On the Need for Higher Order Interpolation with Overset Grid Methods

Sébastien Lemaire[*†], **Guilherme Vaz**[*‡], **and Stephen Turnock**[†]
[*]MARIN, Wageningen/Netherlands, [‡]WavEC, Lisbon/Portugal,
[†]FSI Group, University of Southampton, Southampton/UK;
sebastien.lemaire@soton.ac.uk

1 Introduction

Overset methods, by overlapping several grids, allow large and arbitrary motion of bodies. First developed for the aerospace industry (Benek et al., 1985), it is now used for a wider range of applications like maritime ones (Carrica et al., 2013). At the fringes of each grid, the coupling is done by interpolating field variables. This interpolation step is key in any overset computations since the choice of the interpolation scheme directly influences the accuracy of the solution and the performance. Most overset implementations done for unstructured CFD codes uses 1st or at most 2nd order interpolation schemes (Gatin et al., 2018, Völkner et al., 2017, Chandar et al., 2018) but higher order interpolation schemes (3rd or 4th) are not often studied (Chandar, 2019). This paper evaluates the use of higher order (higher than 2nd order) interpolation schemes for the overset method by comparing the accuracy of different interpolation schemes on a Poiseuille flow test case, no performance study is however conducted here. The overset implementation used in this study is a coupling between the CFD solver ReFRESCO and the overset library Suggar++ (Noack and Boger, 2009).

The paper is organised as follows: section 2 presents the overset approach and the different interpolation schemes used in this study. Section 3 discusses the robustness of Least Squares interpolation scheme. Section 4 describes the test cases ran for the study as well as the computational setup. Results are presented in section 5 and finally concluding remarks are given in section 6.

2 Overset method and Interpolation

ReFRESCO (`refresco.org`) is a community-based open-usage CFD code for the maritime world. It solves multiphase incompressible flows using the Navier-Stokes equations, complemented with turbulence models, cavitation models and volume-fraction transport equations for different phases. The equations are discretised using a finite-volume approach with cell-centered collocated variables. Like most unstructured solver, ReFRESCO's discretisation is at most 2nd order.

The overset implementation used in this study has been described in Lemaire et al., 2018. It uses the two external libraries Suggar++ and DiRTlib (Noack and Boger, 2009) to compute the domain connectivity information. The domain connectivity information or DCI defines a status for each cell; *Hole* cells are being ignored by the solver, they correspond to cells outside the domain or that are being overlapped by any other mesh. *In* cells are normal domain cells. Finally, between *Hole* and *In* cells as well as at the edge of an overlapping mesh *Fringe* cells are placed. These cells are the ones performing the coupling between the different meshes. Each *Fringe* cell will receive interpolated field data from an other mesh. The interpolation phase is done in two steps: first each *Fringe* cell is assigned several donor cells that belong to the other mesh (a donor cell being a normal *In* cell); then donor cells field data are gathered and the interpolation is performed, the interpolated value is placed at the *Fringe* cell center. In an explicit coupling between meshes, the interpolated value is placed on the right hand side corresponding to each *Fringe* cell (for more details see Lemaire et al., 2018).

Multiple interpolation schemes can be used to perform this coupling, the number of donor cells needed depending on the scheme employed. In this study, three different schemes are tested resulting in an interpolation order between 1 and 4. First, the *inverse distance* scheme, it is a weighted average where the weights are based on the distance between the interpolation location and the donor cell center location. The number of donor cells N can be freely chosen. Equation 1 shows a formal description where ϕ is the field to interpolate, $\tilde{\phi}$ the interpolated function, $\boldsymbol{x_i}$ the i-th donor location, $\boldsymbol{x}$ the interpolation location (cell center of the *Fringe*) and ω_i the weight corresponding to the i-th donor cell.

$$\tilde{\phi}(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} \omega_i \phi(\boldsymbol{x_i})}{\sum_{i=1}^{N} \omega_i}, \qquad \omega_i = \frac{1}{\|\boldsymbol{x} - \boldsymbol{x_i}\|^p}. \tag{1}$$

The *inverse distance* interpolation is a first order scheme that guaranties for the interpolated value to be bounded. It is very often implemented in overset capable solvers like StarCCM+ (Schreck and Peric, 2012), FOAM-extend (Gatin et al., 2018) or FRESCO+ (Völkner et al., 2017).

