Manual for ***CRestRelax.py***

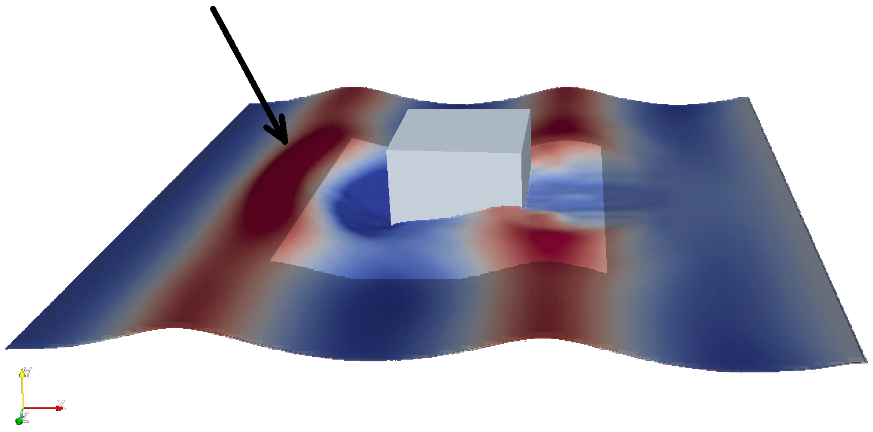
a computer program for
estimating reflection coefficient C_R for implicit relaxation zones
in flow simulations with free-surface wave propagation

written by Robinson Perić

at the Institute for Fluid Dynamics and Ship Theory,
Hamburg University of Technology (TUHH), Germany
and at the Fakultet Strojarstva i Brodogradnje,
Sveučilište u Zagrebu, Croatia

version March 11, 2021

relaxation zones (shaded gray) gradually blend-out the governing equations and blend-in the far-field wave solution



Contents

1 One-minute-explanation of the code

When to use this code?

- You perform flow simulations of free-surface wave propagation
- You want to minimize undesired wave reflections at the domain boundaries
- To do so, you want to use *implicit relaxation zones* (cf. Sect. 5), which blend the flow solution towards the far-field wave in a zone usually attached to the vertical domain boundaries

What does the code do?

- The user can enter wave period T, water depth h, wavelength λ, zone thickness, blending function $b(\mathbf{x})$, and blending exponent n
- The code calculates reflection coefficient C_R[1] for various values of relaxation parameter τ according to [19] and writes these results to the file ′C_R.csv′ in the same folder
- If matplotlib is installed: A window will open with an interactive plot of C_R over τ
- Thus the code can be used to quickly fine-tune the case-dependent relaxation-zone parameters

Requirements:

- Check that your relaxation zone can be formulated in terms of Eqs. (5) and (6)
- Install *python* programming language
- For full functionality, install *matplotlib*
- Operating systems: Linux, macOS, Windows

Optimizing implicit relaxation zones:

- Increasing the zone thickness x_d tends to lower reflection coefficient C_R and to widen the range of wavelengths λ for which reflections will be satisfactorily minimized
- Typically, confidence in minimizing reflections is more important than the last few percent efficiency $\Rightarrow$ use slightly thicker zones than necessary
- Common values for zone thickness are $1\lambda \leq x_\mathrm{d} \leq 2\lambda$
- For irregular waves, tune the zone to the peak period or the longest period (quick approach [16]) or calculate the reflection coefficient for each wave component's period and then tune accordingly (more accurate approach)

Benefits and Limitations:

- The code closely estimates the optimum relaxation-zone parameters for 2D- and 3D-flow simulations with regular, irregular, linear, and highly nonlinear waves for any water depth
- Additionally, the code provides an estimate of the upper-limit for the corresponding reflection coefficient C_R
- When implicit relaxation zones were optimized via the code, simulation results for reflection coefficient C_R were mostly lower or equal the code's predictions, but never more than 3.4% larger (cf. [15, 19])

[1]Reflection coefficient $C_\mathrm{R} = H_\mathrm{refl}/H$ with height H_refl and H of reflected and generated wave, $C_\mathrm{R} = 1$ for full wave reflection and $C_\mathrm{R} = 0$ for perfect wave absorption.

- For complex flows, reflection coefficients may be a few percent larger than predicted, so thicker zones ($x_\mathrm{d} \geq 1.5\lambda$) are recommended to ensure satisfactory wave absorption
- Undesired reflections can also be due to other mechanisms, e.g. the use of inappropriate grids
- Therefore, tuning the relaxation-zone parameters according to this code does NOT guarantee that the actual reflection coefficient in the simulation will equal the prediction

2 Requirements

The programming language python version 2.7 or 3.0 or higher (`https://www.python.org/downloads/`) must be installed.

It is recommended to also have matplotlib (`https://matplotlib.org/users/installing.html`) installed[2]. Then the code will open a window with an interactive plot of the results.

3 What does the code do?

Implicit relaxation zones can reduce undesired wave reflections at boundaries of the computational domain. However, their source terms contain user-defined parameters. These parameters are case-dependent and must be optimized for every simulation, or else the substantial wave-reflection can occur at the relaxation zone.

