1 vector sized to hold eta_n_plus_1-values for all
% "internal" x-points (excluding x_l and x_r).
% eta_n_plus_1 values are the new solution points (when at the current
% time t = n*delta_t) for the eta-values at time t = (n+1)*delta_t

eta_n_plus_1 = zeros(N+2,1);

eta_n_plus_1(1) = a0;     % First element of eta_n_plus_1 vector should
eta_n_plus_1;             % be the LHS boundary condition for eta

%=========================================================================
% ETA STORAGE-MATRIX
%=========================================================================

% Creates a matrix to save all eta-values for all time steps
% One column in eta_storage will contain all eta_values for a given
% time-step. The first column vector represents eta(x,0), the initial
% surface profile

eta_storage = zeros(N+2, n_max+1);

% Below: Taking LHS boundary conditions into eta_storage matrix for all
% time. The RHS boundary condition is already 0 from the creation of the
% matrix.

for i = 1:n_max+1
    eta_storage(1,i) = a0;
    %eta_storage(N+2,i) = 0;
end

eta_storage;

%[m,n] = size(eta_storage)
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
%=========================================================================
% TRIDIAGONAL MATRIX T
%=========================================================================

% Constructing tridiagonal matrix T to be used in calculations of matrix A
% Using sparse matrix to save memory during simulations.
% Code from
% www.mathworks.com/help/matlab/math/constructing-sparse-matrices.html
% found 02.10.18, 19:20

D = sparse(1:N,1:N,-2*ones(1,N),N,N);
E = sparse(2:N,1:N-1,ones(1,N-1),N,N);
S = E+D+E';

T = S;

%T = full(S);

%=========================================================================
% A-MATRIX
%=========================================================================

% Constructing matrix A to be used in the calculation of linear systems of
% equations with solutions respectively eta_star vector and "eta_(n+1)"
% vector
% Note: A should be a tridiagonal matrix, as it is a summation of a
% diagonal matrix I and a tridiagonal matrix T. It should also be
% sparse, as both I and T are created as sparse.
% Note that using sparse matrices is crucial for simulation speed!
% For a certain reference problem used for testing, this step reduced
% simulation time usage by 97 %

% Below: Creating sparse identity matrix with dimensions (N x N)

I = sparse(1:N,1:N,ones(1,N),N,N);

% Below: Creating matrix A (N x N).

A = (1/delta_t)*I - (d/(((delta_x)^2)*delta_t))*T;

%=========================================================================
% B1-VECTOR
%=========================================================================

b1 = zeros(N,1);     % RHS vector (N x 1) in the system of equations
                     % A*eta_star_calc = b1, the system
                     % which is to be solved for eta_star_calc (Euler pred.)

%=========================================================================
% B2-VECTOR
%=========================================================================

b2 = zeros(N,1);     % RHS vector (N x 1) in the system of equations
                     % A*eta_n_plus_1_calc = b2, the system
                     % which is to be solved for eta_n_plus_1_calc
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
%=========================================================================
% TIME OVERVIEW-VECTOR
%=========================================================================

t_overview = zeros(n_max+1,1);      % Creating a vector containing all
                                    % time values for control purposes

%=========================================================================
% ETA MAX STORAGE-VECTOR
%=========================================================================

eta_max_storage = zeros(n_max+1,1); % Creating a vector for storing the
                                     % maximum value of eta for each
                                     % time step

%=========================================================================
% ETA MAX X-INDEX STORAGE-VECTOR
%=========================================================================

% Creating a vector for storing the x-index for the maximum value
% of eta for each time step

eta_max_x_index_storage = zeros(n_max+1,1);

%=========================================================================
% ETA MAX X-POSITION STORAGE-VECTOR
%=========================================================================

% Creating a vector meant to store the x-position for the maximum value
% of eta for each time step

eta_max_x_position_storage = zeros(n_max+1,1);

%=========================================================================
% U STORAGE-VECTOR
%=========================================================================

% Vector to hold calculated (with averaging) bore front/leading wave-crest
% velocities U for all time steps

U_storage = zeros(n_max+1,1);

%=========================================================================
% U STORAGE_SMOOTHED-VECTOR
%=========================================================================

% Vector to hold calculated values (with averaging AND smoothing) for
% bore front/leading wave-crest velocities U for all time steps

U_storage_smoothed = zeros(n_max+1,1);

%=========================================================================
% u_eta_max STORAGE-VECTOR
%=========================================================================

% Vector to hold calculated horizontal fluid velocity u(x,z,t)
% evaluated at x = x(eta_max), z = h0 + eta_max, t = n*delta_t,
% for all time steps.
```

```
% Note that the wave breaking criterion is u > U.

u_eta_max_storage = zeros(n_max+1,1);

%=========================================================================
% u_eta_max STORAGE_SMOOTHED-VECTOR (OK! Checked 30.11.18)
%=========================================================================

% Vector to hold calculated values (with smoothing) for horizontal fluid
% velocity u(x,z,t) evaluated at x = x(eta_max), z = h0 + eta_max,
% t = n*delta_t, for all time steps

u_eta_max_storage_smoothed = zeros(n_max+1,1);

%=========================================================================
% ADVANCING SOLUTION VECTOR ETA FORWARD IN TIME
%=========================================================================

n = 0;      % Time step index initially at zero

t = 0;      % Time initially at zero

% Below: Updates vector eta to hold the initial condition/
% initial surface profile eta(x,0).

for j = 1:(N+2)
    eta(j) = 0.5*a0*(1 - tanh(kappa*(x(j)-(L/b_start_pos))));
end

% NB! Note that L/2 above places the borefront at t = 0 (initially) in the
% middle of the spatial interval [0,L]. This can be adjusted as desired
% with input parameter b_start_pos.