By using the gradient of the donor cell, a second order scheme can be constructed using only one donor cell. This scheme will be called *nearest cell gradient* in the following:

$$\tilde{\phi}(\boldsymbol{x}) = \phi(\boldsymbol{x_1}) + \nabla\phi(\boldsymbol{x_1}) \cdot (\boldsymbol{x} - \boldsymbol{x_1}). \tag{2}$$

Finally, unlimitedly high order interpolations can be performed using the *least squares* approach in conjunction with polynomial based interpolations. To obtain a n-th order interpolation, an $n-1$-th degree polynomial function is used. The polynomial function is constructed by minimising the square of the errors made on the donor locations (*least squares* approach). In this approach, the number of donor cells used has to be higher or equal to the number of unknown coefficients in the polynomial function. Table 1 shows the minimum number of donor cells depending on the polynomial degree and the dimension of the problem. One should note that a n-th degree polynomial function will lead to an $n+1$ interpolation order. Section 3 discusses the robustness of *least squares* interpolation depending on the number of donor cell used.

Dimension	N (number of donor cells)			
	n (degree)	$n=1$	$n=2$	$n=3$
2D	$\frac{(n+1)(n+2)}{2}$	3	6	10
3D	$\sum_{i=0}^{n} \frac{(i+1)(i+2)}{2}$	4	10	20

Table 1: Minimum number of donor cells N for *least squares* interpolation depending on the dimension of the problem and the degree of the polynomial function used. The resulting interpolation order is $n+1$, with n the degree of the polynomial.

3 Study of Least Squares interpolation robustness

Amongst the interpolation schemes presented in the previous section, the *least squares* approach is the only one that can provide an interpolation order of 3 and higher. Its robustness however depends on the number of donor cells it uses, and this section studies the influence of the number of donor cells used in the interpolation error. Note that increasing the number of donor cells reduces parallel performance due to an increased number of communications.

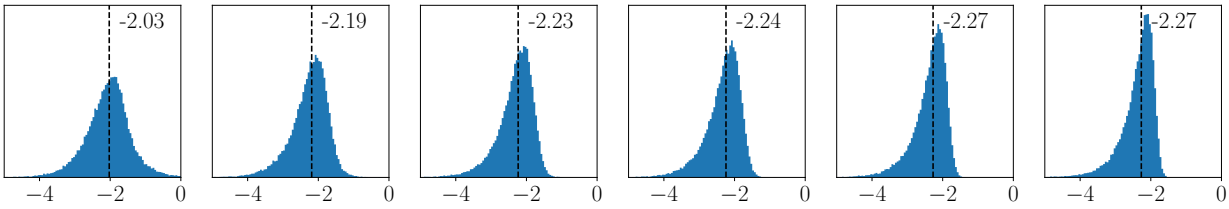
When using the *least squares* scheme, for a given interpolation order, a minimum number of donor cells is needed. When this minimum is used, the interpolation becomes a pure polynomial interpolation where the interpolated function goes through every donor cell point (the error made on each donor cell is 0). In this section the *least squares* interpolation using a degree two polynomial function (leading to a 3rd order interpolation scheme) is tested in isolation without overset computations. An analytical function (Eq 3) is used as a field data to compute the interpolation on. This function was chosen because it is bounded between -1 and 1, it does not show any oscillation in the $[-1,1]\times[-1,1]$ domain and is constructed with non polynomial functions. Donor points are randomly placed on the $[-1,1]\times[-1,1]$ domain and the interpolation error is measured by performing a *least squares* interpolation close to the donor points average location, which ensures that no extrapolation is performed. For each set of donor point, the interpolation error is measured by taking the difference between the interpolated value and the result of f at the interpolation location.

$$f(x,y) = \sin(-2x) + \cos(\frac{3y}{2}). \tag{3}$$

Figure 1 shows error distributions for various number of donor cells used. On the horizontal axis, the log of the error is displayed. For each plot, 50 000 different sets of donor points were generated and interpolations performed. The median error is marked by the dashed line and C_{mult} is the multiplier coefficient that gives the number of donor N compared to the minimum number of donors (6). Using more donor points tends to reduce the median error here, however this is not a general result since it has been observed that for some

functions f (different from Eq 3) this was not true. When C_{mult} increases, the spreading of the errors higher than the median reduces, which leads to a more robust interpolation since higher errors are less likely to be returned. Using only one more donor than the minimum ($N = 7$) the effect is already visible and no set of donor cells result in errors higher than 10^{-1}.

