Solving a 2D Wave Equation using Finite Difference Methods

Steinnes L.

Institute of Physics, University of Oslo

lasse.steinnes@fys.uio.no

October 4, 2020

Abstract

A numerical scheme using the finite difference method is here applied on a general 2D wave equation. It is correctly verified for a constant solution, and reproduce the theoretical convergence rate $r = 2$ for a 2D standing wave with no damping. It fails to reproduce the convergence rate for a 2D-wave with damping and variable wave velocity, due to a bug in the implementation. Waves over three cases of subsea hills shows how numerical artefacts can accumulate over time, depending on how the wave is initialized and if the wave meet steep or sharp edge surfaces.

I Introduction

Solving partial differential equations by hand require use of sophisticated mathematical tools known from Calculus. However, with the rise of electronic computers around and after WWII, other numerical equations possibilities and approximations were made possible [1]. Today, solving a large number of equations numerically has become a vital tool for scientist and high-tech industry alike. With increasing CPU-capacity, computers can handle larger and larger mesh sizes, further propelling the versatility and promise of numerical solutions to large systems and challenging mathematical problems [2].

Amongst the methods to solve partial differential equations (PDEs) are the finite difference methods [3]. They discretize the differential equations by approximating the derivative. Thus, a numerical scheme can be implemented, and a solution can be obtained for a given dimensionality.

The aim of this paper is using a central finite difference scheme abd to solve a wide scope of 2D wave equations. It is laid out by first describing a general 2D-wave equation, and derive a discrete solution to this equation. Thereafter, special cases of the 2D-wave equation is presented. The first of these cases are used for convergence rate tests. As a last case, a description of numerical stability by exploring ocean waves with the same initial condition on different seabeds is given. The results are presented with figures and benchmarks, followed by a discussion and critical evaluation. The paper is ended with some brief concluding remarks.

All relevant programmes to solving the PDE's described are available at `https://github.com/lasse-steinnes/IN5270/tree/master/wave_project`.

II Theory and Methods

The theory section is mainly based upon Xing Cai's lectures in the course Numerical Methods for Partial Differential Equation (IN5270) at UiO [4] and the book Finite Difference Computing with PDEs [3].

II.I A general 2D-wave equation

A general planar wave can be described by the PDE

$$\frac{\partial^2 u}{\partial t^2} + b\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(g(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(g(x,y)\frac{\partial u}{\partial y}\right) + f(x,y,t). \tag{1}$$

Here, $u(x,y,t)$ is the solution to the wave equation (eq. 1) with source term $f(x,y,t)$, space dependent velocity $g(x,y)$ and a damping coefficient b. Since this presents itself as a second order PDE, intial conditions (ICs) and boundary conditions (BCs) are needed to solve it, both analytically and numerically.

Here, a Neumann boundary condition is considered

$$\frac{\partial u}{\partial n} = 0. \tag{2}$$

The initial conditions are

$$u(x,y,0) = I(x,y) \quad \text{and} \quad u_t = V(x,y), \tag{3}$$

where u_t is the partial differential of u wrt. time. The rectangular domain $\Omega = [0,L_x] \times [0,L_y]$ is considered.

II.II Derivation of the Numerical Scheme

To described a general 2D-wave equation (eq. 1), one can use a central difference approximation to the derivatives.

This entails using the approximations

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j}^{n+1} - u_{i,j}^{n-1}}{2\Delta t} = [D_{2t}u]_{i,j}^{n}\,, \tag{4}$$

$$\frac{\partial^2 u}{\partial t^2} \approx \frac{u_{i,j}^{n+1} - 2u_{i,j}^{n} + u_{i,j}^{n-1}}{\Delta t^2} = [D_t D_t u]_{i,j}^{n}\,, \tag{5}$$

and

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} = [D_x D_x u]_{i,j}^{n}\,. \tag{6}$$

The same approximation is made for $\partial^2 u/\partial y^2$ as for the x-variable in eq. 6. Thus the exact solution $u(x,y,t)$ is fulfilled approximately at internal mesh points i,j in space and n points in time, i.e. at x_i, y_i and t_n. Let the internal points be described by $i = 0,1,...,N_x$, $j = 0,1,...,N_y$ and $n = 0,1,...,N_t$. This is the method of finite differences.