% Note that we are currently letting the initial condition function control
% the value of eta for the LHS and RHS grid points at t = 0, which could
% possibly have been "overridden"/set to a0 and 0, respectively. Code
% a few lines below to implement this, if desired.

eta;

% Below: Storing the initial (at n=0 / t=0) maximum value of eta(x,t=0)
% as well as the index of the maximum eta value in the eta vector.
% As this index is the same for the max eta value in the eta vector and
% the corresponding x-value in the x vector, we use this to find the
% x-position of the maximum eta value for all time steps.
% Note that this is really not interesting at this point,
% it is at a later time, when a front wave (which will have the
% maximum eta value) has appeared, that we need the
% x-positions of this front wave at different times, to calculate
% the phase speed U [m/s] for the bore front

[eta_max_storage(n+1), eta_max_x_index_storage(n+1)] = max(eta);

eta_max_x_position_storage(n+1) = x(eta_max_x_index_storage(n+1));

% Below: Plotting initial surface profile eta(x,0)
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
figure
plot(x,eta + h0,'-black');
hold on
plot(x,ones(size(x)) * h0,'--black');  % Plot undisturbed water height h0
xlabel('x [m]');
ylabel(['Eta(x) [m] at time: ' num2str(t) ...
            ' seconds (initial surface profile)']);
xlim([0 L])
ylim([0 ((a0 + h0)+1)])
hold off

% Below: Creating (N x 1) vector r_eta_n to be used in calculations

r_eta_n = zeros(N,1);

% Below: Creating (N x 1) vector r_eta_star to be used in calculations

r_eta_star = zeros(N,1);

% Below: While-loop advancing solution forward in time starts here!

while n < n_max     % Running calculations until desired maximum number
                    % of time steps is reached

    % Below: Adding current time for storage in time overview vector.
    % Note that as we start at n = 0, the first element in time overview
    % vector to be updated is element 1, where we store t = 0.

    t_overview(n+1) = t;

    % Below: Storing surface profiles eta(x,t) in the eta_storage matrix
    % for all time steps.
    % We only need to update "internal" values, first and last
    % elements are the BCs a0 and 0 for all time.
    % The first run of the while loop (for n = 0) will store the
    % initial surface profile eta(x,0) as the first column vector
    % in the eta_storage matrix.

    for i = 2:N+1
        eta_storage(i,(n+1)) = eta(i);
    end

    % Below: Calculating current r_eta_n vector, which is a function of
    % the current eta vector, as well as spatial steplength delta_x
    % and the constants b and c0

    for j = 2:N+1 % Starting at j=2 due to Matlab array indexing method
        r_eta_n(j-1) = -(b/2) * (((eta(j+1))^2 - ...
                        (eta(j-1))^2)/(2*delta_x))...
                        - c0*((eta(j+1) - eta(j-1))/(2*delta_x));
    end

    r_eta_n;

    % Below: Creating (N x 1) vector eta_calc to use in linear system of
    % equations, as the vector dimensions must be (N x 1) when solving
    % the linear system
```

```
eta_calc = eta;

eta_calc(1) = [];
eta_calc(length(eta_calc)) = [];

%eta;
%eta_calc;

% Below: Creating (N x 1) vector eta_star_calc to use in linear system
% of equations, as the vector dimensions must be (N x 1) when solving
% the linear system.

eta_star_calc = zeros(N,1);

% Below: Calculating/updating b1 vector

b1 = r_eta_n + (1/delta_t)*eta_calc - (d/((delta_x^2)*delta_t)) ...
        * T * eta_calc;

%b1;

% Below: Solving linear system of equations for Euler prediction
% vector eta_star_calc

eta_star_calc = A\b1;

% Below: Adding current "internal" eta_star values from vector
% eta_star_calc to eta_star vector
% First and last elements of eta_star, corresponding to indices i=1
% and i=N+2 respectively, remains the boundary conditions: a0 and 0

for i = 2:N+1
    eta_star(i) = eta_star_calc(i-1);
end

%eta_star_calc;
%length(eta_star_calc);
%eta;
%eta_star;
%length(eta_star);

% Below: Calculating current r_eta_star vector, which is a function of
% the current eta_star vector, as well as spatial steplength delta_x
% and the constants b and c0

for j = 2:N+1 % Starting at j=2 due to Matlab array indexing method
    r_eta_star(j-1) = -(b/2) * (((eta_star(j+1))^2 - ...
                      (eta_star(j-1))^2)/(2*delta_x))...
                    - c0*((eta_star(j+1) - eta_star(j-1))/(2*delta_x));
end

% Below: Calculating/updating b2 vector

b2 = (1/2)*(r_eta_n + r_eta_star) + (1/delta_t)*eta_calc - ...
        (d/((delta_x^2)*delta_t)) * T * eta_calc;

% b2;
```

```
% Below: Creating (N x 1) vector eta_n_plus_1_calc to use in linear
% system of equations, as the vector dimensions must be (N x 1) when
% solving the linear system.

eta_n_plus_1_calc = zeros(N,1);

% Below: Solving linear system of equations for eta_n_plus_1_calc
% vector, the solution for the surface profile one step forward
% in time

eta_n_plus_1_calc = A\b2;

% Below: Adding current "internal" eta_n_plus_1 values from vector
% eta_n_plus_1_calc to eta_n_plus_1 vector
% First and last elements of eta_n_plus_1, corresponding to indices i=1
% and i=N+2 respectively, remains the boundary conditions: a0 and 0

for i = 2:N+1
    eta_n_plus_1(i) = eta_n_plus_1_calc(i-1);
end

%eta;           % Approx. solution for eta(x,t) at current time t
%eta_n_plus_1;  % Approx. sol. for eta(x,t+delta_t) at time t + delta_t

n = n + 1       % Updating time step number for next run of while-loop
t = n*delta_t;  % Updating the time for the next run of the while-loop

eta = eta_n_plus_1; % Updating eta vector to be ready to repeat
                    % while-loop for next time step

% Below: Storing the current maximum value of eta(x,t)
% as well as the index of the maximum eta value in the eta vector.
% As this index is the same for the max eta value in the eta vector and
% the corresponding x-value in the x vector, we use this to find the
% x-position of the maximum eta value for all time steps.
% At the time when a front wave (which will have the maximum eta value)
% has appeared, we need the x-positions of this front wave at different
% times, to calculate the phase speed U [m/s] for the bore front