This test was conducted by placing donor points randomly in space, however, when placing the donor points on a Cartesian grid some interpolations result in infinitely high errors. The reason for this is the alignment of donor points that polynomial based interpolations are sensible to. When using a Cartesian mesh, increasing the number of donor points reduces the number of diverging results. If the minimum number of donor point ($N = 6$) is used, 2.9% of the interpolations result in infinitely high errors, this decreases to 0.3% when using only 1 more donor point ($N = 7$), and goes to 0% at $N = 9$. Fully Cartesian meshes are not often found in complex computations, however locally Cartesian or locally structured meshes with enough cell alignments are more common, and in such situations, using the minimum number of donor cells is not advisable. A C_{mult} of 1.5 was enough to remove any diverging results and gives robust *least squares* interpolations, and therefore this value is used in the following section for overset computations.



(a) $C_{mult} = 1$ ($N = 6$) (b) $C_{mult} = 1.2$ ($N = 7$) (c) $C_{mult} = 1.3$ ($N = 8$) (d) $C_{mult} = 1.5$ ($N = 9$) (e) $C_{mult} = 2.0$ ($N = 12$) (f) $C_{mult} = 2.7$ ($N = 16$)

Fig. 1: Influence of the number of donor points (N) on the error distribution. Histograms show the log of the errors for a degree two *least squares* interpolations. Plot 1a has a C_{mult} of 1, and it is mathematically equivalent to a Polynomial Interpolation of degree 2.

4 Test case description

For this study, the *inverse distance* scheme, *nearest cell gradient* scheme and *least squares* interpolation using a polynomial function of degree 1, 2 and 3 are used, leading to 1st, 2nd (both *nearest cell gradient* and *least squares 1*), 3rd and 4th order interpolation schemes. The *inverse distance* scheme is using here 4 donor cells, however very similar results were found with either 1 or 10 donor cells. All of the computations were done on a *Poiseuille* flow case using 5 grid assemblies consisting of a background grid and a foreground one, both being 2D Cartesian meshes with a 2:1 ratio (like in Lemaire et al., 2018). The foreground grid is twice smaller in both direction compared to the background grid and is positioned in the center of the background grid and rotated by about 19 degrees. Table 2 shows the different cell counts and Figure 2 shows the grid assembly and cell status as computed by Suggar++. Two layers of *Fringe* cells are used in order to reconstruct gradients properly. Finally, computations done using only the background grid, hence without the overset method, were performed for comparison purposes.

	Background	N_i	Foreground	N_i^f
G1	32x16	512	16x8	128
G2	64x32	2048	32x16	512
G3	128x64	8192	64x32	2048
G4	256x128	32768	128x64	8192
G5	512x256	131072	256x128	32768

Table 2: Details of the Cartesian grids used and cell count (N_i)

The *Poiseuille* flow case is an academic case for which an analytical solution is known but still relevant for maritime applications being a boundary layer problem. The 2D flow is bounded between a top and bottom wall and driven by an axial pressure gradient. The analytical solution is presented in Eq. 4 with L the length of the channel (X direction), and h the distance between the walls.

$$U_x(y) = \frac{Ph^2}{2\mu}\left(1 - \frac{y^2}{h^2}\right) \quad , \quad U_y = 0 \quad , \quad \frac{\partial p}{\partial x} = -P \quad , \quad p(x) = -P(x - L) \quad . \tag{4}$$

In this test case steady computations are performed, and the momentum equation and the pressure correction via the SIMPLE method are solved. The Reynolds number, based on the height of the channel, is $Re_h = 10$. At the inlet, the analytical velocity profile is set and at the outlet the pressure is enforced. The top and bottom of the domain uses non-slip wall boundary conditions. Convergence is ensured by letting the infinity norm of the residual stagnate, that happens below 10^{-14} for all the computations performed. A second order central difference scheme is used for the discretisation of convective and diffusive fluxes.

