A New Volume-Of-Fluid Method in Openfoam

Pedersen, Johan Rønby; Eltard-Larsen, Bjarke; Bredmose, Henrik; Jasak, Hrvoje

Published in:
MARINE 2017 Computational Methods in Marine Engineering VII

Publication date:
2017

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Pedersen, J. R., Eltard-Larsen, B., Bredmose, H., & Jasak, H. (2017). A New Volume-Of-Fluid Method in Openfoam. In M. Visonneau, P. Queutey, & D. Le Touzé (Eds.), MARINE 2017 Computational Methods in Marine Engineering VII (pp. 266-278). Nantes: International Center for Numerical Methods in Engineering.

VII International Conference on Computational Methods in Marine Engineering
MARINE 2017
M. Visonneau, P. Queutey and D. Le Touzé (Eds)

A NEW VOLUME-OF-FLUID METHOD IN OPENFOAM

Johan Roenby[*], Bjarke Eltard Larsen[⋆], Henrik Bredmose[†] AND Hrvoje Jasak[‡]

[*]DHI, Agern All 5, 2970 Hørsholm, Denmark, e-mail: jro@dhigroup.com
[⋆]DTU Mechanical Engineering, Richard Petersens Plads, 2800 Kgs. Lyngby, Denmark, e-mail: bjelt@mek.dtu.dk
[†] DTU Wind Energy, Technical University of Denmark, Nils Koppels Alle, 2800 Lyngby, Denmark
[‡]Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Ivana Lucica 5, Zagreb, Croatia

Key words: CFD, Marine Engineering, Interfacial Flows, IsoAdvector, VOF Methods, Surface Gravity Waves

Abstract. To realise the full potential of Computational Fluid Dynamics (CFD) within marine science and engineering, there is a need for continuous maturing as well as verification and validation of the numerical methods used for free surface and interfacial flows. One of the distinguishing features here is the existence of a water surface undergoing large deformations and topological changes during transient simulations e.g. of a breaking wave hitting an offshore structure. To date, the most successful method for advecting the water surface in marine applications is the Volume-of-Fluid (VOF) method. While VOF methods have become quite advanced and accurate on structured meshes, there is still room for improvement when it comes to unstructured meshes of the type needed to simulate flows in and around complex geometric structures. We have recently developed a new geometric VOF algorithm called isoAdvector for general meshes and implemented it in the OpenFOAM interfacial flow solver called interFoam. We have previously shown the advantages of isoAdvector for simple pure advection test cases on various mesh types. Here we test the effect of replacing the existing interface advection method in interFoam, based on MULES limited interface compression, with the new isoAdvector method. Our test case is a steady 2D stream function wave propagating in a periodic domain. Based on a series of simulations with different numerical settings, we conclude that the introduction of isoAdvector has a significant effect on wave propagation with interFoam. There are several criteria of success: Preservation of water volume, of interface sharpness and shape, of crest kinematics and celerity, not to mention computational efficiency. We demonstrate how isoAdvector can improve on many of these parameters, but also that the success depends on the solver setup. Thus, we cautiously conclude that isoAdvector is a viable alternative to MULES when set up correctly, especially when interface sharpness, interface smoothness and calculation times are important. There is, however, still potential for improvement in the coupling of isoAdvector with interFoam's PISO based pressure-velocity solution algorithm.

1 INTRODUCTION

Computational Fluid Dynamics (CFD) is quickly gaining popularity as a tool for testing and optimising marine structural designs and interaction with the surrounding water environment. A concrete example is the assessment of extreme wave loads on offshore wind turbine foundations of various types and shapes. From a numerical perspective, one of the great challenges within marine CFD is accurate description and advection of a complex free surface, or air-water interface. Various solution strategies have been developed to cope with this challenge[1]. The most widely used within practical interfacial CFD is the Volume-of-Fluid (VOF) method. In VOF, the interface is indirectly represented by a numerical field describing the volume fraction of water within each computational cell. The game of VOF is then about "guessing" how much water is floating across the faces between adjacent cells within a time step. VOF methods come in two variants: 1) geometric VOF methods, using geometric operations to reconstruct the fluid interface inside a cell and to approximate the water fluxes across faces, and 2) algebraic VOF methods, relying on the limiter concept to blend first and higher order schemes in order to retain sharpness and boundedness of the time advanced VOF field. Geometric VOF schemes are typically much more accurate, but also computationally more expensive, complex to implement, and restricted to certain types of computational meshes, such as hexahedral meshes. Algebraic VOF schemes, on the other hand, are less accurate, but often faster, easier to implement, and developed for general mesh types[2].