The present code can be used to optimize these case-dependent parameters *before performing the flow simulation.* The code is an implementation of the analytical approach from [19], which predicts the reflection coefficient C_R for a given relaxation-zone setup. The reflection coefficient is $C_\mathrm{R} = H_\mathrm{refl}/H$ with wave heights H_refl and H of reflected and generated wave. For practical discretizations ($\gtrsim$ 30 cells per wavelength), implicit relaxation zones were found to behave basically independent of the discretization and the used flow solver.

The code's predictions for reflection coefficient and optimum relaxation-zone parameters were demonstrated to be of satisfactory accuracy in 2D- and complex 3D-flow simulations with regular, irregular, linear, and highly nonlinear waves in both shallow, intermediate and deep water depths (cf. [19, 15, 17, 18]). When implicit relaxation zones were optimized via the code, simulation results for reflection coefficient C_R were mostly lower or equal the code's predictions, but never more than 3.4% larger (cf. [15, 19]). Thus the present code can be recommended for practical 3D-flow simulations with relaxation zones featuring nonlinear wave phenomena.

4 Does the code *CRestRelax.py* apply to the relaxation zones implemented in my flow-solver?

Various different approaches have been presented for wave-generation and wave-damping via domain-internal source terms. These can be classified into: *implicit relaxation zones*, *explicit relaxation zones* and *forcing zones.*

The following brief overview may be helpful to determine which approach is implemented in the code you are using.

[2]For debain-based linux systems, install e.g. via: `sudo apt-get install python3-matplotlib`.

Implicit relaxation zones (implemented e.g. in Naval Hydro Pack, cf. [6, 15, 19, 22, 23, 24]) introduce source terms in the governing equations to blend, say a general transport equation $\mathcal{T}$ for transport quantity ϕ, over to a reference solution ϕ_{ref} via

$$(1 - b(\mathbf{x}))\,\mathcal{T} + \frac{b(\mathbf{x})}{\tau}\mathcal{R} = 0 \quad , \tag{1}$$

where $b(\mathbf{x})$ is a blending function such as Eq. (12), $\mathcal{T}$ corresponds e.g. to the conservation equations for fluid momentum or volume fraction, and $\mathcal{R}$ corresponds to $\int_V (\phi - \phi_{\mathrm{ref}})\ \mathrm{d}V$. Implicit relaxation zones are described in more detail in Sect. 5. **The present code applies to implicit relaxation zones only!**

Explicit relaxation zones (implemented e.g. in waves2Foam, cf. [5]) modify the fields for volume fraction α and velocity $\mathbf{u}$ by replacing computed values ϕ_{computed} by

$$\phi = (1 - b(\mathbf{x}))\phi_{\mathrm{target}} + b(\mathbf{x})\phi_{\mathrm{computed}} \quad , \tag{2}$$

where $b(\mathbf{x})$ is a weighting function and ϕ_{target} is the target solution. This modification is performed in each time-step, e.g. prior to the solution of the pressure-velocity coupling (cf. [5]). Among the most influential implementations of explicit relaxation zones are [2, 3, 5, 10, 11]; further references and comparison to other wave-generation and wave-damping approaches can be found e.g. in [9, 20, 26, 27]. Case-dependent parameters are the zone thickness x_{d}, the blending function $b(\mathbf{x})$ and possibly also the time-step Δt. In contrast to implicit relaxation zones, it is not directly apparent to which source terms in the governing equations the 'explicit' manipulation of the flow field corresponds. At present, no analytical approach exists for optimizing explicit relaxation zones.

Forcing zones (implemented e.g. in ANSYS Fluent, Siemens STAR-CCM+ and many other flow-solvers, cf. corresponding software manuals) add source terms on the right-hand side of the conservation equations. [15] showed that various approaches, including 'absorbing layers' (e.g. [25]), 'damping zones' (e.g. [13, 14]), 'dissipation zones' (e.g. [12]), 'numerical beaches' (e.g. [21]), 'sponge layers' (e.g. [1, 4, 8]) or the 'Euler overlay method' (e.g. [7]), can be formulated as special cases of *forcing zones*. To gradually force velocity u_i and volume fraction α towards a prescribed reference solution, $u_{i,\mathrm{ref}}$ and α_{ref}, near the domain boundaries, apply the following source terms:

$$q_i = \int_V \rho\gamma b(\mathbf{x})(u_{i,\mathrm{ref}} - u_i)\ \mathrm{d}V \quad , \tag{3}$$

$$q_\alpha = \int_V \gamma b(\mathbf{x})\,(\alpha_{\mathrm{ref}} - \alpha)\ \mathrm{d}V \quad , \tag{4}$$

with volume V and fluid density ρ. The case-dependent parameters of forcing zones are the zone thickness x_{d}, the forcing strength γ, which regulates the source-term magnitude, and the blending function $b(\mathbf{x})$, which regulates how the source-term magnitude varies within the zone. The optimum values of these parameters can be determined analytically (cf. [15, 17, 18]. A computer program to optimize the forcing zone's parameters is available as free software under: `https://github.com/wave-absorbing-layers/relaxation-zones-for-free-surface-waves`
Under the above link, a more detailed description of forcing zones can be found.