Now, an approximation to the composite derivative in the 2D-wave equation (1) must be made. It is possible to use the chain rule, however, this is not common practice in numerical methods on PDEs, because it the resulting equation does not properly describe a net flux. A way to work around this issue, is approximating the composite derivative with a mean over derivatives, so that

$$\frac{\partial}{\partial x}(g(x,y)\frac{\partial u}{\partial x}) \approx \frac{1}{\Delta x^2}\left[g_{i+1/2,j}(u_{i+1,j}^{n} - u_{i,j}^{n}) - g_{i-1/2,j}(u_{i,j} - u_{i-1,j}^{n})\right] = [D_x(\bar{g}_{x,y} D_x u)]_{i,j}^{n}\,, \tag{7}$$

where

$$g_{i\pm 1/2,j} = \frac{1}{2}(g_{i,j} + g_{i\pm 1,j}), \tag{8}$$

is the arithmetic mean. A similar numerical approximation is made for the derivate wrt. the y-variable for the composite derivative.

The numerical approximations needed are now described, and a numerical scheme for the 2D-wave solution can now be obtained. Eq. 1 becomes

$$[D_tD_tu]^n_{i,j} + b\,[D_{2t}u]^n_{i,j} = [D_xD_xu]^n_{i,j} + [D_yD_yu]^n_{i,j} + [D_x(\bar{g_{x,y}}D_xu)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_yu)]^n_{i,j} + f^n_{i,j}. \tag{9}$$

II.II.1 Wave equation with damping

Now, by writing out the left hand side and putting the terms in a common denominator yields

$$\frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2} + \frac{u^{n+1}_{i,j} - u^{n-1}_{i,j}}{2\Delta t} = \tag{10}$$
$$\frac{1}{2\Delta t^2}\left[(2u^{n+1}_{i,j} - 4u^n_{i,j} + 2u^{n-1}_{i,j}) + b\Delta t u^{n+1}_{i,j} - b\Delta t u^{n-1}_{i,j}\right]$$
$$= \frac{1}{2\Delta t^2}\left[u^{n+1}_{i,j}(2 + b\Delta t) - 4u^n_{i,j} + u^{n-1}_{i,j}(2 - b\Delta t)\right].$$

Moving everything that is not $u^{n+1}_{i,j}$ over to the right hand side yields

$$u^{n+1}_{i,j} = \frac{1}{2 + b\Delta t}\left[\left(4u^n_{i,j} + u^{n-1}_{i,j}(b\Delta t - 2)\right) + 2\Delta t^2([D_x(\bar{g_{x,y}}D_xu)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_yu)]^n_{i,j} + f^n_{i,j})\right]. \tag{11}$$

The above equation is the general scheme which advances the solution one step forward in time for all mesh points i and j. However, a special first step is needed, because $u^{-1}_{i,j}$ is unknown, in which case the IC u_t (eq. 3) becomes useful. Applying a central difference scheme to the IC yields, for $n = 0$

$$u_t = \frac{u^1_{i,j} - u^{-1}_{i,j}}{2\Delta t} = V_{i,j}. \tag{12}$$

Rewriting the equation results in

$$u^{-1}_{i,j} = u^1_{i,j} - 2\Delta t V_{i,j}. \tag{13}$$

Inserted into eq. 11 and moving all $u^1_{i,j}$ over to the left hand side yields

$$\left(1 - \frac{b\Delta t - 2}{2 + b\Delta t}\right)u^1_{i,j} = \frac{4}{2 + b\Delta t}u^1_{i,j}, \tag{14}$$

All together

$$u^1_{i,j} = \frac{1}{4}\left[4u^0_{i,j} - 2\Delta t V_{i,j}(b\Delta t - 2) + 2\Delta t^2([D_x(\bar{g_{x,y}}D_x u)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_y u)]^n_{i,j} + f^n_{i,j})\right]. \tag{15}$$

Eqs. 11 and 15 are the essential schemes for solving a wave equation with damping coefficient $b \neq 0$.

