User manual for the stochastic part of the PFL wave model

Mathias Klahn[1], Per A. Madsen[1] and David R. Fuhrman[1]

[1]*Department of Mechanical Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark*

Contents

1 Introduction

This document describes the structure of the stochastic part of the PFL wave model. This part can be used to simulate the time evolution of multi-directional irregular wave fields described by JONSWAP spectra in water of both finite and infinite water depth.

In the numerical simulations, a cartesian coordinate system is adopted with the horizontal (x, y)-plane located at the still water level and the z-axis pointing vertically upwards. The wave fields are discretized on a horizontal domain of size $L_x \times L_y$, and the water depth h can take any finite value. The rapid solution of the Laplace problem inherent to the potential flow formalism is made possible by using, among other techniques, an artificial boundary condition, whose vertical position is $-b$, where b is some positive length (typically satisfying $b/h \ll 1$). For a full description and validation of the method, the reader is referred to the papers by Klahn et al. [4], [5], [6]. The remainder of this document simply serves the purpose of outlining the structure of the code and, perhaps most importantly, how it can be used.

2 Structure of the stochastic part

The overall structure of the stochastic part of the code is illustrated in figure 1. The code consists of a directory containing the source code and a simulation directory in which the simulations of irregular wave fields can be carried out. one for each type of deterministic wave simulation. The contents of the simulation directory are illustrated in figure 2, from which it can be seen that the simulation directory contains a simulation program (`simulateIW.m`), five programs for analyzing the results of the simulation (`surfSkewKur.m`, `surfElevPDF.m`, `surfVelDist.m`, `velStdProf.m` and `accStdProf.m`) and a directory for output data (`OutputData`). The simulation program initializes an irregular wave field, integrates it in time and writes the results of the computation to file in the folder `OutputData`. The analysis programs can then be used to read and analyze these files. In the following, we elaborate on the structure of the simulation and analysis programs, and we describe the most important variables used in the programs.

2.1 Structure of the simulation program

The simulation program for irregular waves is divided into six sections. These are:

1. Addition of the path of the source directory to the search path.
2. Definition of non-dimensional physical and computational parameters.

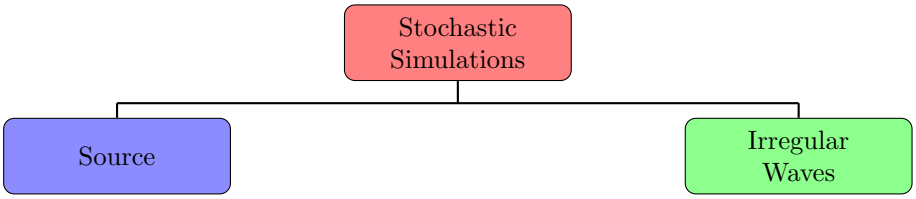

Figure 1: The overall structure of the code for the stochastic simulations. The blue source directory only contains subroutines, while the green simulation directory contains files for the simulation and analysis of irregular wave fields.

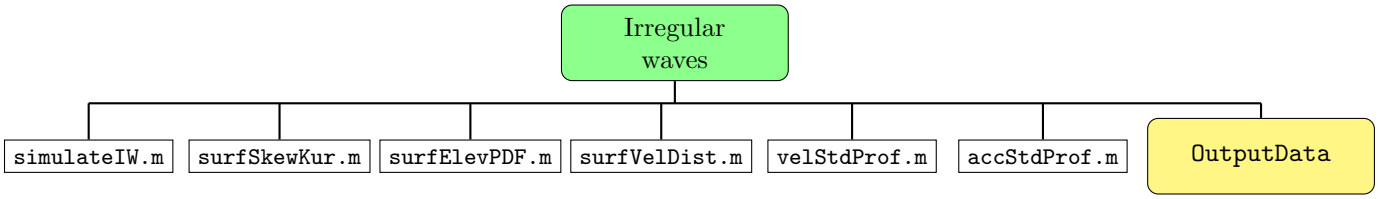

Figure 2: The structure of the simulation directory for the irregular waves. The white boxes represent Matlab files for the simulation and subsequent analysis of the relevant wave field. Output files are send to the yellow directory `OutputData`.

3. Definition of fundamental dimensional physical scales (in SI units) and calculation of all other relevant dimensional quantities based on the non-dimensional quantities defined earlier.
4. Initialization of the surface elevation and the surface potential.
5. Calculation of time independent vectors and matrices which, among other things, are used for differentiation and preconditioning during the time integration.
6. Time integration of the wave field over the desired time interval and export of the results to the directory `OutputData`.