[eta_max_storage(n+1), eta_max_x_index_storage(n+1)] = max(eta);

% Below: The max(vector) function in matlab will pick the first
% (leftmost) element in the vector, if there are more than one maximum
% value in the vector! In the start of the simulation, this can
% typically result in the max(eta)-code part picking the very first
% element in eta-vector, with index 1, as the maximum. This is because
% the initial surface profile has a constant value a0 as the
% maximum eta-value at its left part, so there are many max values.
% This will result in an error message further down in the code, as the
% code for calculating the horizontal velocity profile below the
% maximum eta-value employs finite differences for the second order
% derivative, and will then try to pick element 0 from the eta-vector,
% which will not work since matlab starts indexing at element 1.
% Note that this little code piece is only used if the maximum eta
% value actually has index value 1 in the eta-vector, so it will
% have no effect on the when we are actually interested in the max
% eta-value, that is after many time steps, when a leading wave-crest
% has formed.
```

```
%eta_max_x_index_storage(n+1);

if eta_max_x_index_storage(n+1) == 1
    eta_max_x_index_storage(n+1) = eta_max_x_index_storage(n+1) + 1;
end

%eta_max_x_index_storage(n+1);

eta_max_x_position_storage(n+1) = x(eta_max_x_index_storage(n+1));

%eta_max_storage(n+1);
%max(eta);

% Below: Calculating bore front/leading wave-crest speed U [m/s]
% from a simple "v = s/t" formula.

% Note that by calculating over individual time steps, the U-value we
% find is actually the AVERAGE bore front speed
% between two time index values

% U_avg_steps is the number of time-steps to average U over. This
% takes care of the main "stuttering" in the graph of U(t), but it
% still has to be smoothed using Matlabs "smooth"-function, which is
% a moving average filter.

if n >= U_avg_steps
    U = (eta_max_x_position_storage(n+1)...
            - eta_max_x_position_storage((n+1)-U_avg_steps))...
                /(U_avg_steps*delta_t);

    % Below: Storing current value of U [m/s] in the U_storage vector

    U_storage(n+1) = U;

    % Below: U_storage_smoothed vector is updated in every time-step,
    % and will eventually serve as the values used for the leading
    % wave-crest phase speed U [m/s] in the calculations

    U_storage_smoothed = smooth(U_storage,smooth_span_U);
end

% Below: Creating a vector of z-values, at which the horizontal fluid
% velocity u [m/s] "on the vertical line" below the maximum eta-value
% will be calculated, thus enabling us to plot a profile of the
% horizontal fluid velocity at different depths below the maximum
% eta-value.

delta_z = (h0 + max(eta))/10;

z = zeros(11,1);

for i = 1:11
    z(i) = (i-1)*delta_z;
end

%z;

u_profile = zeros(11,1);
```

```
% Below: Using the equation for u(x,z,t) to calculate the profile for
% the horizontal fluid velocity. Using discretized equation
% derived from Whitham, p. 465.
% Note that u is actually only implicitly dependent on x and t,
% through eta(x,t).

% As expected, the equation below produces a horizontal
% fluid velocity profile below the leading wave-crest
% that is quadratically dependent/proportional
% to z, and the horizontal fluid velocity always increases with z
% and reaches its maximum value for z = h0 + eta_max (at the top of
% the leading wave-crest).

for i = 1:11
    u_profile(i) = c0*((1/h0)*max(eta) - ...
                    ((1/(4*(h0^2)))*((max(eta))^2)) ...
        + (h0*(((eta((eta_max_x_index_storage(n+1))-1)) ...
            - 2*max(eta) ...
        + (eta((eta_max_x_index_storage(n+1))+1)))/(delta_x^2)))* ...
                ((1/3) - ((z(i))^2)/(2*(h0^2))));
end

%max(u_profile);

% Below: Storing the horizontal fluid speed evaluated
% at z = h0 + eta_max for each time step

u_eta_max_storage(n+1) = max(u_profile);

% Below: u_eta_max_storage_smoothed vector is updated in every
% time-step, and will eventually serve in the calculations as
% the values used for the horizontal fluid velocity u [m/s]
% at top of the leading wave

u_eta_max_storage_smoothed = smooth(u_eta_max_storage,smooth_span_u);

%u_profile;

% Below: Plotting the surface profile eta(x,n*delta_t).
% As we are typically advancing the solution forward in time by
% very small time steps, we won't see much difference in the profile
% from one time step to the next, so the logical if-check below makes
% sure there is a bigger time interval between each plot.
% Currently set to plot for every xx-th time step, by checking if the
% time-step index number n can be divided by xx, this can of course
% be modified as desired.

% Also plotting the horizontal fluid velocity profile beneath
% the leading wave-crest (the maximum eta-value) for each xx-th time
% step.

%if n > 175
if rem(n,8000) == 0
    figure
    plot(x,eta + h0,'-black');      % Plotting surface profile eta(x,t)
    hold on
    plot(x,ones(size(x)) * h0,'--black')  % Plotting water height h0
    hold off
```

```
        xlim([0 L])
        ylim([0 ((a0 + h0)+1)])
        xlabel('x [m]');
        ylabel(['Eta(x) [m] at time: ' num2str(t) ...
                    ' seconds, n =' num2str(n)]);
        hold off

        %z;
        %u_profile;
        %max(u_profile);

        figure
        plot(u_profile,z,'-black');  % Plotting the u(z) velocity profile
        hold on
        xlim([0 5])
        ylim([0 ((h0 + max(eta))+0.5)])
        xlabel(['Horizontal fluid velocity u(z) [m/s] beneath max. eta'...
                , 10,' at time: ' num2str(t) ' seconds, n =' num2str(n)]);
        ylabel('0 < z [m] < h0 + eta max');
        hold off

    end

end    % End of while loop and all calculations for current time step

% Below: We need to store the LAST calculated eta vector,
% that is eta(x, n_max*delta_t) to the eta_storage_matrix, as we don't
% go through the while-loop after eta(x, n_max*delta_t) has been
% calculated.