In marine applications, we often encounter complex geometries that are impossible, or at least very difficult, to represent properly with a structured mesh. Hence, most free surface CFD within marine engineering is based on algebraic VOF methods. Therefore, such simulations often require excessive mesh resolution and therefore long calculation times to obtain the desired solution quality.

To address the need for an improved computational interface advection method, we have developed a new VOF approach called IsoAdvector[3]. It is geometric of nature both in the interface reconstruction and advection step, but is applicable on general meshes consisting of arbitrary polyhedral cells. In the interface reconstruction, we take a novel approach using isosurface calculations to find the interface position and orientation in the intersected cells. In the advection step, we rely on calculation of the *face-interface intersection line* sweeping a mesh face during a time step. This avoids expensive calculations of intersections between cells and flux polyhedra[4]. For a thorough description of the isoAdvector concept the reader is referred to [3].

We have previously demonstrated using pure advection test cases that our new approach leads to accurate interface advection on both structured and unstructured meshes without compromising calculation times[3]. In OpenFOAM's interfacial flow solver, interFoam, each time step starts by a MULES based update of the interface, followed by an update of the pressure and velocity, using a variant of the PISO algorithm[2]. In this segregated solution approach we can simply remove the MULES code snippet and replace it by a corresponding isoAdvector based snippet. To complete the replacement of MULES with isoAdvector, we must also calculate the mass flux across the faces – the quantity called `rhoPhi` in the interFoam code – based on isoAdvector, since this is needed in the convection term for the velocity field in construction and solution of the discretised momentum equations. In [5], we show how to derive a simple expression for `rhoPhi` from the mass fluxes provided by isoAdvector. The resulting solver is called interFlow and is provided as open source together with the isoAdvector library and various test cases at github.com/isoadvector.

In the following, we investigate the ability of interFlow and interFoam to propagate a stream function wave for 10 wave periods across a computational domain, which is exactly one wave length long and has periodic boundary conditions on the sides.

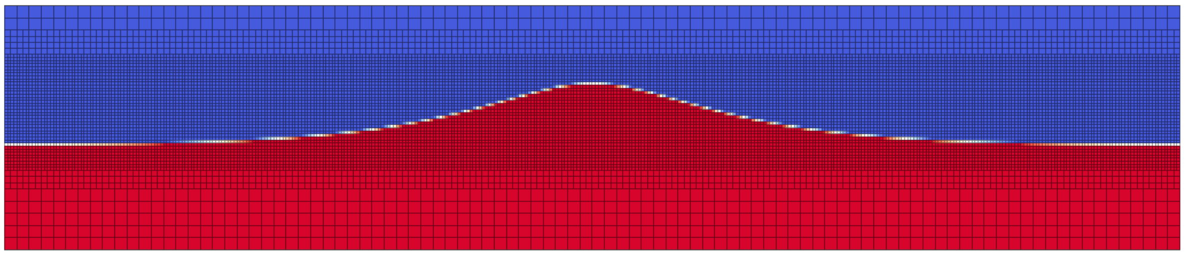

Figure 1: The initial wave shape. The defining wave parameters are the water depth: D = 20 m, wave height: H = 10 m, wave period: T = 14 s and mass transport velocity: $\bar{u}_2$ = 0 m/s. Some derived quantities are the wave length: L = 193.23 m, steepness: H/L = 0.052, celerity: c = 13.80 m/s, crest height: h_{crest} = 7.25 m, crest particle speed: u_{crest} = 5.95 m/s, trough height: h_{trough} = -2.75 m, trough particle speed: u_{trough} = -2.25 m/s.

2 PHYSICAL SETUP

A stream function wave is a periodic steady wave calculated from potential flow theory using a truncated Fourier expansion of the surface and stream function describing the wave. The Fourier coefficients are calculated using a numerical root finding method in parameter space and by growing the wave height in steps so the solution procedure can be seeded with an Airy wave. For details on the solution procedure the reader is referred to [6]. Here we adopt the wave used in [7] and shown in Fig. 1, which also gives the wave parameters in the caption. The advantage of using stream function wave theory as opposed to Stokes N'th order theory for the input wave is that the former does not rely on the smallness of the wave amplitude, which is the Taylor expansion parameter of Stokes wave theory.

One thing to keep in mind, when attempting to reproduce potential theory waves in CFD is that these waves are calculated under the assumption of a free surface, i.e. zero pressure and no air phase on top of the water surface. In our simulation we have a second phase above the water and we are free to set the densities of the two phases. The water density will be set to 1000 kg/m^3. Ideally, we would like to set the air density to zero for our stream function test case, but numerical stability issues limit how low we can set the air density. We choose an air density of 1 kg/m^3, which is close to the real physical value. This is a good compromise, on one hand high enough to limit high density ratio related instabilities at the interface, and on the other hand low enough to make the air behaving like a “slave fluid” moving passively out of the way in response to motion of the much heavier water surface.