II.II.2 Wave equation without damping

A 2D wave equation with no damping ($b = 0$), reduces the numerical equality (eq. 9) to

$$[D_t D_t u]^n_{i,j} = [D_x D_x u]^n_{i,j} + [D_y D_y u]^n_{i,j} + [D_x(\bar{g_{x,y}}D_x u)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_y u)]^n_{i,j} + f^n_{i,j}. \tag{16}$$

As a result of no first order terms in the time derivative, a non-damped wave has a different discretized version. One can easily use the same approximation as described above (sec. II.I) and take the same approach to rewriting the equations as seen in sec. II.II.1. This leads to the first step,

$$u^1_{i,j} = u^0_{i,j} - V_{i,j}\Delta t + \frac{\Delta t^2}{2}\left([D_x(\bar{g_{x,y}}D_x u)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_y u)]^n_{i,j} + f^n_{i,j}\right). \tag{17}$$

Whereas the general step becomes

$$u^{n+1}_{i,j} = 2u^n_{i,j} - u^{n-1}_{i,j} + \Delta t^2\left([D_x(\bar{g_{x,y}}D_x u)]^n_{i,j} + [D_y(\bar{g_{x,y}}D_y u)]^n_{i,j} + f^n_{i,j}\right). \tag{18}$$

II.II.3 Implementing Neumann Boundary Conditions

A homogeneous Neumann boundary condition require that

$$\frac{\partial u}{\partial n} \equiv \overrightarrow{n} \cdot \nabla u = 0, \tag{19}$$

at the boundary. In essence, it demands no net flux out of the boundaries. At the boundary $i = 0$ then

$$[D_{2x}u)]^n_{i=0,j} = \frac{u^n_{i=-1,j} - u^n_{i=1,j}}{2\Delta x} = 0, \tag{20}$$

which leads to the restriction

$$u_{i=-1,j} = u^n_{i=1,j}. \tag{21}$$

Similarly, at the boundary $i = Nx$, then

$$u^n_{i=N_x+1,j} = u^n_{N_x-1,j}. \tag{22}$$

The same requirements must be met at boundaries $j = 0$ and $j = N_y$ as well. These relations becomes important when evaluation u at these discrete mesh points. In the implementation of the finite difference scheme, one can either replace values of physical mesh points at the boundaries. However, another way to circumvent this problem is to include the points $i = \{-1, N_x + 1\}$ and $j = \{-1, N_y + 1\}$ as ghost cells to the mesh. The numerical solution u then has dimensions $\mathbb{C}^{N_x+3\times N_y+3}$.

II.III Method of Manifactured Solution and Convergence Rates

A way to verify correct implementation is to use the method of manifactured solutions (MMS) and check convergence rates.

Choosing an exact solution (u_e) to the PDE, the MMS procedure entails fitting the source term, boundary conditions and and initial condition to be consistent with the chosen solution. By inserting the chosen u_e into the PDE, these terms can be decided.

Thereafter a convergence rate test can be made, to verify that the numerical solution u corresponds with u_e up to a chosen tolerance. The convergence rate describes by which exponent the error norm (E) is expected to decrease by a decrease in $\Delta t, \Delta x$ and Δy. Making the analysis easier with several parameters, one defines a relation to a single discretization parameter. Mathematically, let $h = \Delta t = k\Delta x$ where $k = C/\sqrt{\|g\|}$ ($C \in [0, 1]$), so that

$$\Delta x = \Delta t \frac{\sqrt{\|g\|}}{C} = \Delta y. \tag{23}$$

Thus if $e^n_{i,j} = u_e(x_i, y_j, t_n) - u^n_{i,j}$, then let the error norm be

$$E = \|e^n_{i,j}\|_{l,\infty} = \max_i \max_j \max_n |e^n_{i,j}|. \tag{24}$$

By choosing an common parameter h, then mathematically E should go as

$$E = \hat{K}h^r, \tag{25}$$

where $\hat{K}$ is a constant. For N experiments $m = \{0, 1, ..., N\}$, then

$$r_i = \frac{\ln(E_{i+1}/E_i)}{\ln(h_{i+1}/h_i)}, \tag{26}$$

for $i = \{0, 1, ..., m-2\}$, where r is the convergence rate. The error is expected to be of order 2 for finite difference methods, so that $r = 2$. If the verification comes close to this value for r, the implementation reproduces the desired exact solution up to a given tolerance.

The Courant number, C, can be chosen so that the stability criteria

$$\Delta t \leq \frac{1}{\sqrt{\|g\|}} \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)^{-1/2} \tag{27}$$

is fulfilled.

III Exploration of the Implementation and Results

III.I MMS and verification

III.I.1 Constant Solution

A solution $u_e(x, y, t) = U$ is used to verify the implementation through a test function, which is callable through the commandline argument `pytest`. f, b, I and V is determined for $g(x, y, t) = 1$ so that u_e fullfills the PDE.