% We only need to update "internal" values, first and last elements are
% the boundary conditions a0 and 0 for all time

% The eta_storage matrix should now contain the approximative solution
% of eta(x,t) for all timesteps.

for i = 2:N+1
        eta_storage(i,(n+1)) = eta(i);
end

%eta_storage;

% Below: Wave-breaking criterion check for leading wave-crest:

for i = 2:n_max+1
    if u_eta_max_storage_smoothed(i) > U_storage_smoothed(i) ...
        && abs(U_storage_smoothed(n_max+1) - U_storage_smoothed(i)) < 0.2
        disp(['Wave breaking starting to occur at ' ...
              num2str((i-1)*delta_t) ' seconds, n = ' num2str(i-1) ...
              ', verify by visual control of plot'])
        break
    end
end

% Below: Adding final time as last element of t_overview vector
% Need to do it here as we are not going through the while-loop
% after t has been updated with the value for the final time

t_overview(length(t_overview)) = t;
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
eta;          % Final surface profile eta(x,t)

max(eta);     % The maximum value of eta at the final time

t_overview;   % Vector containing overview of all time values

eta_max_storage; % Vector containing overview of the maximum values of eta
                 % for all time steps

eta_max_x_index_storage; % Vector containing overview of the x-vector index
                         % for the maximum value of eta for each time step

eta_max_x_position_storage; % Vector containing overview of the x position
                            % for the maximum value of eta for each time
                            % step

U_storage;  % Vector containing calculated bore front/leading wave-crest
            % speed U [m/s] for all time steps (after first averaging)

U_storage_smoothed; % Vector containing calculated bore front/leading wave
                    % speed U [m/s] for all time steps (after first
                    % averaging AND smoothing)

u_eta_max_storage_smoothed;  % Vector containing the horisontal fluid
                             % velocity u(x,z,t) evaluated at
                             %  z = h0 + eta_max for each time step.

% Below: TEST vector only. Smoothing U-storage data to get a smooth graph
% for time series U(t). See Matlab documentation for explanation.
% NB! This vector should be equal to final U_storage_smoothed vector, do
% a difference check on the vector pair and verify that the resulting
% vector is zeros.

U_storage_smoothed_test = smooth(U_storage,smooth_span_U);

% Below: TEST vector only. Smoothing u-storage data to get a smooth graph
% for time series u(t). See Matlab documentation for explanation.
% NB! This vector should be equal to final u_storage_smoothed vector, do
% a difference check on the vector pair and verify that the resulting
% vector is zeros.

u_eta_max_storage_smoothed_test = smooth(u_eta_max_storage,smooth_span_u);

% Below: Plotting the averaged and smoothed leading wave phase speed [m/s],
% and plotting u(x,z = h0 + eta_max,t) [m/s], the horizontal fluid velocity
% evaluted at z = h0 + eta_max and x = x(eta_max) for each time step
% Also possibility for plot of "test"-smoothed leading wave speed U [m/s]

figure
plot(t_overview, U_storage_smoothed,'-black');
hold on
%plot(t_overview, U_storage_smoothed_test,'-red');
plot(t_overview, u_eta_max_storage_smoothed,'--black');
legend('phase speed U','horizontal fluid velocity u');
xlim([0 delta_t*n_max])
ylim([0 2])
xlabel('Time, t [s]');
```

```
ylabel('U and u (evaluated at top of leading wave-crest) [m/s]');
hold off

% Below: Plotting the maximum horizontal fluid velocity
% u(x,z = h0 + eta_max,t) [m/s] as a function of eta_max

figure
plot(eta_max_storage, u_eta_max_storage_smoothed,'-black');
hold on
%plot(t_overview, u_eta_max_storage,'--black');
xlim([0 max(eta_max_storage)])
ylim([0 5])
xlabel('Eta max [m]');
ylabel('u eta max [m/s]');
hold off

% Below: Plotting a time-series for the height eta_max of the leading wave
% Possibility also for plotting u(x,z = h0 + eta_max,t) [m/s],
% the horizontal fluid velocity evaluted at z = h0 + eta_max
% and x = x(eta_max) for each time step.

figure
plot(t_overview, eta_max_storage,'-black');
hold on
%plot(t_overview, u_eta_max_storage,'--black');
xlim([0 delta_t*n_max])
ylim([0 0.8])
xlabel('Time, t [s]');
ylabel(['Maximum eta [m]']);
hold off

% Below: Code for finding the approx. time where the leading wave-crest is
% first visible, by the criterion that there should be a local maximum
% (a bump) in the surface profile, with eta < a0, as this is by experience
% from simulations always the observed start of the development of the
% leading wave-crest.

for j = 2:n_max
    [pks, pks_index] = findpeaks(eta_storage(1:N+2,j));
    if pks(length(pks)) < a0 && pks(length(pks)) > a0/2
        disp(['Leading wave-crest first visible after ' ...
            num2str(t_overview(j)) ' seconds, n = ' num2str(j-1)])
        disp(['Leading wave-crest first visible height (eta) ' ...
            num2str(pks(length(pks)),15) ' meters'])
        break
    end
end

n      % Displays final time step index value

t      % Displays final value of the time

delta_x     % Displays spatial step size/spacing

delta_t     % Displays temporal step size

eta_max_x_position_storage(n+1) % Displays the x-position of the leading
                                % wave-crest/bore front at the final
                                % time-step
```

APPENDIX A – SOURCE CODE FOR BBM_SOLVER

```
max(eta)     % The maximum value of eta at the final time

%=========================================================================

toc  % Stops timer for program run
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% BBM error checker
% Program to solve the BBM equation (as given in "Linear and nonlinear
% waves" by G.B Whitham, p. 463) numerically with Heuns method for
% time-stepping, applied to the physical problem of a bore with height a0
% flowing over undisturbed water with height h0.
% Also checks the L2-error of the numerical scheme against an analytical
% solitary wave solution.