The viscosities in both phases is set to zero in accordance with potential flow theory and we have deactivated the turbulence model. This amounts to running the solver in “Euler equation mode", albeit the numerics will to some extend introduce an effective dissipation leading to a lack of strict energy conservation.

For waves on the space and time scales considered here surface tension is irrelevant and we set it to zero in our simulations.

3 NUMERICAL SETUP

The interFoam and interFlow solvers used in this study are based on the OpenFOAM-v1612+ version provided by ESI-OpenCFD. The details of the PISO algorithm implementation are described in [2] and can be studied by inspecting the OpenFOAM code library, which is freely available at openfoam.com.

For all simulations in the following the sides have periodic boundary conditions for both the VOF-field, α, the velocity field, U, and the pressure, p. On the top and bottom we have zero normal gradient for α and p, and a slip boundary condition for U.

The mesh type with square cells and two refinement zones covering the interface region is show in Fig. 1. This is the finest mesh used in this study with 20 cells per wave height and 384 cells per wavelength in the interface region. Two coarser meshes with square cells were also used: One with the finest refinement removed, yielding 10 cells per wave height, and a very coarse mesh with no refinement at all and only 5 cells per wave height.

In all simulations we use adaptive time stepping based on a maximum allowed CFL number. We show results with CFL = 0.1, 0.2 and 0.4. It should be noted that in water wave simulations with interFoam the velocities in the air phase above the water surface are often higher than the maximum velocities in the water volume. The air behaviour depends a lot on the choices of numerical schemes and settings, but for our density ratio of 1:1000, it is not uncommon to see air velocities that are 2-3 times higher than the velocity of the water particles in the wave crests. Thus, in a simulation with a maximum allowed CLF number of 0.1 the actual maximum CFL number in the water volume may in fact stay below 0.05. It might be fruitful to introduce in the time stepping algorithm a separate CFL limit for each of the two phases.

Besides mesh and time resolution, the accuracy of wave propagation simulations depends on the choices of schemes for the different terms in the momentum equations. In particular the results are sensitive to the choice of time integration scheme. Therefore, in what follows, we show results for both pure Euler time integration and a 50% mixture of Euler and Crank-Nicolson. Another influential scheme is the momentum convection scheme. The convective term is linearized and treated implicitly, so we use the face mass fluxes from a previous time step or iteration to advect the updated velocity field through the face. For the cell-to-face interpolation involved in the discretisation of the convective term we use a TVD method specialised for vector fields, called `limitedLinearV` in OpenFOAM terminology. This scheme requires specification of a coefficient in the range $\psi \in [0, 1]$, where 0 gives best accuracy and 1 gives best convergence[8]. In the following we use $\psi = 1$.

All discretisation schemes and solver settings used in the presented simulations are listed in Appendix A and B to allow the reader to verify our results.

4 RESULTS

In the subsequent two sections we first vary the spatial resolution and then the CFL number to investigate its effect on the wave propagation properties of interFlow (isoAdvector) and interFoam (MULES).