From the initial conditions (eq. 3),

$$I = u_e|_{t=0} = U, \tag{28}$$

and

$$V = \left(\frac{\partial u_e}{\partial t} \right)_{t=0} = 0 \tag{29}$$

For a constant solution the 2D-wave equation (eq. 1) yields $f = 0$, since all derivatives equal zero. The damping coefficient (b) can then also be set to 0. Since the finite difference method is exact up to second order equations, the numerical scheme should be exact at mesh points. The test function `test-constant-solution()` verifies correct implementation through pytest or by running the programme directly gives output:

```
Maximum error in computing a constant solution 0.000e+00.
```

However, introducing bugs in the code, such as updating $u^{n-1}_{i,j}$ to $u^{n}_{i,j}$ or using wrong indexing for the boundary condition $u^n_{N_x-1,j} = u[N_x + 1]$ (see eq. 22) gives an error message in with `pytest` and a non-zero maximum error.

III.I.2 Standing undamped waves

Verification of the programme is performed for standing undamped ($b = 0$) waves with no source term ($f = 0$). It has exact solution

$$u_e(x, y, t) = A\cos(k_x x)\cos(k_y y)\cos(\omega t), \qquad k_x = \frac{m_x \pi}{L_x}, \;\; k_y = \frac{m_y \pi}{L_y}, \tag{30}$$

where m_x and m_y are arbitrary constants. Now, the MMS procedure can be applied to obtain parameters I, ω and V so that u_e fullfills the PDE.

For $g(x, y) = constant$, the PDE reduces to

$$\frac{\partial^2 u}{\partial t^2} = g\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right). \tag{31}$$

Setting $g = 1$ and inserting u_e into the PDE yields

$$\frac{\partial^2 u_e}{\partial t^2} = -\omega^2 A\cos(k_x x)\cos(k_y y)\cos(\omega t), \tag{32}$$

$$\frac{\partial^2 u_e}{\partial x^2} = -k_x^2 A\cos(k_x x)\cos(k_y y)\cos(\omega t), \tag{33}$$

and

$$\frac{\partial^2 u_e}{\partial y^2} = -k_y^2 A\cos(k_x x)\cos(k_y y)\cos(\omega t). \tag{34}$$

Equating the RHS and LHS, then $\omega = \sqrt{k_x^2 + k_y^2}$. Applying the ICs (eq. 3) on eq. 30, then

$$I = u_e|_{t=0} = A\cos(k_x x)\cos(k_y y), \tag{35}$$

and

$$V = \left(\frac{\partial u_e}{\partial t}\right)_{t=0} = -\omega A\cos(k_x x)\cos(k_y y)\sin(\omega t)|_{t=0} = 0. \tag{36}$$

One cannot expect the numerical be exact on mesh points for standing waves, due to truncation error of order $O(h^3)$. Hence, in this instance, a convergence test is suitable. A convergence test is run through the `main.py` file, which for $N = 3$ experiments gives the output

```
stability criteria dt:  0.5 dx:  0.8333 dy 0.8333 dt less
than 0.5892 [...]
```

```
r [2.0054868211844754, 2.0011721706428913]
Expected converge rate:  2.  tol:  0.05
Expected convergence rate after experiment:  1 , with h:
5.0000e-01.
```

Thus, with an initial $h = \Delta t = 0.5$ that is halved for each experiment, the expected convergence rate is reproduced immediately within the given tolerance.

III.I.3 Waves with damping and variable wave velocity

As a last verification a 2D wave with damping ($b \neq 0$) and variable wave velocity (g(x,y)) with a source term (f(x,y)) is used. The PDE then becomes the general equation with all terms included (eq. 1).

An exact solution which solves this PDE is

$$u_e = [A\cos(\omega t) + B\sin(\omega t)]\, e^{-dt} \cos(k_x x)\cos(k_y y), \tag{37}$$

with A,B,d arbitrary constants, and k_y and k_x defined as in sec. III.I.2. Using the ICs (eq.3), then

$$I = u_e|_{t=0} = A\cos(k_x x)\cos(k_y y), \tag{38}$$

and

$$V = \left(\frac{\partial u_e}{\partial t}\right)_{t=0} = -e^{-dt}\left[(A\omega + Bd)\sin(\omega t) + (Ad - B\omega)\cos(\omega t)\right]|_{t=0} = Ad - B\omega. \tag{39}$$

One can determine the source term f by rearranging terms in the original PDE, so that

$$\frac{\partial^2 u}{\partial t^2} + b\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(g(x,y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(g(x,y)\frac{\partial u}{\partial y}\right) = f(x,y,t). \tag{40}$$

Setting $g(x,y)$ as a spatially dependent wave velocity, f is determined by using the module `sympy`'s `diff` and `lamdify`.