5 Introduction to implicit relaxation zones

Hence the conservation equations for momentum and volume fraction take the form

$$\begin{aligned}(1-b(\mathbf{x}))\Bigg[\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho u_i \;\mathrm{d}V + \int_S \rho u_i(\mathbf{v}-\mathbf{v}_g)\cdot\mathbf{n}\;\mathrm{d}S \\ -\int_S(\tau_{ij}\mathbf{i}_j - p\mathbf{i}_i)\cdot\mathbf{n}\;\mathrm{d}S - \int_V \rho\mathbf{g}\cdot\mathbf{i}_i\;\mathrm{d}V\Bigg] \\ +\frac{b(\mathbf{x})}{\tau}\left[\int_V \rho\left(u_i - u_{i,\mathrm{ref}}\right)\;\mathrm{d}V\right] = 0 \quad ,\end{aligned} \tag{5}$$

$$\begin{aligned}(1-b(\mathbf{x}))\left[\frac{\mathrm{d}}{\mathrm{d}t}\int_V \alpha\;\mathrm{d}V + \int_S \alpha(\mathbf{v}-\mathbf{v}_\mathrm{g})\cdot\mathbf{n}\;\mathrm{d}S\right] \\ +\frac{b(\mathbf{x})}{\tau}\left[\int_V (\alpha-\alpha_\mathrm{ref})\;\mathrm{d}V\right] = 0 \quad ,\end{aligned} \tag{6}$$

with volume V of control volume (CV) bounded by the closed surface S, fluid velocity $\mathbf{v}= (u_1,u_2,u_3)^\mathrm{T} = (u,v,w)^\mathrm{T}$, grid velocity $\mathbf{v}_g$, unit vector $\mathbf{n}$ normal to S and pointing outwards, time t, pressure p, fluid density ρ, components τ_{ij} of the viscous stress tensor, unit vector $\mathbf{i}_j$ in direction x_j, and volume fraction α of the liquid phase, reference velocities $u_{i,\mathrm{ref}}$ and reference volume fraction α_ref.

The relaxation parameter τ regulates the magnitude of the source term in such a way that a large value of τ implicates a small source term and vice versa. Note that τ has sometimes been interpreted as a numerical stability parameter, and thus has occasionally been omitted[3] from Eq. (1), so in literature Eq. (1) is sometimes written in the following form: $(1-b(\mathbf{x}))\,\mathcal{T} + b(\mathbf{x})\mathcal{R} = 0$. However, dimensional analysis shows that τ has the unit $[s]$, so it is recommended to include τ in the equation as in Eq. (1).

The blending function $b(\mathbf{x})$ regulates how the magnitude of the source term varies within the relaxation zone. Outside the relaxation zone holds $b(\mathbf{x}) = 0$, and inside the relaxation zone holds $0 \le b(\mathbf{x}) \le 1$. Many different types of blending functions can be applied. The following common choices are implemented in the code:

Constant blending

$$b(\mathbf{x}) = 0.5 \quad , \tag{7}$$

linear blending

$$b(\mathbf{x}) = \left(\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}}\right) \quad , \tag{8}$$

quadratic blending

$$b(\mathbf{x}) = \left(\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}}\right)^2 \quad , \tag{9}$$

$\cos^2$-blending

$$b(\mathbf{x}) = \cos^2\left(\frac{\pi}{2}+\frac{\pi}{2}\left(\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}}\right)\right) \quad , \tag{10}$$

[3] If the description of your code does not include τ, then it is probably set internally to $\tau = 1\,\mathrm{s}$ or to some other default value; the default value in the Naval Hydro Pack is $\tau = \Delta t$ with time-step Δt . With a look at Eqs. (1–6), the reader should have all tools available to judge whether this is so and, if necessary, to modify the source code accordingly so that the default value of τ can be adjusted by the user, which is necessary to obtain optimum tuning of the relaxation zone's parameters.

exponential blending

$$b(\mathbf{x}) = \left(\frac{e^{((x_\mathrm{d}-\tilde{x})/x_\mathrm{d})^2} - 1}{e^1 - 1} \right) \quad , \tag{11}$$

power blending

$$b(\mathbf{x}) = \left(\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}} \right)^n \quad , \tag{12}$$

exponential blending with power n

$$b(\mathbf{x}) = \left(\frac{e^{((x_\mathrm{d}-\tilde{x})/x_\mathrm{d})^n} - 1}{e^1 - 1} \right) \quad , \tag{13}$$

and $\cos^{2n}$-blending

$$b(\mathbf{x}) = \left[\cos^2 \left(\frac{\pi}{2} + \frac{\pi}{2} \left(\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}} \right) \right) \right]^n \quad , \tag{14}$$

where $\tilde{x}$ is the shortest distance of location $\mathbf{x}$ to the closest domain boundary to which a relaxation zone of thickness x_d is attached, and n regulates the shape of the blending function. Some of these blending functions are illustrated in Fig. 1.