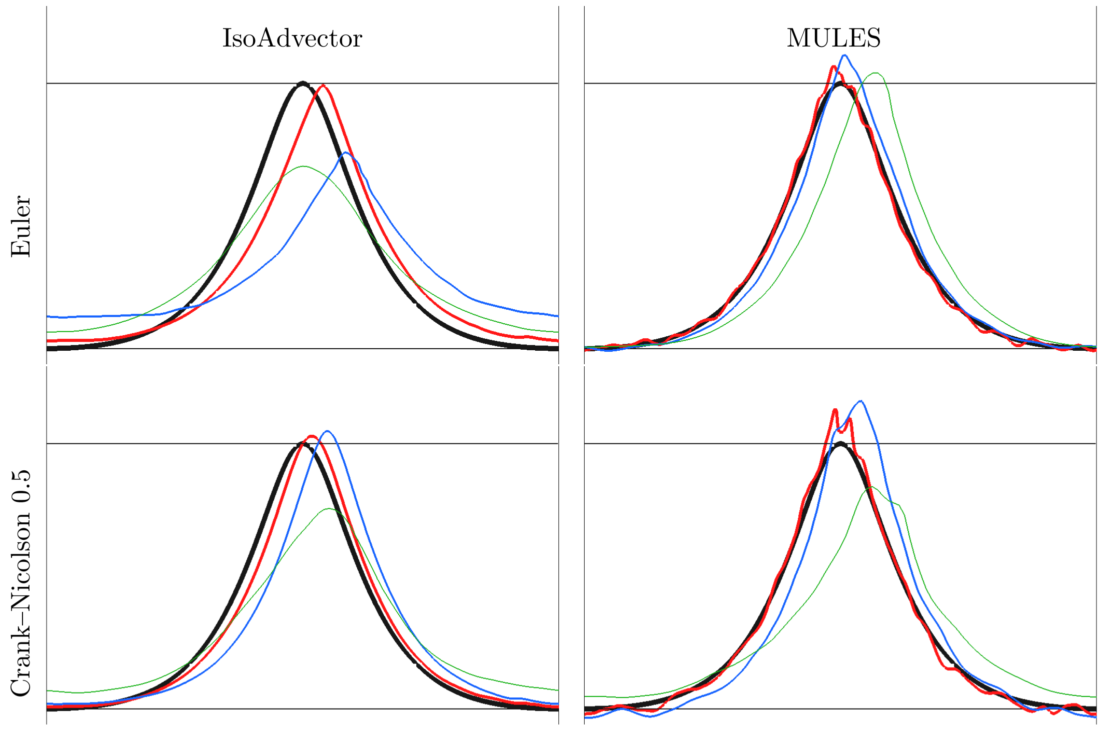

Figure 2: Surface elevation after 10 wave periods with CFL = 0.1. For convenience of plotting the horizontal axis has been compressed by a factor of 10. Black: Exact, Green: H/dx = 5, Blue: H/dx = 10, Red: H/dx = 20. Top panels: Euler time discretisation. Bottom panels: 50% blended Crank-Nicolson and Euler time integration. Left panels: interFlow/IsoAdvector. Right panels: interFoam/MULES.

4.1 Mesh refinement study

To investigate the effect of spatial resolution we simulate for L/dx = 5, 10 and 20 the propagation of the wave through the periodic domain for 10 wave periods (140 s) and plot the final surface curve compared to the exact theoretical solution. The results are shown in Fig. 2. We observe that:

- In terms of surface shape preservation the best performance is obtained with isoAdvector on the finest mesh where MULES gives a wiggly surface.
- In spite of the wiggly surface MULES is superior in terms celerity on the finest mesh with

almost no visible phase shift.

- Using isoAdvector on the coarsest mesh leads to excessive decay in wave height.
- On the intermediate mesh isoAdvector also has excessive wave height decay with Euler but not with Crank-Nicolson.
- MULES with Euler looks surprisingly good on the coarsest mesh. Inspection of the time series reveals that this is a "lucky" snapshot right after the wave has broken due to excessive steepening. In general it can not be recommended to use meshes with only 5 cells per wave height with the numerical setup used here.

In Table 1 we show the time it took for the simulation of the 10 periods to finish on a single core for the different combinations of schemes and resolutions. IsoAdvector is significantly faster than MULES for all combinations except for the H/dx = 10 with Euler. For the best settings, H/dx=20 and Crank-Nicolson, isoAdvector is 32% faster than MULES and slightly faster than the MULES-Euler combination.

H/dx	isoAdvector	MULES
5	314	335
10	1892	1228
20	4356	5741

(a) Euler

H/dx	isoAdvector	MULES
5	304	435
10	918	1669
20	5624	8151

(b) Crank-Nicolson 0.5

Table 1: Simulation times in seconds on a single core for 10 periods.

4.2 Time refinement study

As shown in [3], isoAdvector is accurate in pure advection test cases for CFL number up to 0.5. It is our experience that isoAdvector works well for such cases even for CFL numbers closer to (albeit not exceeding) 1. In [3] we also demonstrate how MULES requires CFL < 0.1 to converge. We would therefore hope that replacing MULES with isoAdvector in interFoam could allow more accurate solutions with larger time steps. In Fig. 3 we show the results of an exercise where we keep the mesh resolution fixed at H/dx = 20 and vary the CFL time step limit from 0.1 to 0.2 and on to 0.4. We observe that:

- IsoAdvector with Euler gives excessive wave damping for CFL = 0.2 and 0.4.
- IsoAdvector with Crank-Nicolson 0.5 gives slightly worse but acceptable results with CFL = 0.2 with an increase in phase error and overprediction of wave height.
- IsoAdevctor with Crank-Nicolson and CFL = 0.4 causes severe wave breaking.
- MULES with Euler and CFL = 0.4 crashes before the simulation has finished.
- In spite of its wiggly surface MULES with CFL = 0.2 is very close to the CFL = 0.1 result only differing by a small phase error.