Running the test for given parameter values, however, does not give the expected convergence rate. The output from the running convergence test, for a damped wave with variable wave velocity, from `main.py` is

```
stability criteria dt:  0.5 dx:  5.27 dy 5.27 dt less
than 1.17 [...]
```

```
E [0.973, 1.24, 1.58, 1.785, 1.88, 1.944]
r [-0.3550, -0.34, -0.17, -0.082, -0.0413]
Expected converge rate:  2.  tol:  0.05 [...]
```

III.II Propagation over a Subsea Hill

An investigation is made of properties of the numerical solution for a wave propagating over a subsea hill. Let $b = 0$ (damping) and $f = 0$, and the Neumann condition $\partial u/\partial n = 0$ implying waves are reflected perfectly at the boundary. To describe the height of the subsea hill, three different surfaces are chosen:

$$\text{1)Smooth hill: } B(x,y) = B_0 + B_a \exp\left(-(\frac{x - B_{mx}}{B_s})^2 - (\frac{y - B_{my}}{bB_s})^2\right). \tag{41}$$

$$\text{2)Less smooth hill: } B(x,y) = B_0 + B_a \cos\left(\frac{\pi x - B_{mx}}{2Bs}\right) \cos\left(\frac{\pi y - B_{my}}{2B_s}\right), \tag{42}$$

when $0 \leq \sqrt{(x - B_{mx})^2 + (y - B_{my})^2} \leq B_s$ and $B = B_0$ outside.

$$\text{3)Rectangle hill shape: } B(x,y) = B_0 + B_a, \tag{43}$$

for $B_{mx} - B_s \leq x \leq B_{mx} + B$, $B_{my} - bB_s \leq y \leq B_{my} + bB_s$.

An overview of the parameters in the above equations are given below (tab 1).

Table 1: **Parameter overview:** Relevant parameters for the IC and the different subsea hill surfaces (eqs.41-42).

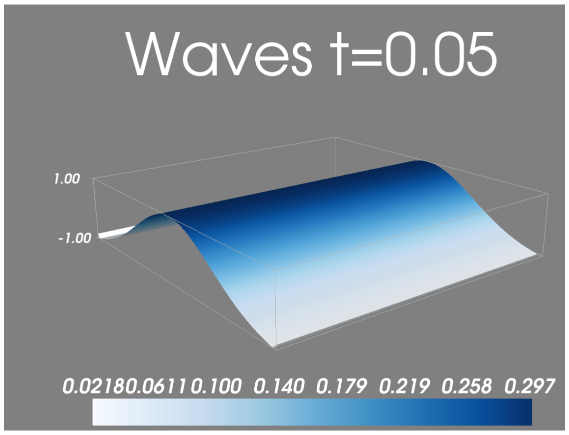
B_0	B_a	B_s	B_{mx}	B_{my}	b	H_0	I_a
2	$0.3 \cdot B_0$	$L_x \cdot 0.3$	$0.5 \cdot L_x$	$0.5 \cdot L_y$	1.1	$3 \cdot B0$	$0.15 \cdot B0$

Then, if depth of the seafloor is H_0 for a flat surface, the hill depth becomes $H(x,y) = H_0 - B(x,y)$. Now, the wave velocity can be described function of $H(x,y)$, namely $g(x,y) = g_a H(x,y)$. $g_a$is the gravitational constant on earth ($g_a = 9.81$).

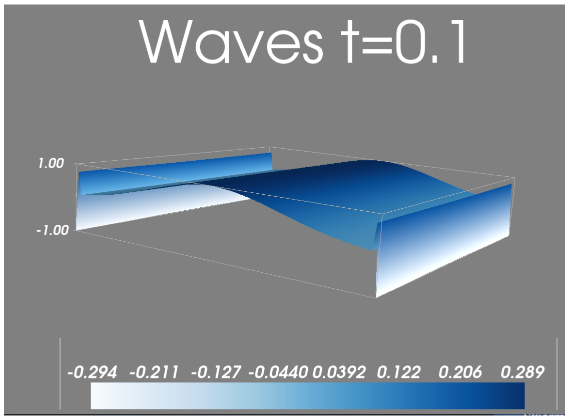
The initial wave is set to

$$I(x,y) = I_a \exp\left(\left(\frac{x - B_{mx}}{B_s}\right)^2\right). \tag{44}$$