The analytical approach in [19] was derived so that it works for any continuous or discontinuous blending function. Custom blending functions can be entered at the location indicated in the source code.

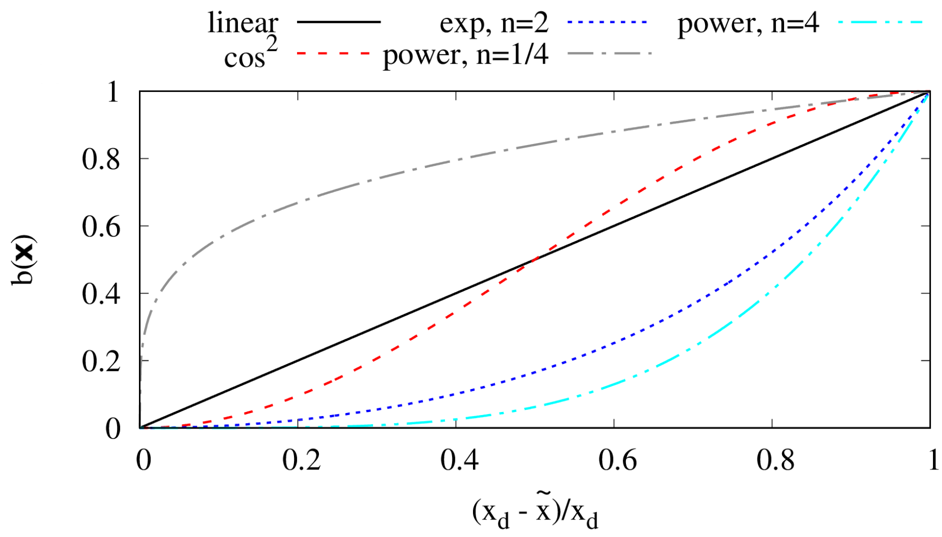


Figure 1: Different blending functions $b(\mathbf{x})$ over location in relaxation zone, with entrance to the relaxation zones at $\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}} = 0$ and boundary to which the relaxation zone is attached at $\frac{x_\mathrm{d}-\tilde{x}}{x_\mathrm{d}} = 1$; for blending via Eq. (12) with $n = 1$ ('linear'), $n = 1/4$ ('power, n=1/4') and $n = 4$ ('power, n=4'), blending via Eq. (10) with $n = 1$ ('$\cos^2$, n=1'), and blending via Eq. (11) with $n = 2$ ('exp, n=2')

Thus implicit relaxation zones have three user-defined parameters, which are case-dependent and have to be adjusted for each simulation: the relaxation parameter τ, the blending function $b(\mathbf{x})$ and the zone thickness x_d.

The present code *CRestRelax.py* can be used to predict the optimum values for these parameters. The code evaluates the analytical approach from [19], which was derived by extending the analytical approach for optimizing forcing zones (cf. [17]) to implicit relaxation zones. Thus, implicit relaxation zones can be considered as a special-case of forcing zones, although they differ as follows:

Whereas implicit relaxation zones ‘blend out’ the governing equations via the factor $(1 - b(\mathbf{x}))$, forcing zones do not have this factor. Thus with forcing zones, the whole governing equations remain active in the whole domain, whereas within relaxation zones as in Eqs. (5) and (6), the terms from the governing equations that are active in the solution domain of interest are faded out and the reference solution is faded in. Thus the solution at the domain boundary equals the prescribed reference solution, which is not necessarily so for forcing zones.

In contrast to forcing zones, where constant or linear blending functions were found to typically perform worse than the other blending functions, for implicit relaxation zones this is different: For example, constant blending with $b(\mathbf{x}) = 1$ should be avoided, since then the entrance to the relaxation zone would behave like a velocity boundary condition, i.e. may be fully reflective. Consider that the theory for predicting implicit relaxation-zone behavior was derived by reformulating Eqs. (5) and (6) as a forcing zone, which was done be multiplying both equations with $(1 - b(\mathbf{x}))^{-1}$. Therefore in relaxation zones, the simultaneous blending-out of one solution (the parts in Eqs. (5) and (6) which are multiplied by $(1 - b(\mathbf{x}))$) while blending-in another solution (the parts in Eqs. (5) and (6) which are multiplied by $\frac{b(\mathbf{x})}{\tau}$) can be interpreted as a forcing ‘with a different blending’, so the “effective” increase in source term magnitude is not directly proportional to $b(\mathbf{x})$, but rather to $b(\mathbf{x})/\,(1 - b(\mathbf{x}))$.

Because forcing zones and implicit relaxation zones are closely related, both approaches perform equally well when optimized. Thus the choice between using relaxation or forcing zones can be made based on which one is already implemented or which one is easier to implement in one’s flow solver.

See [19] for a detailed discussion.

6 Benefits and limitations

The code closely predicts the optimum relaxation-zone parameters (i.e. τ, x_d and $b(\mathbf{x})$), which was verified for various 2D- and 3D-flow simulations under a wide range of conditions, including deep-, intermediate- and shallow-water depth, (ir-)regular and (highly non-)linear waves (cf. [15, 16, 17, 18, 19].