% Note: Simulations with different step-lengths should be run to the
% same final time when checking for the L2-error!

% Output:
% i)   Approximative solution of the surface profile eta(x,t) on an
%      (x,t)-grid (so dimensions are 1 spatial + 1 temporal) for all
%      time-steps.
% ii)  L2-error of the numerical scheme

% Results displayed in plots:
% i)   Surface-profile eta(x) for a user-controlled set of t-values
% ii)  Time series of eta_max, the maximum wave-height in the domain

%=========================================================================

% NB! Remember to clear the memory between each run of the program

% Use command "close all" to close all figures from a simulation

% The program contains a lot of "greened-out" code pieces used during the
% development of the program, they are intentionally left in the program.
% They can be useful in many ways, e.g. if you want the value of a variable
% printed to the screen at a certain time/point in the program

% Elaborate explanations of code pieces are intentionally left in the
% program, hopefully to easy the use of the program for any new users.

% Last review/update of code: 20.12.18

%=========================================================================
% MISCELLANOUS
%=========================================================================

clear  % Clearing workspace, freeing up system memory

tic    % Starts timer for program run

% Below: Set plot outline color to white, as it works better when copying
% plots into documents. Code from
% https://www.mathworks.com/help/matlab/creating_plots/
% default-property-values.html, checked 18.10.18, 11:32

set(groot,'defaultFigureColor','w')
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
%=========================================================================
% INPUT PARAMETERS
%=========================================================================

a0 = 0.5;     % Height of solitary wave (ABOVE undisturbed water depth) [m]
              % NOTE: The phase speed of the solitary wave is given
              % by c = a0/2 + 1 [m/s]

h0 = 1;       % Undisturbed water depth [m]

L = 150;      % Length of spatial domain (in x-direction) [m]
              % Recommended to set this to an even number

s_start_pos = 7.5; % Divisor controlling the starting position of the
                   % solitary wave in the spatial interval. s_start_pos = 2
                   % corresponds to solitary wave initial position in the
                   % middle of the spatial interval

n_max = 1000;    % The desired maximum number of steps to take forward in
                 % time (Preferably this number should be divisible by
                 % rem-control number for plots of surface profile)

kappa = 1;       % Controls steepness of the initial surface profile in the
                 % transition from a0 to 0.
                 % Higher value = steeper bore front slope, and vice versa
                 % Note: Changing this value will affect the result for the
                 % approximative solution of eta(x,t) somewhat, but will
                 % apparently not have an effect on the occurence of
                 % wave-breaking depending on alpha = a0/h0

%=========================================================================
% CONSTANTS GIVEN BY INPUT PARAMETERS
%=========================================================================

c = (a0/2) + 1;     % Phase speed of solitary wave [m/s]

g = 1;              % Gravitational acceleration [m/(s^2)]

c0 = sqrt(g*h0); % Phase speed for linear surface waves in the shallow
                 % water approximation [m/s].
                 % See Whitham p. 454/KCD eq. (8.49) p. 362

gamma = (1/6)*c0*(h0^2); % Constant given in eq. (13.94),
                         % Whitham p. 462 [m^3/s]

b = (3/2)*(c0/h0);       % Constant for simplification

d  = gamma/c0;           % Constant for simplification

%=========================================================================
% SPATIAL AND TEMPORAL STEP SIZES, X-VECTOR
%=========================================================================

% NB! As per 20.12.18: LARGEST KNOWN stable delta_x and delta_t:

% delta_x = = 0.04 delta_t = delta_x/4 = 0.01

% UPDATE here if larger stable step-sizes are found
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
%=================================================================

k = 11;                     % Parameter used for controlling step-sizes in
                            % order-verification runs for the numerical scheme

delta_x = L/(2^k);          % Alternate formula for spatial steplength, used
                            % for order-verification runs for the numerical
                            % scheme. Note that this method ENSURES that
                            % L is divisible by delta_x, as then we have
                            % L/delta_x = 2^k, which is always an integer

delta_t = delta_x;          % Controls the temporal steplength (timestep)

N = (L/delta_x) - 1;        % The number of "internal" spatial grid points.
                            % Controls the "inner" size of vectors and the
                            % size of matrices, which will be N and N x N,
                            % respectively. Follows logically, and also from
                            % delta_x = L/(2^k) set equal to delta_x = L/(1+N)
                            % Note that x_l and x_r come in addition,
                            % so the total number of grid points
                            % on the spatial domain is N + 2

x = zeros(N+2,1);

for j = 1:length(x) % N+2       % Updates vector x with the correct
    x(j) = (j-1)*delta_x;       % x-values for the grid points
end

x;

%=========================================================
% ETA-VECTOR
%=========================================================

eta = zeros(N+2,1);     % Creates a (N+2 x 1) vector sized to hold
                        % eta-values for all x-points

%=========================================================================
% ETA STAR-VECTOR
%=========================================================================

% Creates a N x 1 vector sized to hold eta_star-values for all
% "internal" x-points (excluding x_l and x_r).
% eta_star values are the Euler predictions (when at the current time
% t = n*delta_t) for the eta-values at time t = (n+1)*delta_t

eta_star = zeros(N+2,1);

eta_star(1) = 0;        % First element of eta_star vector should be
eta_star;               % the LHS boundary condition for eta

%=========================================================================
% ETA (N+1)-VECTOR
%=========================================================================

% Creates a N x 1 vector sized to hold eta_n_plus_1-values for all
% "internal" x-points (excluding x_l and x_r).
% eta_n_plus_1 values are the new solution points (when at the current
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% time t = n*delta_t) for the eta-values at time t = (n+1)*delta_t

eta_n_plus_1 = zeros(N+2,1);

eta_n_plus_1(1) = 0;      % First element of eta_n_plus_1 vector should
eta_n_plus_1;             % be the LHS boundary condition for eta

%=========================================================================
% ETA STORAGE-MATRIX
%=========================================================================