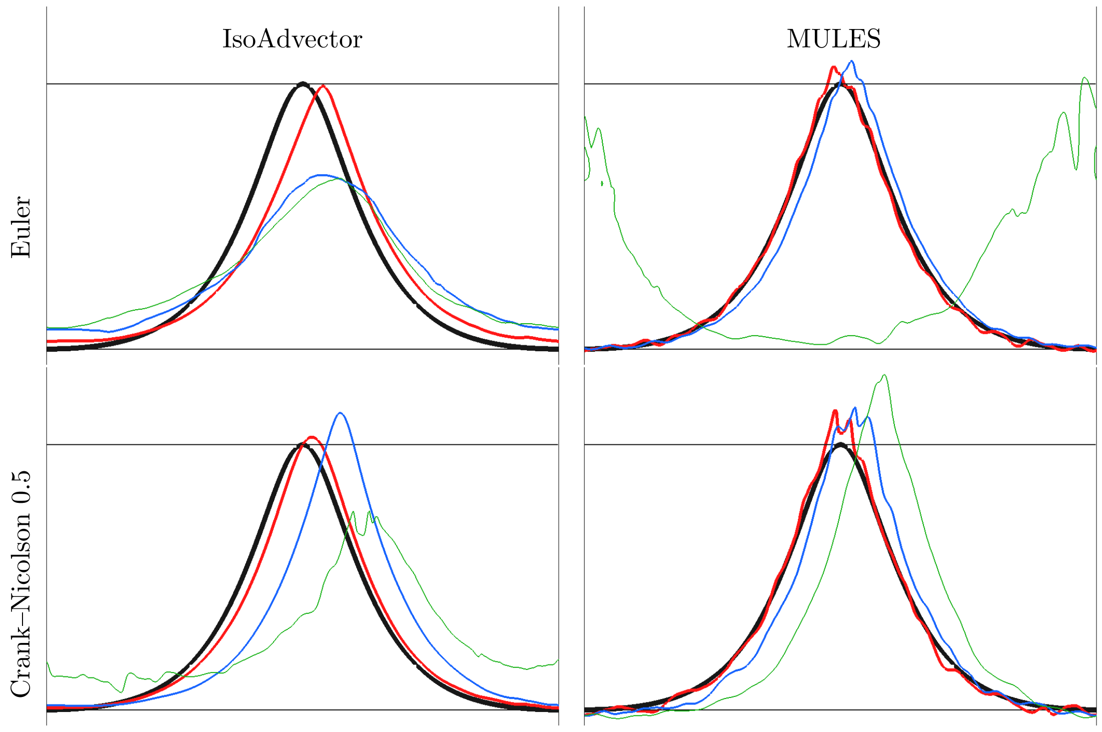

Figure 3: Surface elevation after 10 wave periods. For convenience of plotting the horizontal axis has been compressed by a factor of 10. Black: Exact, Green: CFL = 0.4, Blue: CFL = 0.2, Red: CFL = 0.1. Top panels: Euler time discretisation. Bottom panels: 50% blended Crank-Nicolson and Euler time integration. Left panels: interFlow/IsoAdvector. Right panels: interFoam/MULES.

This is somewhat disappointing for our hopes that isoAdvector would allow accurate simulations with large time steps. It should be noted, that the current coupling of isoAdvector with the pressure-velocity coupling is the simplest possible. Probably one should look for an improvement in this coupling rather than for an improvement in the inner workings of the isoAdvector method itself.

4.3 Crest velocity profiles

An important feature to be able to capture accurately in wave propagation simulations is the particle kinematics in the wave crest. As for instance shown in [9], many solvers have issues with overshooting in the particle velocities in the top of the crest. To investigate this, we show in Fig. 4 the variation in the x-component of the velocity along a line of cells going up through the wave crest. The results are shown for the simulations with H/dx = 20 and CFL = 0.1 at time t = 70 s, i.e. after 5 wave periods. It is evident from this figure that with the current implementation of isoAdvector into interFoam we get higher overshoots in the crest velocities than the original interFoam solver with MULES which does a remarkably good job with the