The user of the simulation programs only needs to consider sections 1 and 2 and, if a different output is desired, also section 6.

2.2 Structure of the analysis programs

The analysis programs are also divided into sections. The number of sections varies with the type of analysis. Nevertheless, the first two sections of the analysis programs for irregular wave fields are conceptually the same. These two sections are:

1. Addition of the path of the source directory to the search path.
2. Definition of non-dimensional physical and computational parameters.

The definition of the non-dimensional variables is needed, as the files in the output directory are named after these to distinguish different cases. The sections 3, 4,... perform the data analysis.

2.3 Important variables in the programs

Knowing the most important variables of the programs is always of great value to the user. Moreover, the current simulation and analysis programs will most certainly not do exactly what the user wants them to do, and for that reason it may also be important to know how to manipulate the variables. In this section we therefore very briefly describe the key variables in the programs and how these can be modified.

The horizontal dimensions of the computational domain is always discretized using `2*Nx` equidistant grid points in the x-direction and `2*Ny` equidistant grid points in the y-direction. The surface elevation and surface potential are represented by the column vectors `eta` and `Phis`, respectively, which both have length `2*Nx*2*Ny`. In the simulation programs, these quantities are combined into the single variable `y = [eta; Phis]`, and the values of `eta` and `Phis` at the `(nx,ny)`th grid point can thus be extracted using the lines of code

```
>> etanxny = y(nx+2*Nx*(ny-1))
>> Phisnxny = y(nx+2*Nx*(ny+2*Ny-1))
```

where 1 ≤ nx ≤ 2*Nx and 1 ≤ ny ≤ 2*Ny.

The vertical dimension of the computational domain (which is only relevant when solving the Laplace equation or otherwise dealing with quantities in the bulk of the water) is always discretized using Ns+1 grid points which are denoted s1, s2, ..., sNs+1. These grid points are the quadrature points of a Legendre-Gauss-Lobatto quadrature of order Ns, and are ordered such that the first grid point in the vertical is s1 = -b (i.e. the location of the artificial boundary condition) and such that for the horizontal grid point with index (nx,ny) the last grid point in the vertical is sNs+1 = etanxny. The solution of the Laplace equation is denoted by F and is calculated in the source program compws.m. To extract its value at the (nx,ny,ns)th grid point one needs the line of code

```
>> Fnxnyns = F(nx+2*Nx(ny-1)+2*Nx*2*Ny*(ns-1))
```

where 1 ≤ nx ≤ 2*Nx, 1 ≤ ny ≤ 2*Ny and 1 ≤ ns ≤ 2*Ns.

To differentiate the two-dimensional objects eta and Phis and the three-dimensional object F with respect to the spatial coordinates, one can use the vectors and matrices Dx1, Dx2, Dy1, Dy2, Ds1 and Ds2. These are computed in section 5 of the simulation programs (section 4 for the focusing event). To calculate the values of the first x- and y-derivatives of eta on the horizontal grid, the lines of code

```
>> etax = reshape(ifft(Dx1.*fft(reshape(y(1:2*Nx*2*Ny),[2*Nx,2*Ny]))),[2*Nx*2*Ny,1])
>> etay = reshape(ifft(fft(reshape(y(1:2*Nx*2*Ny),[2*Nx,2*Ny]),2*Ny,2).*Dy1,2*Ny,2),[2*Nx*2*Ny,1])
```

should be executed. Note that the multiplication is done pointwise with the ".*"-operator. To calculate the second x- and y-derivatives of eta, Dx1 and Dy1 are simply replaced by Dx2 and Dy2. Moreover, to calculate the x- and y-derivatives of Phis, the vector y(1:2*Nx*2*Ny) should be replaced by the vector y(2*Nx*2*Ny+1:2*2*Nx*2*Ny). The velocities in the bulk of the water are obtained by differentiating the potential. This calculation is, however, a little tricky since the program works with the potential in the transformed coordinates, i.e. F, which is defined as $F(x, y, s(x, y, z)) = \Phi(x, y, z)$, where $\Phi(x, y, z)$ is the velocity potential in Cartesian coordinates (see eq. 13 of Klahn et al. for the definition of s). To calculate the components of the fluid velocity one first executes the lines of code

```
>> Fx  = reshape(ifft(Dx1.*fft(reshape(F,[2*Nx,2*Ny*(Ns+1)]))),[2*Nx*2*Ny*(Ns+1),1])
>> Fy  = reshape(ifft(fft(reshape(F,[2*Nx,2*Ny,Ns+1]),2*Ny,2).*Dy1, 2*Ny, 2),[2*Nx*2*Ny*(Ns+1),1])
>> Fs  = reshape(reshape(F,[2*Nx*2*Ny,Ns+1])*Ds1,[2*Nx*2*Ny*(Ns+1),1])
```