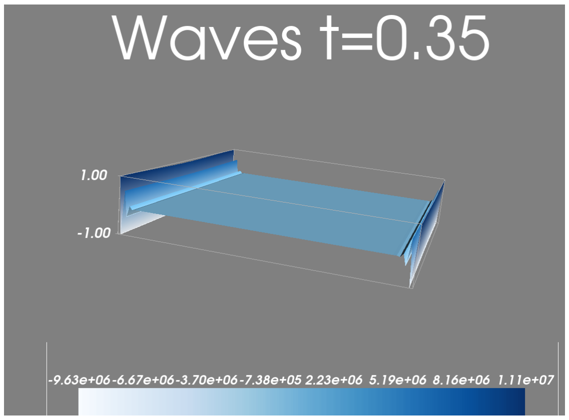
The resulting waves are plotted as snapshots in time below, for the three given surfaces (eqs. 41-42, fig. 1-3).



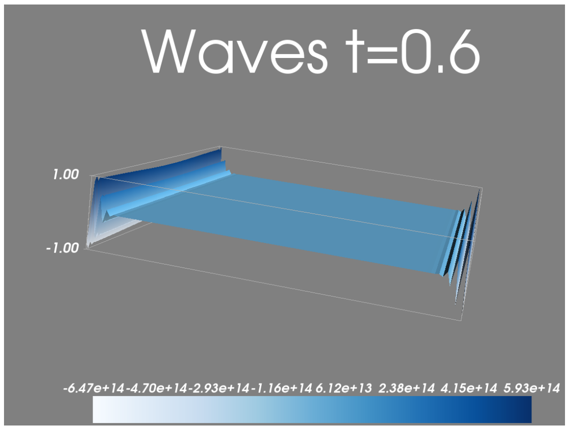
(a)



(b)

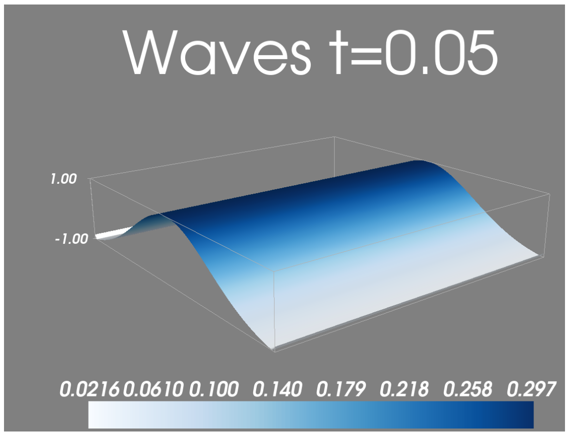

Waves t=0.35
1.00
-1.00
-9.63e+06 -6.67e+06 -3.70e+06 -7.38e+05 2.23e+06 5.19e+06 8.16e+06 1.11e+07

(c)

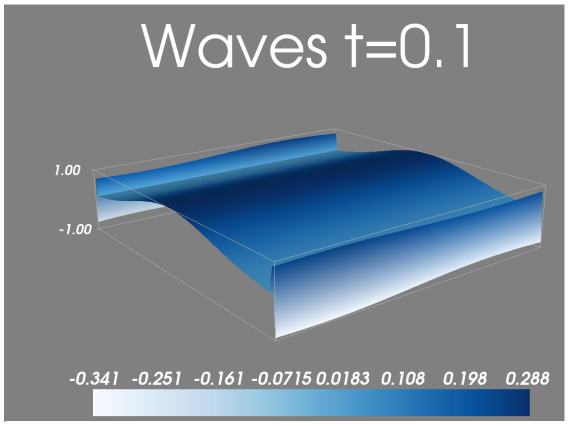

Waves t=0.6
1.00
-1.00
-6.47e+14 -4.70e+14 -2.93e+14 -1.16e+14 6.12e+13 2.38e+14 4.15e+14 5.93e+14

(d)