Reflection coefficients C_R in flow simulations were typically lower or similar to the code’s predictions, but never more than a few percent larger when optimally tuned using the code.

Although the code’s predictions are usually quite accurate [19], please keep in mind that every theory has its limitations! Wave reflection in the simualtions may be a few percent larger than predicted analytically, especially when the relaxation-zone parameters are further away from their optimum values.

Thus it is recommended to select the zone thickness x_d larger than theoretically necessary to increase confidence in the prediction of the reflection coefficient C_R. This means, that although experienced users may obtain reflection coefficients of ca. $C_\mathrm{R} = 1 - 2\%$ with a zone thickness of $x_\mathrm{d} \leq 1\lambda$, it is generally recommended to use thicker zones, i.e. $x_\mathrm{d} \geq 1.5\lambda$.

Furthermore, undesired wave-reflections can be due other mechanisms as well, e.g. the use of inappropriate grids or certain relaxation-zone arrangements[4]. Therefore, tuning the relaxation zone's parameters according to the present theory does NOT guarantee that the actual reflection coefficient in the simulation will equal the prediction. See [19] and [15] for more details.

7 Recommendations for optimizing implicit relaxation zones

At the wave-maker, it is usually known which waves are generated. For a given wave period T, the resulting wavelength λ is a function of the water depth h and the wave steepness H/λ, thus T, h and H have to be entered in the code. In practice, a relaxation-zone thickness of $1\lambda \leq x_\mathrm{d} \leq 2\lambda$ is usually sufficient to obtain $C_\mathrm{R} \approx 1 - 2\%$.

Increasing the zone thickness x_d tends to widen the range of wavelengths which will be damped satisfactorily and to lower the reflection coefficient C_R at the optimum parameter setting; thus if the wave absorption is not satisfactory, then zone thickness x_d should be increased.

For irregular waves, a quick approach is to tune the zone to the peak period or the longest period of the wave energy spectrum[16]. A more accurate approach is to calculate the reflection coefficient $C_\mathrm{R}(T)$ for each wave component's period T, and thus obtain the spectrum of the reflected waves and in this manner optimize the zone's parameters accordingly (cf. [15]).

8 How to run the code

In windows, double click on the executable file *CRestRelax.py*.

In Linux and macOS, open a terminal and type:

```
python CRestRelax.py
```

Example output of the code is shown in Fig. 2.

Alternatively, the parameters can be passed as as arguments[5].

[4]For example, relaxation of fluid momentum and volume fraction towards the far-field wave solution can create flow disturbances in relaxation zones which lie tangential to the wave propagation direction, due to a mismatch between computed flow solution (including discretization and iteration errors) and the reference solution (without discretization and iteration errors). Such flow disturbances can be radiated as undesired waves into the domain. If the reference solution is highly accurate, these disturbances vanish on fine grids; however, finer grids than commonly used may be required. Several solutions to this problem have been presented, see [18].

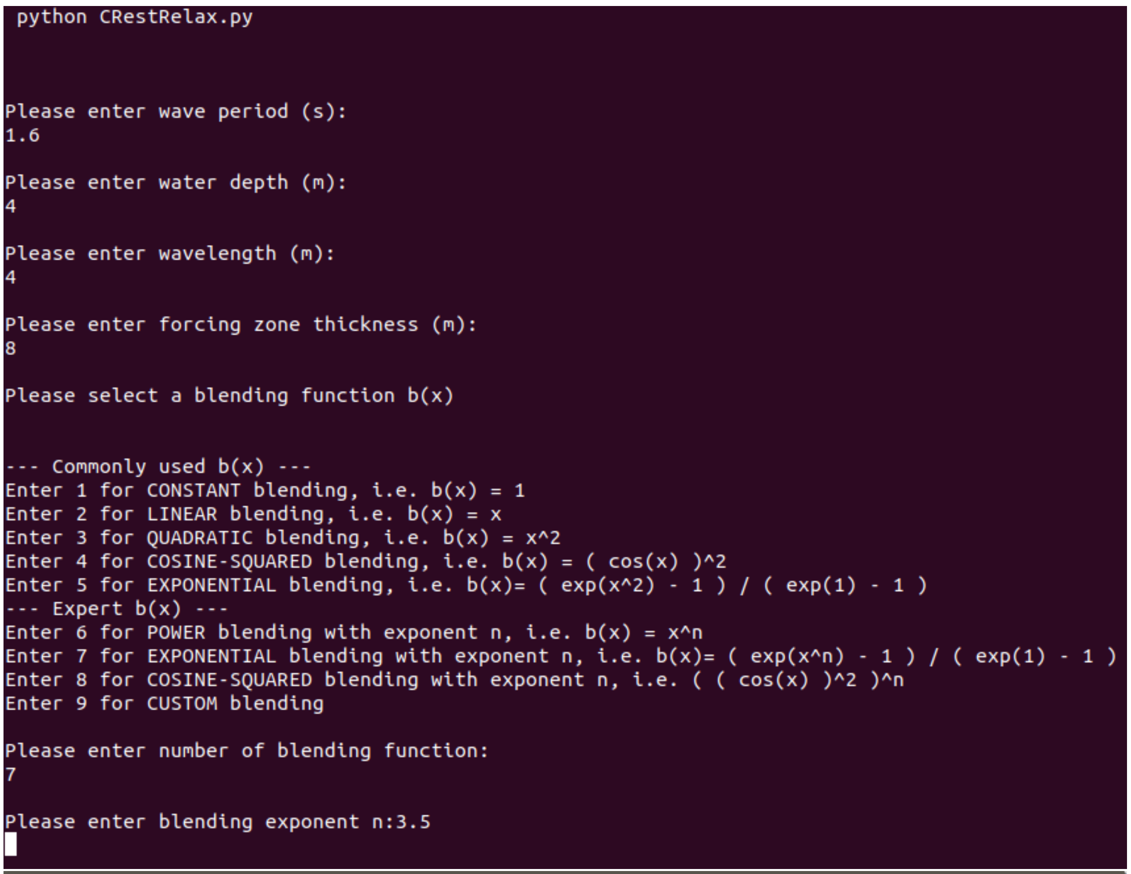
[5]To use the code in batch-mode for blending via Eqs. (7-11), type :

```
python CRestRelax.py T h L xd b
```