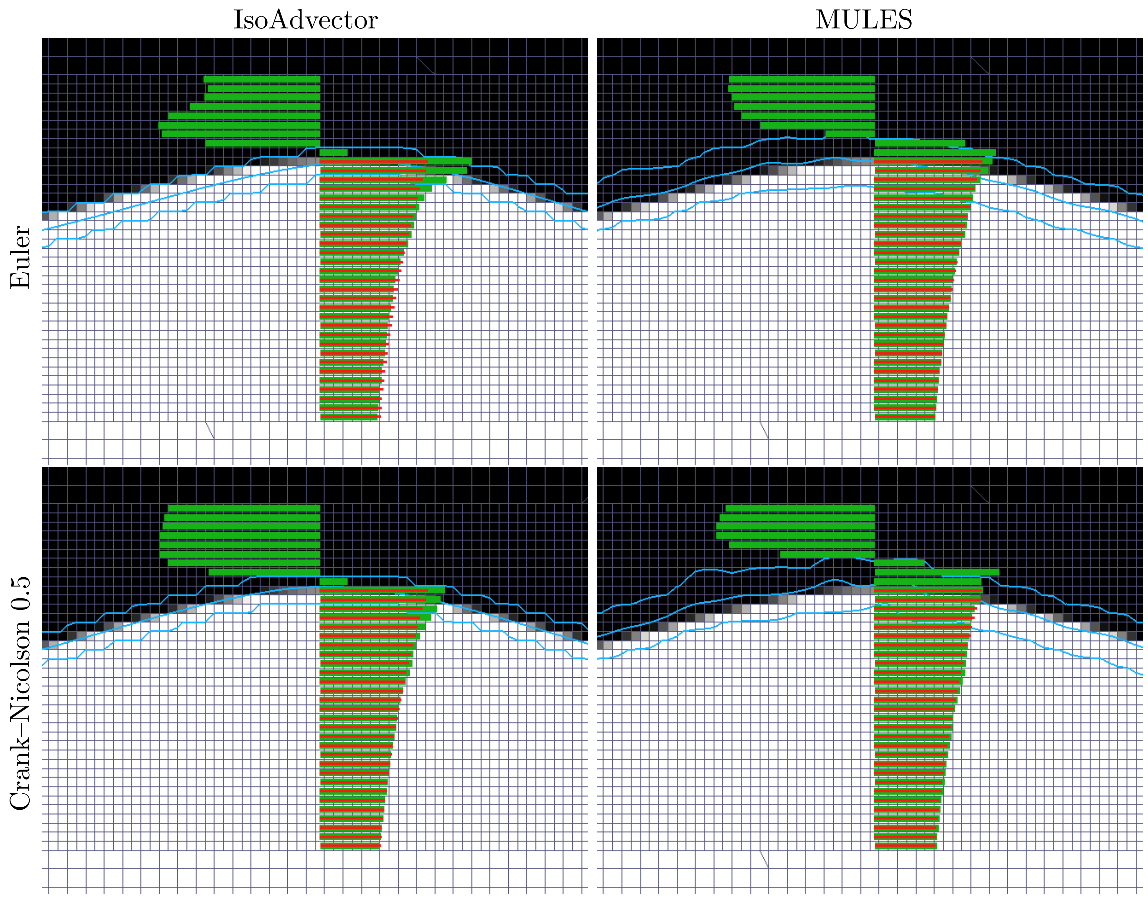

Figure 4: Horizontal velocity in cell centres at wave crest after 5 wave periods of the simulation with $H/dx = 20$ and CFL $= 0.1$. Red: Exact, Green: Simulation result. The α-field is shown in a black-white colour map and the $\alpha = 0.001$, 0.5 and 0.999 contours are plotted in blue. Top panels: Euler time discretisation. Bottom panels: 50% blended Crank-Nicolson and Euler time integration. Left panels: interFlow/IsoAdvector. Right panels: interFoam/MULES.

Crank-Nicolson 0.5 time integration. Since the surface is advected passively in the velocity field, one should think that there was a strong correlation between a solver's ability to represent these velocities accurately near the surface and its ability to accurately propagate the surface and preserve its shape. This does not seem to be the case here where isoAdvector, in spite of its errors in crest kinematics, produces a better surface, and MULES, in spite of its accurate crest kinematics, produces a wrinkled surface.

In Fig. 4, we show the interface width by plotting the $\alpha = 0.001$, 0.5 and 0.999 contours in blue colour. Careful inspection reveals that the distance between the 0.001 and 0.999 contours with isoAdvector is 3 which is essentially the theoretical minimal interface width for a VOF method. The corresponding distance with MULES is approximately twice as large, i.e. approximately 6 cells. This moves the stagnation point, where the air velocity above the crest changes direction, one cell closer to the surface. In a true two-phase potential flow solution the tangential jump in velocity should be right on the interface. In this sense the isoAdvector solution is closer to the theoretical one.