after which the fluid velocity in the x-direction is obtained using the line of code

```
>> u  = reshape(Fx-reshape(((etax./beta).*reshape(Fs,[2*Nx*2*Ny,Ns+1])).*s1Vec,
                [2*Nx*2*Ny*(Ns+1),1]), [2*Nx,2*Ny,Ns+1]),
```

the fluid velocity in the y-direction is obtained using the line of code

```
>> v  = reshape(Fy-reshape(((etay./beta).*reshape(Fs,[2*Nx*2*Ny,Ns+1])).*s1Vec,
                [2*Nx*2*Ny*(Ns+1),1]), [2*Nx, 2*Ny,Ns+1]),
```

and, finally, the fluid velocity in the z-direction is obtained using the line of code

```
>> w  = reshape(((2./beta).*reshape(Fs,[2*Nx*2*Ny,Ns+1])), [2*Nx,2*Ny,Ns+1]).
```

In these lines of code beta = b+eta and s1Vec = sGrid+1. It should be noted that u, v and w only contain the values of the fluid velocity at the grid points within the computational domain whose extent in the vertical dimension is limited to $-b < z < \eta$. For an example of how to calculate the fluid velocities for depths greater than b, the user of the program is referred to the subroutine compVelProfile.m in the source directory.

3 How to execute the simulations of irregular wave fields

As described in the previous section, the execution of the simulation and analysis programs requires that the user modifies these slightly. The path of the source code should be added to the search path in the programs' section 1, and the physical and computational parameters should be declared in the programs' section 2. In the following, we explain the meaning of the (non)-dimensional parameters that the user must specify in order to execute the simulation of irregular wave fields. We also illustrate the output of the analysis programs.

3.1 Irregular wave fields

Steady nonlinear waves can be simulated by executing the program `simulateIW.m`. For these simulations the artificial boundary condition is chosen to be located at $z = -b = -b_{\text{coef}} Hs$, where c_{coef} is a coefficient decided by the user and H_s is the significant wave height (i.e. four times the standard deviation of the initial surface elevation). To execute the program, the user must specify the following non-dimensional parameters in the program's section 2:

- `kph`: The non-dimensional water depth $k_p h$, where k_p is the peak wave number and h is the water depth. Note that infinite water depth is obtained by setting `kph = inf`.
- `steep`: The steepness of the wave field, ε, as defined above.
- `ND`: The parameter of the directional distribution. See e.g. eq. (2.4) of Klahn et al. [6].
- `gamma`: The peak enhancement factor of the frequency spectrum. See e.g. eq. (2.2) of Klahn et al. [6].
- `lx` and `ly`: The dimensions of the horizontal domain in units of peak wavelengths.
- `Nx`, `Ny` and `Ns`: The spatial resolution of the wave field. `2*Nx` grid points are used in the x-direction, `2*Ny` grid points are used in the y-direction, and `Ns+1` grid points are used in the vertical dimension. Note that the value of `Ns` is relevant only for the solution of the Laplace problem,
- `NPeriod`: The wave field is integrated in time over the time interval `[0, NPeriod*Tp]` where `Tp` is the peak period.
- `NStep`: The number of time steps every period.
- `tAdjust` and `nAdjust`: The parameters equal the parameters T_a/T_p and n, respectively, where T_a and n are defined in eq. (6) in the paper by Dommermuth [3] and T_p is the peak period.
- `bCoef`: The artificial boundary condition is located at `bCoef*Hs`, where `Hs` is the significant wave height.
- `epsGMRES`: The relative tolerance to which the set of linear equations corresponding to the discretized Laplace problem is solved by GMRES. Note that setting `epsGMRES = 1E-14` corresponds to solving the system of equations to double precision.

During the time integration, the time in units of peak periods is written to the terminal and results of the simulations are written to file in the folder `OutputData`. At the time `t = n*Tp`, where `n` is a non-negative integer, the program writes the vector `y` and the first four statistical moments of the surface elevation at the times `(n-1+1/NStep)*Tp`, `(n-1+2/NStep)*Tp`, ..., `(n-1+(NStep-1)/NStep)*Tp`, and `n*Tp` to file in `OutputData`.