and replace `T` with the wave period in seconds, `h` with the water depth in meters, `L` with the wavelength in meters, `xd` by the zone thickness in meters, and `b` by the number representing the desired blending function (cf. Fig. 2).
To use the code in batch-mode for blending via Eqs. (12-14), type :

```
python CRestRelax.py T h L xd b n
```

and replace n by the blending function exponent.



```
 python CRestRelax.py


Please enter wave period (s):
1.6

Please enter water depth (m):
4

Please enter wavelength (m):
4

Please enter forcing zone thickness (m):
8

Please select a blending function b(x)


--- Commonly used b(x) ---
Enter 1 for CONSTANT blending, i.e. b(x) = 1
Enter 2 for LINEAR blending, i.e. b(x) = x
Enter 3 for QUADRATIC blending, i.e. b(x) = x^2
Enter 4 for COSINE-SQUARED blending, i.e. b(x) = ( cos(x) )^2
Enter 5 for EXPONENTIAL blending, i.e. b(x)= ( exp(x^2) - 1 ) / ( exp(1) - 1 )
--- Expert b(x) ---
Enter 6 for POWER blending with exponent n, i.e. b(x) = x^n
Enter 7 for EXPONENTIAL blending with exponent n, i.e. b(x)= ( exp(x^n) - 1 ) / ( exp(1) - 1 )
Enter 8 for COSINE-SQUARED blending with exponent n, i.e. ( ( cos(x) )^2 )^n
Enter 9 for CUSTOM blending

Please enter number of blending function:
7

Please enter blending exponent n:3.5
```