4.4 Cell aspect ratio

It has previously been shown that the cell aspect ratio can have a significant effect on the propagation of waves in OpenFOAM and on the breaking point of shoaling waves[10]. Clearly, independence of simulation results on cell aspect ratios and cell shapes in general are highly desirable features. To investigate how isoAdvector behaves with different cell aspect ratios we have repeated the simulations on a mesh with flat cells (H/dx = 10 and H/dy = 20) and on a mesh with tall cells (H/dx = 20 and H/dy = 10) in the interface region. The results are shown in Fig. 5 where they are compared to the finest resolution results shown previously. We see that the halving of the cell count in the interface region has only a small effect on the isoAdvector simulation results. For MULES the surface wrinkles are exacerbated when using tall cells. For flat cells the wrinkles completely disappear and a slight phase error is introduced.

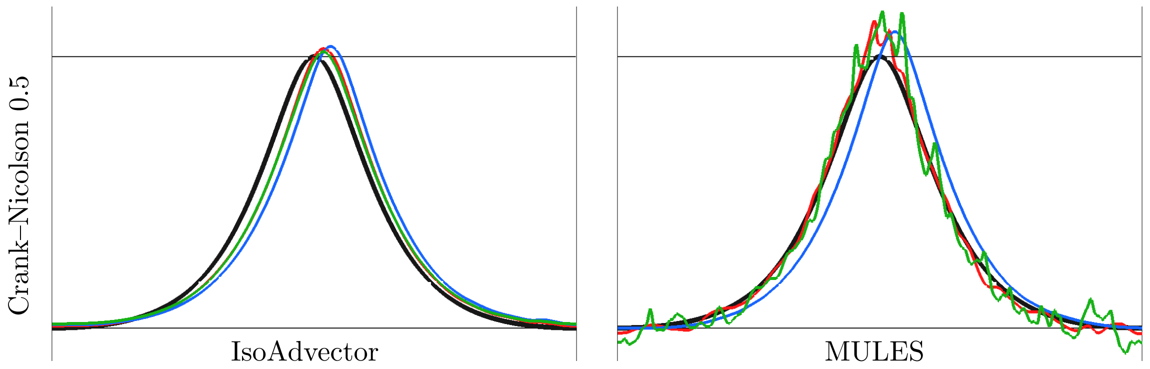

Figure 5: Surface elevation after 10 wave periods. For convenience of plotting the horizontal axis has been compressed by a factor of 10. Black: Exact. Red: square cells, H/dx = H/dy = 20. Blue: Flat cells, H/dx = 10, H/dy = 20. Green: Tall cells, H/dx = 20, H/dy = 10. Top panels: Euler time discretisation. Bottom panels: 50% blended Crank-Nicolson and Euler time integration. Left panels: interFlow/IsoAdvector. Right panels: interFoam/MULES.

5 CONCLUSION

We have demonstrated the feasibility of using the new geometric VOF algorithm, isoAdvector, in the OpenFOAM interfacial flow solver, interFoam, to propagate a steady stream function wave through a periodic domain. The benefits of using interFlow (interFoam with isoAdvector) as opposed to MULES is a sharper and more smooth surface, shorter calculation times and less sensitivity to cell aspect ratio. It is not recommended to use the solver with Euler integration and fewer than 10 cells per wave height. Satisfactory results are obtained with a 50:50 blend of Euler and Crank-Nicolson and 20 cells per wave height.

In spite of problems with a wrinkly surface the original interFoam solver with MULES performs better than interFlow when it comes to phase error (celerity) on the finest mesh and reproduction of the theoretical crest kinematics profile. Also, at this stage interFlow does not produce satisfactory results when running with CFL number > 0.2 as one might otherwise hope due to its ability to advect interfaces accurately at CFL numbers close to 1. We expect that higher accuracy at larger CFL numbers can be obtained by improving the way isoAdvector is coupled with the PISO loop in the interFoam solver.

Finally a word of caution regarding this kind of numerical comparisons. Choosing schemes and solver settings is a delicate procedure which requires some degree of informed guessing. It may well be that one combination of schemes produces accurate results for a particular test case because the energy that, say, the chosen time integration scheme erroneously injects into the system is by pure luck equal to the energy erroneously taken out of the system due to the coarseness of the mesh. Results may then look reasonable even though the numerical calculation does not in reality represent the simulated physics properly. A professional CFD engineer should always stress test her setup with an attitude of trying to prove it wrong, rather than trying to prove it right.

Acknowledgements

This work was funded by JR's Sapere Aude: DFF-Research Talent grant from The Danish Council for Independent Research | Technology and Production Sciences (Grant DFF-1337-00118) and by DHI's GTS grant from the Danish Agency for Science, Technology and Innovation.