% Creates a matrix to save all eta-values for all time steps
% One column in eta_storage will contain all eta_values for a given
% time-step. The first column vector represents eta(x,0), the initial
% surface profile

eta_storage = zeros(N+2, n_max+1);

% Below: Taking LHS boundary conditions into eta_storage matrix for all
% time. Note that since we are now simulating a solitary wave, we want
% both the LHS and RHS boundary conditions to be 0.
% The RHS boundary condition is already 0 from the creation of the matrix.

for i = 1:n_max+1
    eta_storage(1,i) = 0;
    %eta_storage(N+2,i) = 0;
end

eta_storage;

%[m,n] = size(eta_storage)

%=========================================================================
% TRIDIAGONAL MATRIX T
%=========================================================================

% Constructing tridiagonal matrix T to be used in calculations of matrix A
% Using sparse matrix to save memory during simulations
% Code from
% www.mathworks.com/help/matlab/math/constructing-sparse-matrices.html
% found 02.10.18, 19:20

D = sparse(1:N,1:N,-2*ones(1,N),N,N);
E = sparse(2:N,1:N-1,ones(1,N-1),N,N);
S = E+D+E';

T = S;

%T = full(S);

%=========================================================================
% A-MATRIX
%=========================================================================

% Constructing matrix A to be used in the calculation of linear systems of
% equations with solutions respectively eta_star vector and "eta_(n+1)"
% vector
% Note: A should be a tridiagonal matrix, as it is a summation of a
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% diagonal matrix I and a tridiagonal matrix T. It should also be
% sparse, as both I and T are created as sparse.
% Note that using sparse matrices is crucial for simulation speed!
% For a certain reference problem used for testing, this step reduced
% simulation time usage by 97 %

% Below: Creating sparse identity matrix with dimensions (N x N)

I = sparse(1:N,1:N,1*ones(1,N),N,N);

% Below: Creating matrix A (N x N).

A = (1/delta_t)*I - (d/(((delta_x)^2)*delta_t))*T;

%=========================================================================
% B1-VECTOR
%=========================================================================

b1 = zeros(N,1);     % RHS vector (N x 1) in the system of equations
                     % A*eta_star_calc = b1, the system
                     % which is to be solved for eta_star_calc (Euler pred.)

%=========================================================================
% B2-VECTOR
%=========================================================================

b2 = zeros(N,1);     % RHS vector (N x 1) in the system of equations
                     % A*eta_n_plus_1_calc = b2, the system
                     % which is to be solved for eta_n_plus_1_calc

%=========================================================================
% TIME OVERVIEW-VECTOR
%=========================================================================

t_overview = zeros(n_max+1,1);      % Creating a vector containing all time
                                    % values for control purposes

%=========================================================================
% ETA MAX STORAGE-VECTOR
%=========================================================================

eta_max_storage = zeros(n_max+1,1); % Creating a vector for storing the
                                     % maximum value of eta for each
                                     % time step

%=========================================================================
% ETA MAX X-INDEX STORAGE-VECTOR
%=========================================================================

% Creating a vector for storing the x-index for the maximum value
% of eta for each time step

eta_max_x_index_storage = zeros(n_max+1,1);

%=========================================================================
% ETA MAX X-POSITION STORAGE-VECTOR
%=========================================================================
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% Creating a vector meant to store the x-position for the maximum value
% of eta for each time step

eta_max_x_position_storage = zeros(n_max+1,1);

%=========================================================================
% ERROR SUM STORAGE-VECTOR
%=========================================================================

% Creating storage vector to be used in calculations of the L2-error

error_sum_storage = zeros(n_max+1,1);

%=========================================================================
% ADVANCING SOLUTION VECTOR ETA FORWARD IN TIME
%=========================================================================

n = 0;      % Time step index initially at zero

t = 0;      % Time initially at zero

% Below: Updates vector eta to hold the initial condition/
% initial surface profile eta(x,0). Note that input parameter
% s_start_pos is used to control the initial position of the
% solitary wave in the spatial interval.

for j = 1:(N+2)     %
    eta(j) = a0*(sech((sqrt(3/2))*...
                (sqrt(a0/(a0+2)))*(x(j) - c*t - L/s_start_pos)))^2;
end

% Note that we are currently letting the initial condition function control
% the value of eta for the LHS and RHS grid points at t = 0, which could
% possibly have been "overridden"/set to a0 and 0, respectively. Code
% a few lines below to implement this, if desired.

eta;

% Below: Creating variables to be used for L2-error calculations.

error_sum = 0;

for j = 2:(N+1)
    error = (abs(eta(j) - a0*(sech((sqrt(3/2))...
            *(sqrt(a0/(a0+2)))*(x(j) - c*t - L/s_start_pos)))^2))^2;
    error_sum = error_sum + error;
end

error_sum;

% Below: Storing the error for n = 0, t = 0. This should be zero, check
% by running "error_sum_storage(1)" in command window.

error_sum_storage(n+1) = error_sum;

% Below: Storing the initial (at n=0 / t=0) maximum value of eta(x,t=0)
% as well as the index of the maximum eta value in the eta vector.
% As this index is the same for the max eta value in the eta vector and
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% the corresponding x-value in the x vector, we use this to find the
% x-position of the maximum eta value for all time steps.