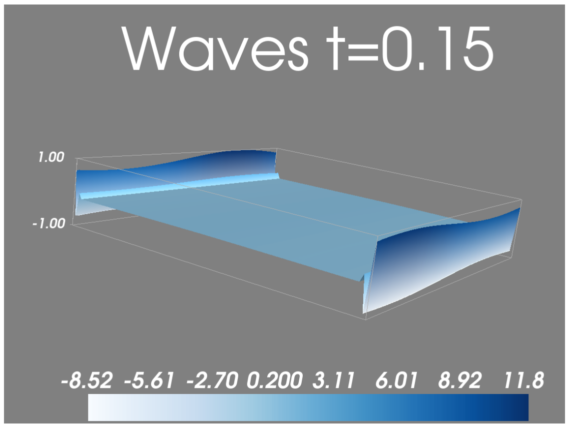
Figure 1: Numerical results for the 2D-wave equation, for a smooth subsea hill (case 1), presented as snapshots over time. Made with library `Mayavi`



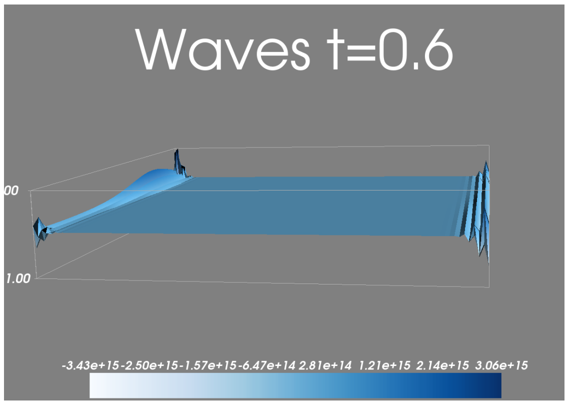
(a)



(b)

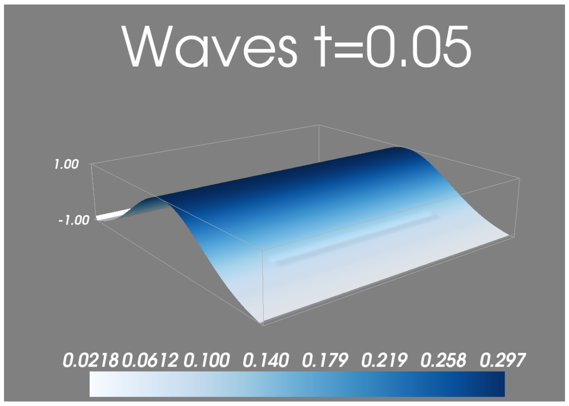


(c)

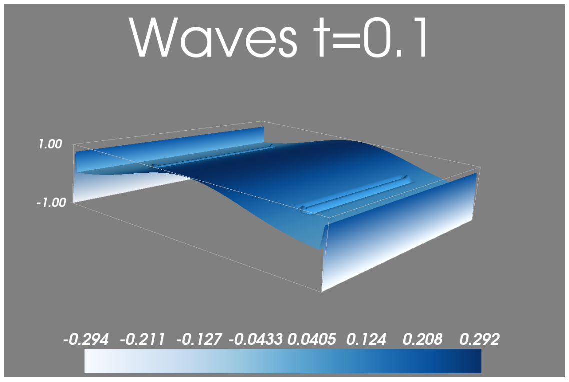

Waves t=0.6
00
1.00
-3.43e+15 -2.50e+15 -1.57e+15 -6.47e+14 2.81e+14 1.21e+15 2.14e+15 3.06e+15

(d)

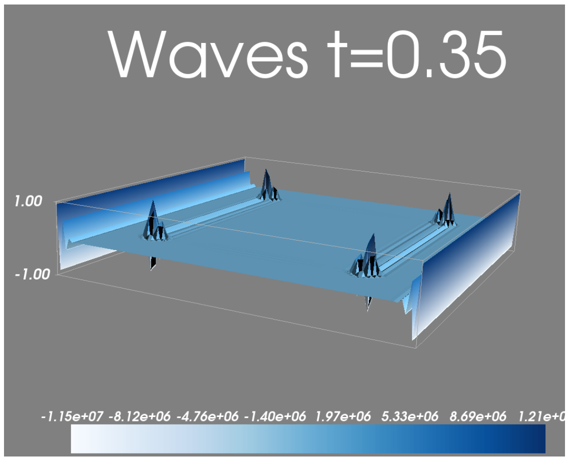
Figure 2: Numerical results for the 2D-wave equation, for a less smooth subsea hill (case 2), presented as snapshots over time. Made with library `Mayavi`