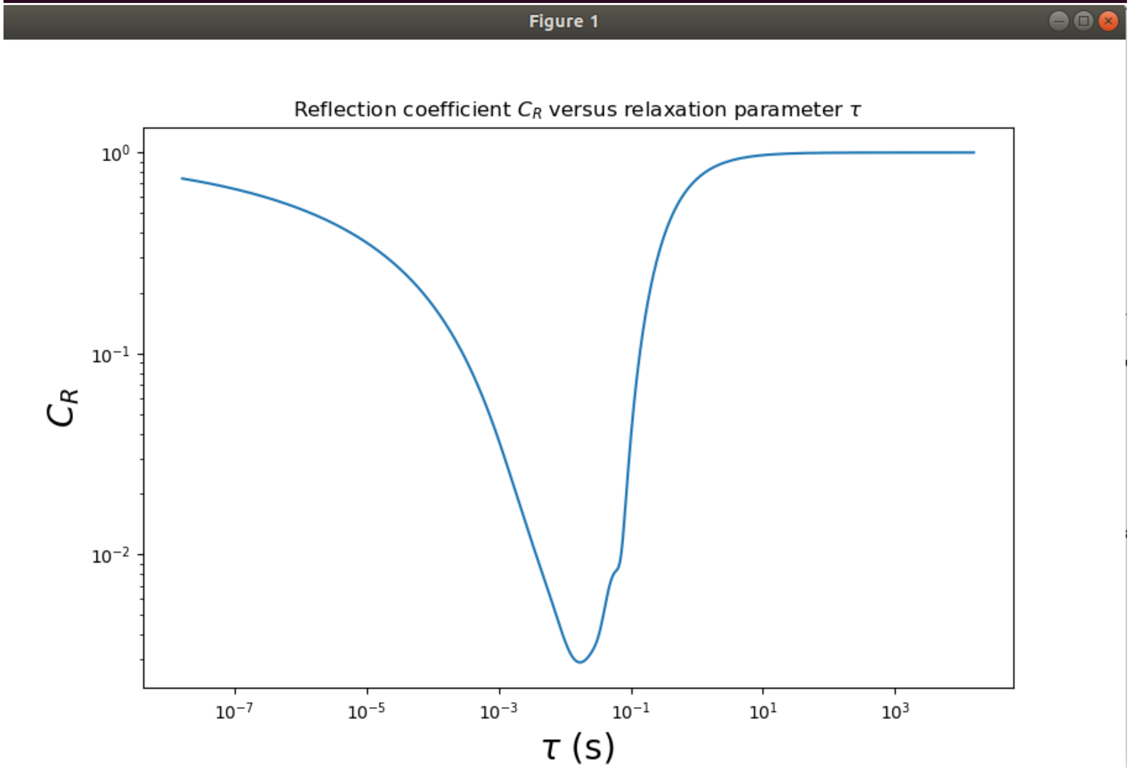


Figure 2: Example output of the code to compute the reflection coefficient C_R depending on the value of relaxation parameter τ; for a wave with period $T = 1.6\,\mathrm{s}$, water depth $h = 4\,\mathrm{m}$, wavelength $\lambda = 4\,\mathrm{m}$; with a relaxation zone of thickness $x_\mathrm{d} = 8\,\mathrm{m}$ and exponential blending via Eq. (13) with exponent $n = 3.5$

9 Reporting bugs

If you find bugs, or if you have questions or suggestions, please contact the author:

Dr.-Ing. Robinson Perić
Hamburg University of Technology (TUHH)
Institute for Fluid Dynamics and Ship Theory (M8)
Am Schwarzenberg-Campus 4
D-21073 Hamburg, Germany
Room C5.004
Phone: +49 40 42878 6031
Fax: +49 40 42878 6055
E-mail: robinson.peric@tuhh.de
URL: http://www.tuhh.de/fds/staff/

The code is available at
`https://github.com/wave-absorbing-layers/relaxation-zones-for-free-surface-waves`
Updates and further useful information will be posted there as well.

10 Bug fixes & previous versions of the code

The 2017 version of this code produced slightly inaccurate predictions for shallow water conditions. This has been amended, and now the code should work for all water depths and all relaxation-zone settings. Furthermore, the interactive-mode was adjusted to be more user-friendly, and a batch mode was added, so parameters can also be passed as arguments.

March 11, 2021: The preprint [19] describing the analytical approach behind the code was updated and this manual was improved, among other things to provide a better distinction between explicit and implicit relaxation zones.

11 Copyright

12 References

[1] Choi, J., Yoon, S. B., 2009. Numerical simulations using momentum source wave-maker applied to RANS equation model. Coastal Eng., 56 (10), 1043-1060. `https://doi.org/10.1016/j.coastaleng.2009.06.009`.

[2] Engsig-Karup, A. P., Hesthaven, J. S., Bingham, H. B., Madsen, P. A., 2006. Nodal DG-FEM solution of high-order Boussinesq-type equations. J. Eng. Math., 56 (3), 351-370. `https://doi.org/10.1007/s10665-006-9064-z`.

[3] Fuhrman, D. R., Madsen, P. A., Bingham, H. B., 2006. Numerical simulation of lowest-order short-crested wave instabilities. J. Fluid. Mech., 563, 415. `https://doi.org/10.1017/S0022112006001236`.

[4] Israeli, M., Orszag, S. A., 1981. Approximation of radiation boundary conditions. J. Comput. Phys., 41 (1), 115-135. https://doi.org/10.1016/0021-9991(81)90082-6.

[5] Jacobsen, N. G., Fuhrman, D. R., Fredsøe, J., 2012. A wave generation toolbox for the open-source CFD library: OpenFoamő. Int. J. Numer. Meth. Fl., 70 (9), 1073-1088. https://doi.org/10.1002/fld.2726.

[6] Jasak, H., Vukčević , V., Gatin, I., 2015. Numerical Simulation of Wave Loads on Static Offshore Structures. In: CFD for Wind and Tidal Offshore Turbines, Springer Tracts in Mechanical Engineering, Cham, pp. 95-105. ISBN 978-3-319-16201-0. http://dx.doi.org/10.1007/978-3-319-16202-7.

[7] Kim, J., O'Sullivan, J., Read, A., 2012. Ringing analysis of a vertical cylinder by Euler overlay method. In Proc. OMAE2012, Rio de Janeiro, Brazil. https://doi.org/10.1115/OMAE2012-84091.

[8] Larsen, J., Dancy, H., 1983. Open boundaries in short wave simulations – a new approach. Coast. Eng., 7 (3), 285-297. https://doi.org/10.1016/0378-3839(83)90022-4.

[9] Li, Z., Deng, G., Queutey, P., Bouscasse, B., Ducrozet, G., Gentaz, L., Touzé, D. L., Ferrant, P., 2019. Comparison of wave modeling methods in CFD solvers for ocean engineering applications. Ocean Eng., 188, 106237. https://doi.org/10.1016/j.oceaneng.2019.106237.

[10] Madsen, P. A., Bingham, H. B., Schäffer, H. A., 2003. Boussinesq-type formulations for fully nonlinear and extremely dispersive water waves: derivation and analysis. In: Proc. Roy. Soc. Lond. A Mat., 459(2033), 1075-1104. https://doi.org/10.1098/rspa.2002.1067.