[eta_max_storage(n+1), eta_max_x_index_storage(n+1)] = max(eta);

eta_max_x_position_storage(n+1) = x(eta_max_x_index_storage(n+1));

% Below: Plotting initial surface profile eta(x,0)

figure
plot(x,eta + h0,'-black');
hold on
plot(x,ones(size(x)) * h0,'--black');  % Plot undisturbed water height h0
xlabel('x [m]');
ylabel(['Eta(x) [m] at time: ' num2str(t)...
            ' seconds (initial surface profile)']);
xlim([0 L])
ylim([0 h0+a0+0.2])
hold off

% Below: Creating (N x 1) vector r_eta_n to be used in calculations

r_eta_n = zeros(N,1);

% Below: Creating (N x 1) vector r_eta_star to be used in calculations

r_eta_star = zeros(N,1);

% Below: While-loop advancing solution forward in time starts here!

while n < n_max     % Running calculations until maximum desired number
                    % of time steps is reached

    % Below: Adding current time for storage in time overview vector.
    % Note that as we start at n = 0, the first element in time overview
    % vector to be updated is element 1, where we store t = 0.

    t_overview(n+1) = t;

    % Below: Storing surface profiles eta(x,t) in the eta_storage matrix
    % for all time steps.
    % We only need to update "internal" values, first and last
    % elements are the BCs a0 and 0 for all time.
    % The first run of the while loop (for n = 0) will store the
    % initial surface profile eta(x,0) as the first column vector
    % in the eta_storage matrix.

    for i = 2:N+1
        eta_storage(i,(n+1)) = eta(i);
    end

    % Below: Calculating current r_eta_n vector, which is a function of
    % the current eta vector, as well as spatial steplength delta_x
    % and the constants b and c0

    for j = 2:N+1 % Starting at j=2 due to Matlab array indexing method
        r_eta_n(j-1) = -(b/2) * (((eta(j+1))^2 - ...
                        (eta(j-1))^2)/(2*delta_x))...
```

```
                    - c0*((eta(j+1) - eta(j-1))/(2*delta_x));
end

r_eta_n;

% Below: Creating (N x 1) vector eta_calc to use in linear system of
% equations, as the vector dimensions must be (N x 1) when solving
% the linear system

eta_calc = eta;

eta_calc(1) = [];
eta_calc(length(eta_calc)) = [];

%eta;
%eta_calc;

% Below: Creating (N x 1) vector eta_star_calc to use in linear system
% of equations, as the vector dimensions must be (N x 1) when solving
% the linear system.

eta_star_calc = zeros(N,1);

% Below: Calculating/updating b1 vector

b1 = r_eta_n + (1/delta_t)*eta_calc - (d/((delta_x^2)*delta_t)) ...
        * T * eta_calc;

%b1;

% Below: Solving linear system of equations for Euler prediction
% vector eta_star_calc

eta_star_calc = A\b1;

% Below: Adding current "internal" eta_star values from vector
% eta_star_calc to eta_star vector
% First and last elements of eta_star, corresponding to indices i=1
% and i=N+2 respectively, remains the boundary conditions: a0 and 0

for i = 2:N+1
    eta_star(i) = eta_star_calc(i-1);
end

%eta_star_calc;
%length(eta_star_calc);
%eta;
%eta_star;
%length(eta_star);

% Below: Calculating current r_eta_star vector, which is a function of
% the current eta_star vector, as well as spatial steplength delta_x
% and the constants b and c0

for j = 2:N+1 % Starting at j=2 due to Matlab array indexing method
    r_eta_star(j-1) = -(b/2) * (((eta_star(j+1))^2 - ...
                        (eta_star(j-1))^2)/(2*delta_x))...
                    - c0*((eta_star(j+1) - eta_star(j-1))/(2*delta_x));
```

```
end

% Below: Calculating/updating b2 vector

b2 = (1/2)*(r_eta_n + r_eta_star) + (1/delta_t)*eta_calc - ...
        (d/((delta_x^2)*delta_t)) * T * eta_calc;

%b2;

% Below: Creating (N x 1) vector eta_n_plus_1_calc to use in linear
% system of equations, as the vector dimensions must be (N x 1) when
% solving the linear system.

eta_n_plus_1_calc = zeros(N,1);

% Below: Solving linear system of equations for eta_n_plus_1_calc
% vector, the solution for the surface profile one step forward
% in time

eta_n_plus_1_calc = A\b2;

% Below: Adding current "internal" eta_n_plus_1 values from vector
% eta_n_plus_1_calc to eta_n_plus_1 vector
% First and last elements of eta_n_plus_1, corresponding to indices i=1
% and i=N+2 respectively, remains the boundary conditions: a0 and 0

for i = 2:N+1
    eta_n_plus_1(i) = eta_n_plus_1_calc(i-1);
end

%eta;           % Approx. solution for eta(x,t) at current time t
%eta_n_plus_1;  % Approx. sol. for eta(x,t+delta_t) at time t + delta_t

n = n + 1;      % Updating time step number for next run of while-loop
t = n*delta_t;  % Updating the time for the next run of the while-loop

eta = eta_n_plus_1; % Updating eta vector to be ready to repeat
                    % while-loop for next time step

error_sum = 0;  % Resetting error_sum before calculation

for j = 2:(N+1)
    error = (abs(eta(j) - a0*(sech((sqrt(3/2))...
                *(sqrt(a0/(a0+2)))*(x(j) - c*t - L/s_start_pos)))^2))^2;
    error_sum = error_sum + error;
end

% Below: Storing the calculated error_sum for the current time step.

error_sum_storage(n+1) = error_sum;

% Below: Storing the current maximum value of eta(x,t)
% as well as the index of the maximum eta value in the eta vector.
% As this index is the same for the max eta value in the eta vector and
% the corresponding x-value in the x vector, we use this to find the
% x-position of the maximum eta value for all time steps.

[eta_max_storage(n+1), eta_max_x_index_storage(n+1)] = max(eta);
```

```
% Below: The max(vector) function in matlab will pick the first
% (leftmost) element in the vector, if there are more than one maximum
% value in the vector! In the start of the simulation, this can
% typically result in the max(eta)-code part picking the very first
% element in eta-vector, with index 1, as the maximum. This is because
% the initial surface profile has a constant value a0 as the
% maximum eta-value at its left part, so there are many max values.
% This will result in an error message further down in the code, as the
% code for calculating the horizontal velocity profile below the
% maximum eta-value employs finite differences for the second order
% derivative, and will then try to pick element 0 from the eta-vector,
% which will not work since matlab starts indexing at element 1.
% Note that this little code piece is only used if the maximum eta
% value actually has index value 1 in the eta-vector, so it will
% have no effect on the when we are actually interested in the max
% eta-value, that is after many time steps, when a leading wave-crest
% has formed.