A Solver settings

```
PIMPLE
{
    momentumPredictor yes;
    nCorrectors     3;
    nOuterCorrectors 1;
    nNonOrthogonalCorrectors 0;
    nAlphaCorr      1;
    nAlphaSubCycles 1;
    cAlpha          1;
    pRefPoint       (1 0 16);
    pRefValue       0;
}
```

```
isoAdvector
{
    interfaceMethod isoAdvector;
    isoFaceTol  1e-8;
    surfCellTol 1e-8;
    snapAlpha 1e-12;
    nAlphaBounds 3;
    clip   true;
}
```

```
"alpha.water.*"
{
    nAlphaCorr      2;
    nAlphaSubCycles 1;
    cAlpha          1;
```

```
p_rgh
{
    solver          GAMG;
    tolerance       1e-8;
    relTol          0.01;
    smoother        DIC;
    MULESCorr       no;
    nLimiterIter    3;

    solver          smoothSolver;
    smoother        symGaussSeidel;
    tolerance       1e-8;
    relTol          0;
}

pcorr
{
    solver          PCG;
    preconditioner
    {
        preconditioner  GAMG;
        tolerance       1e-5;
        relTol          0;
        smoother        DICGaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration false;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }
    tolerance       1e-06;
    relTol          0;
    maxIter         100;
}


U
{
    solver          smoothSolver;
    smoother        GaussSeidel;
    tolerance       1e-7;
    relTol          0.05;
    nSweeps         2;
}
```

```
    nPreSweeps      0;
    nPostSweeps     2;
    nFinestSweeps   2;
    cacheAgglomeration true;
    nCellsInCoarsestLevel 10;
    agglomerator    faceAreaPair;
    mergeLevels     1;
}

p_rghFinal
{
    solver          PCG;
    preconditioner
    {
        preconditioner  GAMG;
        tolerance       1e-8;
        relTol          0;
        nVcycles        2;
        smoother        DICGaussSeidel;
        nPreSweeps      2;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    tolerance       1e-9;
    relTol          0;
    maxIter         20;
}


UFinal
{
    solver          smoothSolver;
    smoother        GaussSeidel;
    tolerance       1e-8;
    relTol          0;
   nSweeps         2;
}
```

B Discretisation schemes

```
ddtSchemes{default CrankNicolson 0.5;} //Euler
```

```
gradSchemes{default Gauss linear;}
divSchemes
{
    div(rhoPhi,U)  Gauss limitedLinearV 1;
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss interfaceCompression;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes{default Gauss linear corrected;}
interpolationSchemes{default linear;}
snGradSchemes{default corrected;}
```

REFERENCES

[1] G. Tryggvason, R. Scardovelli, and S. Zaleski, *Direct numerical simulations of gas–liquid multiphase flows.* Cambridge University Press, 2011.

[2] S. S. Deshpande, L. Anumolu, and M. F. Trujillo, “Evaluating the performance of the two-phase flow solver interFoam,” *Computational Science & Discovery*, vol. 5, no. 1, p. 014016, 2012.

[3] J. Roenby, H. Bredmose, and H. Jasak, “A computational method for sharp interface advection,” *Royal Society Open Science*, vol. 3, no. 11, p. 160405, 2016.

[4] J. Hernández, J. López, P. Gómez, C. Zanzi, and F. Faura, “A new volume of fluid method in three dimensions-part I: Multidimensional advection method with face-matched flux polyhedra,” *International Journal for Numerical Methods in Fluids*, vol. 58, no. 8, pp. 897–921, 2008.

[5] J. Roenby, H. Bredmose, and H. Jasak, “Isoadvector: Vof on general meshes,” in *11th OpenFOAM Workshop* (J. M. Nóbrega and H. Jasak, eds.), Springer Nature, 2017. submitted.

[6] John D. Fenton, “Numerical methods for nonliner waves,” in *Advances in Coastal and Ocean Engineering*, vol. 5, pp. 241–324, World Scientific, July 1999.

[7] B. T. Paulsen, H. Bredmose, H. Bingham, and N. Jacobsen, “Forcing of a bottom-mounted circular cylinder by steep regular water waves at finite depth,” *Journal of Fluid Mechanics*, vol. 755, pp. 1–34, Sept. 2014.

[8] C. J. Greenshields, “Openfoam user guide,” *OpenFOAM Foundation Ltd, version*, vol. 3, no. 1, 2015.

[9] P. A. Wroniszewski, J. C. G. Verschaeve, and G. K. Pedersen, “Benchmarking of Navier-Stokes codes for free surface simulations by means of a solitary wave,” *Coastal Engineering*, vol. 91, pp. 1–17, Sept. 2014.

[10] N. G. Jacobsen, D. R. Fuhrman, and J. Fredsøe, “A wave generation toolbox for the open-source CFD library: OpenFoam,” *International Journal for Numerical Methods in Fluids*, vol. 70, no. 9, pp. 1073–1088, 2012.