[11] Mayer, S., Garapon, A., Sørensen L., 1998. A fractional step method for unsteady free-surface flow with applications to non-linear wave dynamics. Int. J. for Num. Meth. Fl., 28 (2), 293-315. https://doi.org/10.1002/(SICI)1097-0363(19980815)28:2<293::AID-FLD719>3.0.CO;2-1.

[12] Park, J. C., Zhu, M., Miyata, H., 1993. On the accuracy of numerical wave making techniques. J. Soc. Naval Arch. Japan, 1993, (173), 35-44. https://doi.org/10.2534/jjasnaoe1968.1993.35.

[13] Park, J. C., Kim, M. H., Miyata, H., 1999. Fully non-linear free-surface simulations by a 3D viscous numerical wave tank. Int. J. Numer. Meth. Fl., 29 (6), 685-703. https://doi.org/10.1002/(SICI)1097-0363(19990330)29:6<685::AID-FLD807>3.0.CO;2-D.

[14] Park, J. C., Kim, M. H., Miyata, H., 2001. Three-dimensional numerical wave tank simulations on fully nonlinear wave-current-body interactions. J. Mar. Sci. Tech., 6 (2), 70-82. https://doi.org/10.1007/s773-001-8377-2.

[15] Perić, R., 2019. Minimizing undesired wave reflection at the domain boundaries in flow simulations with forcing zones. PhD-thesis at Hamburg University of Technology, Schriftenreihe Schiffbau, 713, Hamburg, Germany. https://doi.org/10.15480/882.2394.

[16] Perić, R., Abdel-Maksoud, M., 2016. Reliable damping of free-surface waves in numerical simulations. Ship Tech. Res., 63 (1), 1-13. https://doi.org/10.1080/09377255.2015.1119921.

[17] Perić, R., Abdel-Maksoud, M., 2018. Analytical prediction of reflection coefficients for wave absorbing layers in flow simulations of regular free-surface waves. Ocean Eng., 147, 132-147.

https://doi.org/10.1016/j.oceaneng.2017.10.009.

[18] Perić, R., Abdel-Maksoud, M., 2020. Reducing Undesired Wave Reflection at Domain Boundaries in 3D Finite Volume–Based Flow Simulations via Forcing Zones. J. Ship Res., 64, 1. https://doi.org/10.5957/jsr.2020.64.1.23.

[19] Perić, R., Vukčević, V., Abdel-Maksoud, M., & Jasak, H. (2020). Optimizing wave-generation and wave-damping in 3D-flow simulations with implicit relaxation-zones. arXiv preprint arXiv:1806.10995. https://arxiv.org/abs/1806.10995v3.
Previous version of this preprint: Tuning the Case-Dependent Parameters of Relaxation Zones for Flow Simulations With Strongly Reflecting Bodies in Free-Surface Waves. arXiv preprint arXiv:1806.10995. https://arxiv.org/abs/1806.10995.

[20] Schmitt, P., Elsaesser, B., 2015. A review of wave makers for 3D numerical simulations. In: Marine 2015, 6th Int. Conf. on Computat. Meth. in Marine Eng., Rome, Italy. http://hdl.handle.net/2117/332346.

[21] Schmitt, P., Windt, C., Davidson, J., Ringwood, J. V., Whittaker, T., 2019. The efficient application of an impulse source wavemaker to CFD simulations. J. Mar. Sci. Eng., 7 (3), 71. https://doi.org/10.3390/jmse7030071.

[22] Vukčević, V., Jasak, H., Gatin, I., 2017. Implementation of the Ghost Fluid Method for free surface flows in polyhedral Finite Volume framework. Comp. & Fl., 153, 1-19. https://doi.org/10.1016/j.compfluid.2017.05.003.

[23] Vukčević, V., Jasak, H., Malenica, Š., 2016a. Decomposition model for naval hydrodynamic applications, Part I: Computational method. Ocean Eng., 121, 37-46. https://doi.org/10.1016/j.oceaneng.2016.05.022.

[24] Vukčević, V., Jasak, H., Malenica, Š., 2016b. Decomposition model for naval hydrodynamic applications, Part II: Verification and validation. Ocean Eng., 121, 76-88. https://doi.org/10.1016/j.oceaneng.2016.05.021.

[25] Wei, G., Kirby, J. T., Sinha, A., 1999. Generation of waves in Boussinesq models using a source function method. Coast. Eng., 36 (4), 271-299. https://doi.org/10.1016/S0378-3839(99)00009-5.

[26] Windt, C., Davidson, J., Ringwood, J. V., 2018. High-fidelity numerical modelling of ocean wave energy systems: A review of computational fluid dynamics-based numerical wave tanks. Renewable and Sustainable Energy Reviews, 93, 610-630. https://doi.org/10.1016/j.rser.2018.05.020.

[27] Windt, C., Davidson, J., Schmitt, P., Ringwood, J. V., 2019. On the assessment of numerical wave makers in CFD simulations. J. Mar. Sci. Eng., 7 (2), 47. https://doi.org/10.3390/jmse7020047.