%eta_max_x_index_storage(n+1);

if eta_max_x_index_storage(n+1) == 1
    eta_max_x_index_storage(n+1) = eta_max_x_index_storage(n+1) + 1;
end

%eta_max_x_index_storage(n+1);

eta_max_x_position_storage(n+1) = x(eta_max_x_index_storage(n+1));

%eta_max_storage(n+1);
%max(eta);

% Below: Plotting the surface profile eta(x,n*delta_t).
% As we are typically advancing the solution forward in time by
% very small time steps, we won't see much difference in the profile
% from one time step to the next, so the logical if-check below makes
% sure there is a bigger time interval between each plot.
% Currently set to plot for every xx-th time step, by checking if the
% time-step index number n can be divided by xx, this can of course
% be modified as desired.

% Also plotting the horizontal fluid velocity profile beneath
% the leading wave-crest (the maximum eta-value) for each xx-th time
% step.

%if n > 175
if rem(n,100) == 0
    %hold on
    figure
    plot(x,eta + h0,'-black');    % Plotting surface profile eta(x,t)
    hold on
    plot(x,ones(size(x)) * h0,'--black')  % Plotting water height h0
    hold off
    xlim([0 L])
    ylim([0 h0+a0+0.2])
    xlabel('x [m]');
    ylabel(['Eta(x) [m] at time: ' num2str(t) ...
                ' seconds, n =' num2str(n)]);
    hold off
```

```
    end

end    % End of while loop and all calculations for current time step

figure
plot(x,eta + h0,'-black');    % Plotting surface profile eta(x,t)
hold on
plot(x,ones(size(x)) * h0,'--black')  % Plotting water height h0
hold off
xlim([0 L])
ylim([0 h0+a0+0.2])
xlabel('x [m]');
ylabel(['Eta(x) [m] at time: ' num2str(t) ' seconds, n =' num2str(n)]);
hold off

% Below: We need to store the LAST calculated eta vector,
% that is eta(x, n_max*delta_t) to the eta_storage_matrix, as we don't
% go through the while-loop after eta(x, n_max*delta_t) has been
% calculated.

% We only need to update "internal" values, first and last elements are
% the boundary conditions a0 and 0 for all time

% The eta_storage matrix should now contain the approximative solution
% of eta(x,t) for all timesteps.

for i = 2:N+1
        eta_storage(i,(n+1)) = eta(i);
end

%eta_storage;

% Below: Adding final time as last element of t_overview vector
% Need to do it here as we are not going through the while-loop
% after t has been updated with the value for the final time

t_overview(length(t_overview)) = t;

eta;          % Final surface profile eta(x,t)

max(eta);     % The maximum value of eta at the final time

t_overview;   % Vector containing overview of all time values

eta_max_storage; % Vector containing overview of the maximum values of eta
                 % for all time steps

eta_max_x_index_storage; % Vector containing overview of the x-vector index
                         % for the maximum value of eta for each time step

eta_max_x_position_storage; % Vector containing overview of the x position
                            % for the maximum value of eta for each time
                            % step

% Below: Calculating the L2-error:

L2_error = ((1/(N+1))*(max(error_sum_storage)))^(1/2); % N+1 subintervals!
```

APPENDIX B – SOURCE CODE FOR BBM _ERROR_CHECKER

```
% Below: Plotting a time-series for the height eta_max of the leading
% wave-crest. This should logically be a horizontal line when we
% are simulating a solitary wave

plot(t_overview, eta_max_storage,'-black');
hold on
xlim([0 delta_t*n_max])
ylim([0 2])
xlabel('Time, t [s]');
ylabel(['Maximum eta [m] as a function of time]']);
hold off

n      % Displays final time step index value

t      % Displays final value of the time

delta_x     % Displays spatial step size/spacing

delta_t     % Displays temporal step size

eta_max_x_position_storage(n+1) % Displays the x-position of the leading
                                % wave-crest/bore front at the final
                                % time-step

max(eta)     % The maximum value of eta at the final time

L2_error     % The L2-error

%=========================================================================

toc  % Stops timer for program run
```

Bibliography

[1] M. Bjørkavåg and H. Kalisch, *Wave breaking in Boussinesq models for undular bores,* Physics Letters A 375, p. 1570-1578 (2011)

[2] H. Favre, *Ondes de Translation*, Dunod, Paris (1935)

[3] M. K. Brun, *Wave breaking in long wave models and undular bores,* Master thesis in applied and computational mathematics, University of Bergen, Bergen (2015)

[4] G. B. Whitham, *Linear and nonlinear waves,* Wiley-Interscience [John Wiley & Sons], New York (1974).

[5] S. H. Strogatz, *Nonlinear dynamics and chaos (second edition),* Westview Press, Boulder, CO (2015)

[6] P. K. Kundu, I. M. Cohen, D. R. Dowling and G. Tryggvason, *Fluid Mechanics (sixth edition)*, Academic Press, London (2016)

[7] L. C. Evans, *Partial Differential Equations (second edition),* American Mathematical Society, Providence, Rhode Island (2015)

[8] H. Kalisch, *Solitary Waves of Depression,* Journal of Computational Analysis and Applications, vol. 8, no. 1, p. 5-24 (2006)

[9] T. Sauer, *Numerical Analysis (second edition),* Pearson Education Limited, Essex (2014)

[10] A. Quarteroni, R. Sacco and F. Saleri, *Numerical Mathematics (second edition),* Springer, Berlin (2007).

[11] M. K. Brun and H. Kalisch, *Convective wave breaking in the KdV equation,* Analysis and Mathematical Physics, vol. 8, issue 1, p. 57-75 (2018)

[12] A. Senthilkumar and H. Kalisch, *Wave breaking in the KdV equation on a flow with constant vorticity,* European Journal of Mechanics – B/Fluids, vol. 73, p. 48-54 (2019)

[13] A. Ali and H. Kalisch, *Mechanical Balance Laws for Boussinesq Models of Surface Water Waves,* Journal of Nonlinear Science, vol. 22, issue 3, p. 371-398 (2012)