To analyze the time evolution of the skewness and kurtosis of the surface elevation (these quantities are routinely used as a measure of the nonlinearity of the wave field), the user should execute the program `surfSkewKur.m`. An example of the output of the program is shown in figure 3, which has been produced with the parameters `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `lx = 50`, `ly = (1+ND)^(1/2)*lx`, `Nx = 128`, `Ny = 128`, `Ns = 5`, `NPeriod = 5`, `NStep = 50`, `tAdjust = 10`, `nAdjust = 4`, `bCoef = 1.5` and `epsGMRES = 1E-9`.

To compute the probability density function of the surface elevation at a specific time of the simulation, the user should execute the program `surfElevPDF.m`. This program compares the result of the simulation with the result of first-order theory. An example of the output of the program is shown in figure 4, which has been produced with the parameters `time = 0` (the time of comparison), `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`.

To compute the probability density functions of the components of the fluid velocity at the surface at a specific time of the simulation, the user should execute the program `surfVelPDF.m`. The program outputs the PDF of the velocities in all three Cartesian directions, and compares their PDF to the result of first-order theory. An example of the output of the program is shown in figure 5, which has been produced with the parameters `time = 0` (the time of comparison), `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The velocity scale u_0 in the figure is defined as $u_0 = \varepsilon(g/kp)^{1/2}$.

To compute the vertical profile of the standard deviation of each of the fluid velocity components at a specific time of the simulation, the user should execute the program `velStdProf.m`. The program compares the standard deviation profiles to the result of first-order theory. An example of the output of the program is shown in figure 6, which has been produced with the parameters `time = 0` (the time of comparison), `kph = 2`, `steep = 1E-6`, `ND = 2`,

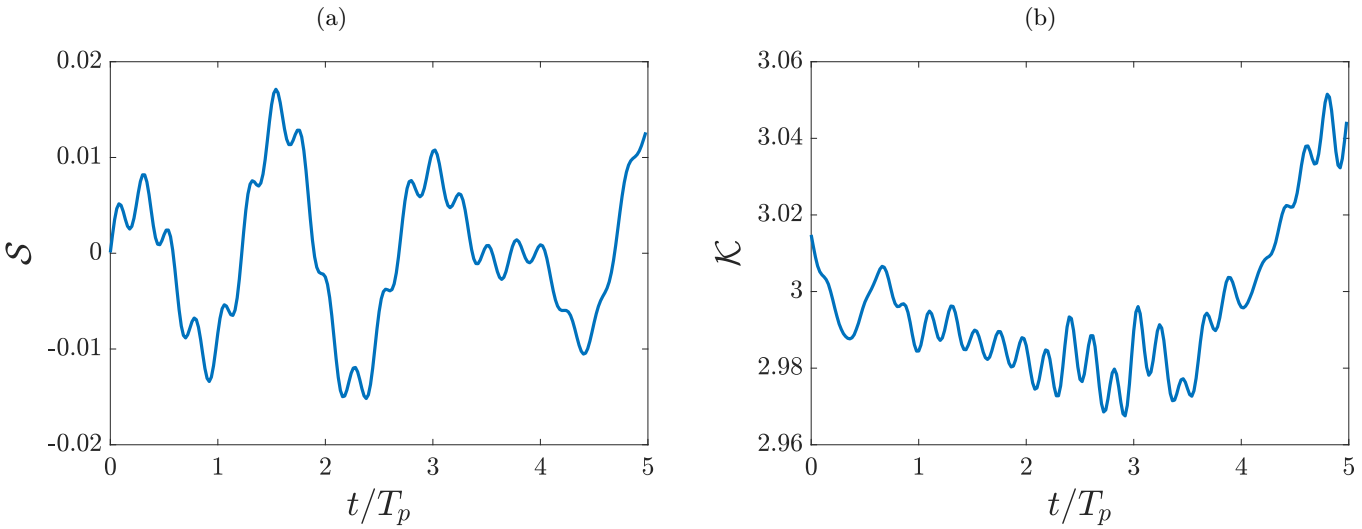

Figure 3: The output of `surfSkewKur.m` when used with the parameters `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `lx = 50`, `ly = (1+ND)^(1/2)*lx`, `Nx = 128`, `Ny = 128`, `Ns = 5`, `NPeriod = 5`, `NStep = 50`, `tAdjust = 10`, `nAdjust = 4`, `bCoef = 1.5` and `epsGMRES = 1E-9`. (a) The skewness of the surface elevation as a function of time. (b) The kurtosis of the surface elevation as a function of time.

`gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The velocity scale u_0 in the figure is defined as $u_0 = \varepsilon(g/kp)^{1/2}$.

To compute the vertical profile of the standard deviation of each of the fluid acceleration components at a specific time of the simulation, the user should execute the program `accStdProf.m`. The program compares the standard deviation profiles to the result of first-order theory. An example of the output of the program is shown in figure 7, which has been produced with the parameters `time = 0` (the time of comparison), `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The velocity scale u_0 in the figure is defined as $u_0 = \varepsilon(g/kp)^{1/2}$.

Acknowledgement

The authors gratefully acknowledge the funding received from Centre for Oil and Gas - DTU/Danish Hydrocarbon Research and Technology Centre (DHRTC). Project name: DEWIOS. Project ID: FL_110.

References

[1] CLAMOND, D. & DUTYKH, D. 2017 SSGW.m Matlab file exchange. URL `https://www.mathworks.com/matlabcentral/fileexchange/61499-surface-gravity-waves`

[2] CLAMOND, D. & DUTYKH, D. 2018 Accurate fast computation of steady two-dimensional surface gravity waves in arbitrary depth. *J. Fluid Mech.* **844**, 491-518.

[3] DOMMERMUTH, D. 2000 The initialization of nonlinear waves using an adjustment scheme. *Wave Motion* **32** (4), 307-317.

[4] KLAHN, M., MADSEN, P.A. & FUHRMAN, D. R. 2021 Simulation of three-dimensional nonlinear water waves using a pseudospectral volumetric method with an artificial boundary condition. *Int. J. Numer. Meth. Fluids* (In production).

[5] KLAHN, M., MADSEN, P.A. & FUHRMAN, D. R. 2021 On the statistical properties of surface elevation, velocities and accelerations in multi-directional irregular water waves. *J. Fluid Mech.* **910**, A23.

[6] KLAHN, M., MADSEN, P.A. & FUHRMAN, D. R. 2021 On the statistical properties of inertia and drag forces in nonlinear multi-directional irregular water waves. Accepted for publication in *J. Fluid. Mech.*

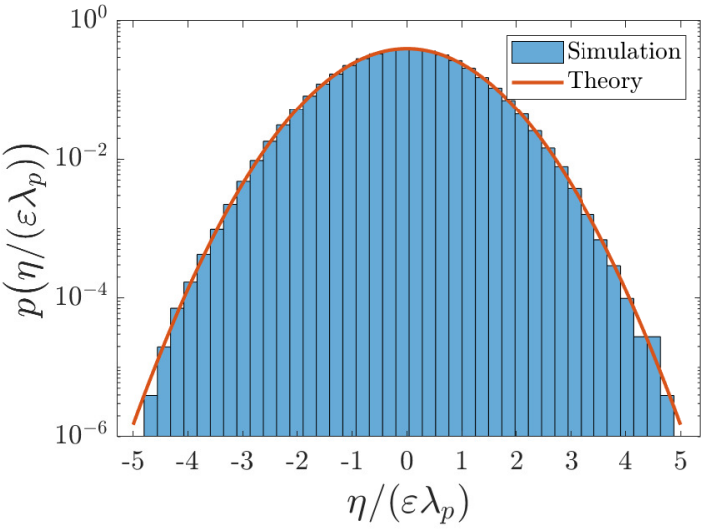

Figure 4: The output of `surfElevPDF.m` when used with the parameters `time = 0`, `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The figure shows a comparison between the probability density function of the surface elevation and the theoretical expectation based on first-order theory.