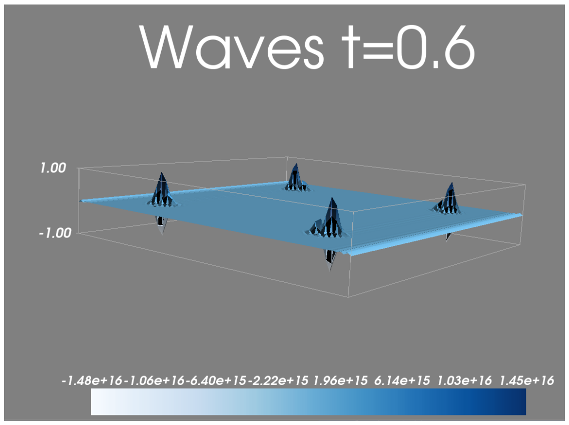
(a)



(b)


Waves t=0.35
1.00
-1.00
-1.15e+07 -8.12e+06 -4.76e+06 -1.40e+06 1.97e+06 5.33e+06 8.69e+06 1.21e+0

(c)


Waves t=0.6
1.00
-1.00
-1.48e+16 -1.06e+16 -6.40e+15 -2.22e+15 1.96e+15 6.14e+15 1.03e+16 1.45e+16

(d)

Figure 3: Numerical results for the 2D-wave equation, for a rectangular subsea hill (case 3), presented as snapshots over time. Made with library `Mayavi`.

IV Discussion

The results obtained by solving a 2D-wave equation, using a finite difference scheme, is discussed in the context of verification, convergence rates and numerical artefacts. First, the performance of the verification and convergence rates are discussed. Then, commentary is made on numerical artefacts, followed by suggestions on to possible improvements in the implementation.

IV.I Verification and Convergence tests

In accordance to what is expected, the verification with a constant solutions gives an maximum error of 0. It implicates a correct implementation in the case of $b = 0$, and that the analytical solution is reproduced exactly up to machine precision [3]. Introducing bugs in the code by choice gives an error message when the programme is run with `pytest`, showcasing the importance of verifying codes through unit tests.

The convergence test on a 2D standing wave without damping results in convergence rate of $r = 2$, which is consistent with the expected convergence rate of a finite difference method [3]. In contrast, a wave with damping and variable wave velocity does not reproduce the expected convergence rate described from the literature. Judging from the output, there are three possible errors:

1) Insufficient or incorrect use of the MMS procedure
2) A faulty implementation for the numerical scheme when $b \neq 0$,
3) A swapping of E and r might somehow occurre, since E seems to converge towards 2.

IV.II Waves over a Subsea Hill: Numerical artefacts

The numerical solution to waves over subsea hills, shows how numerical artefacts accumulate over time for different subsea surfaces (fig. 1-3). For a smooth subsea hill (case 1), the wave is reflected perfectly back from the surface. However, for a steeper hill, numerical artefacts arise faster with time, such as seen in case 2 at $t = 0.6$, and the rectangular, more dramatic, hills in case 3 at $t = 0.35$. Thus, the finite difference scheme described here, is not suitable to correctly illustrate how waves respond to sharper and less smooth surfaces.

IV.III Suggested improvements

The implementation is performed through object orientation, making the different methods available and easy to read. Programmes can easily be run through the `main.py` file.

However, an obvious improvement would be to make the convergence test work for a wave with damping and a non-constant wave velocity.

Another improvement would be to make an animation or GIF. This could be done using stored PNG files, and `ffmpeg` to create videos through commandline arguments. An option to store images is implemented in the code. Time restrictions did not allow for more refinement on these parts of the code.

V Concluding Remarks

Through modern computing, many problems which was previously solved analytically, and problems who do not have an analytical solution, can be solved through numerical schemes. A numerical scheme using the finite difference method is here applied on a general 2D wave equation.

The numerical scheme is correctly verified for a constant solution, and reproduce the theoretical convergence rate $r = 2$ for a 2D standing wave with no damping. However, it fails to reproduce the convergence rate for a 2D-wave with damping and variable wave velocity, due to a bug in the implementation, as discussed above. Waves over three cases of subsea hills shows how numerical artefacts can accumulate over time, depending on how the wave is initialized and if the wave meet steep or sharp edge surfaces.

References

[1] Gene H. Golub and Henk A. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, pages 35–65, 2000.

[2] Morten Hjorth-Jensen. Computational Physics Lectures: Eigenvalue problems, 2020. Accessed: September 2020.

[3] Hans Petter Langtangen; Svein Linge. *Finite Difference Computing with PDEs: A Modern Software Approach*. SpringerOpen, 2017.

[4] Cai X.; Langtangen H.P. IN5270 Recourses, 2020. Accessed: September 2020.