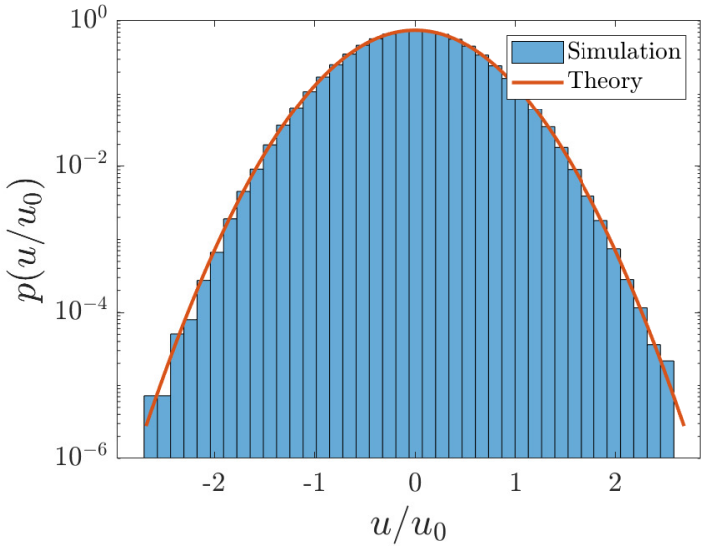

Figure 5: The output of `surfVelPDF.m` when used with the parameters `time = 0`, `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The figure shows a comparison between the probability density function of the fluid velocity in the x-direction at the surface and the theoretical expectation based on first-order theory. The program also outputs the results for the y- and z-directions.

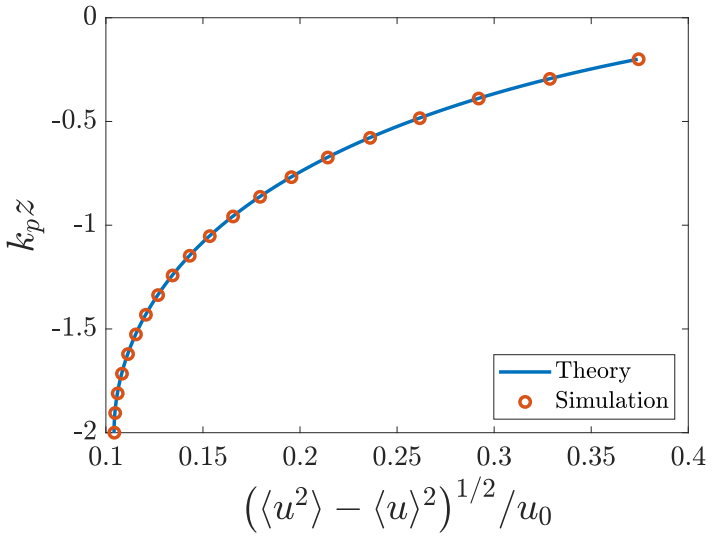

Figure 6: One of the outputs of `velStdProf.m` when used with the parameters `time = 0`, `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The figure shows a comparison between the standard deviation of the fluid velocity in the x-direction as a function of depth and the theoretical expectation based on first-order theory. The program also outputs the results for the y- and z-directions.

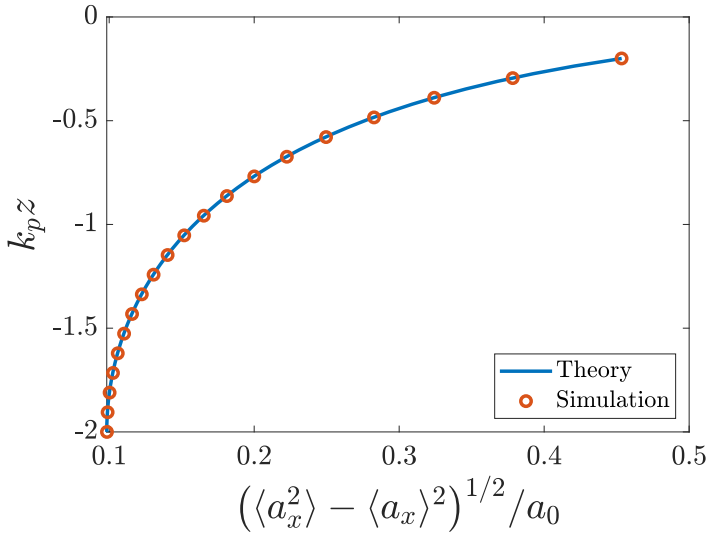

Figure 7: One of the outputs of `velStdProf.m` when used with the parameters `time = 0`, `kph = 2`, `steep = 1E-6`, `ND = 2`, `gamma = 3.3`, `A = 0.078`, `x0 = 20`, `y0 = Ly/2`, `b = 0.12`, `t0 = -20`, `tf = 10`, `deltat = 0.01`, `Nx = 128`, `Ny = 128`, `Ns = 10`, `dampstrat = 2`, `dampCoef = 0.7` and `epsGMRES = 1E-9`. The figure shows a comparison between the standard deviation of the fluid acceleration in the x-direction as a function of depth and the theoretical expectation based on first-order theory. The program also outputs the results for the y- and z-directions.