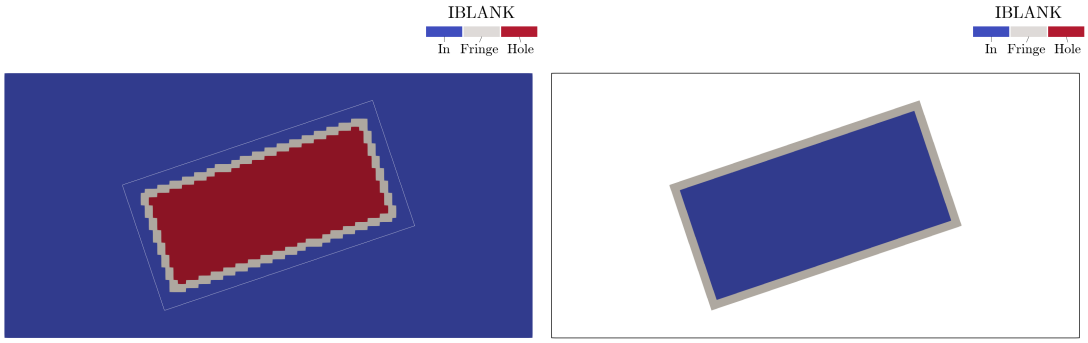


Fig. 2: Cell status computed by Suggar++ for the G3 grid assembly. Two layers of *Fringe* cells are used.

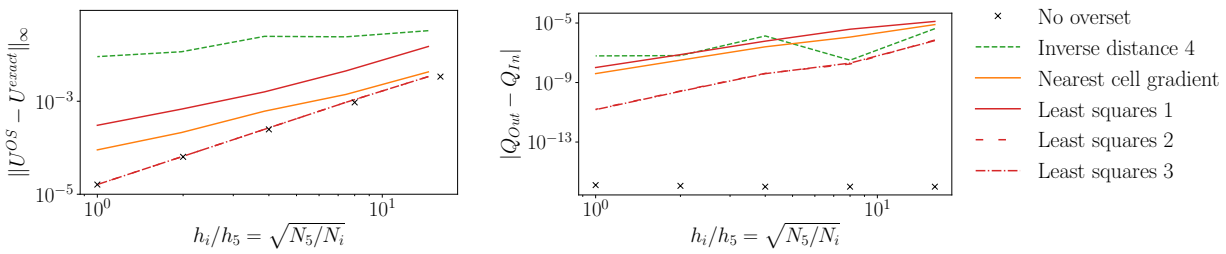
5 Results

Since an analytical solution is known for a Poiseuille flow test case, errors can be evaluated in the entire domain. Figure 3a shows the infinity norm of the error made on the velocity field depending on the grid refinement. Each scheme shows a converging trend, the error levels as well as the convergence order (slope of the curve when refining the grid) however varies between schemes. Computations done with the *inverse distance* scheme resulted in a convergence order of about 0.5, and also yield errors level several orders of magnitude higher than the other schemes. The *nearest cell gradient* scheme results in a convergence order of about 1.5 like the degree 1 *least squares* scheme, with error levels however being lower when using the *nearest cell gradient* scheme. Finally, the two last *least squares* schemes tested (using 2nd and 3rd degree polynomial functions, i.e. 3rd and 4th order schemes) result in the same error levels and a convergence order of 2.0 (the two curves overlapping each others), exactly like the computations done without overset.

The mass imbalance between the inlet and outlet of the domain is also measured in Figure 3b for all scheme and grids. This metric is particularly relevant since it relates to the pressure accuracy in the domain and for unsteady cases mass imbalance can lead to pressure fluctuations. Contrary to the computation done without overset, every scheme introduce some mass imbalance. Once again, the 3rd and 4th order schemes show similar results with the finest grid giving a mass imbalance as low as 1.3×10^{-11}, meaning that there is no gain in using a scheme of interpolation order higher than 3 and that this level of mass imbalance is likely to be negligible. The *inverse distance* scheme is performing closely to the *nearest cell gradient* one here but with a less smooth trend. The *least squares 1* scheme shows a similar slope as the *nearest cell gradient* but with a higher imbalance.

When analysing the error localisation, similar observations can be drawn. Figure 4 shows the log of the error made on the velocity field for the grid assembly G3. Both the foreground and background grid are displayed on a single plot here, only the bottom half of the foreground grid being shown to be able to see the *Fringe* cells of the background grid. Infinitely low errors are present at the inlet of the domain for every computation because the analytical solution is prescribed as a boundary condition. Visible artefacts due to the interpolation in the overlap region are visible for the *inverse distance*, *nearest cell gradient* and *least squares 1* schemes but not for the two higher degree *least squares* computations. Compared to *nearest cell gradient*, the *least squares 1* shows a smoother field close to the *Fringe* cells, partially because it uses more donor cells to perform the interpolation. However, it has a higher maximum but also norm-2 error field.

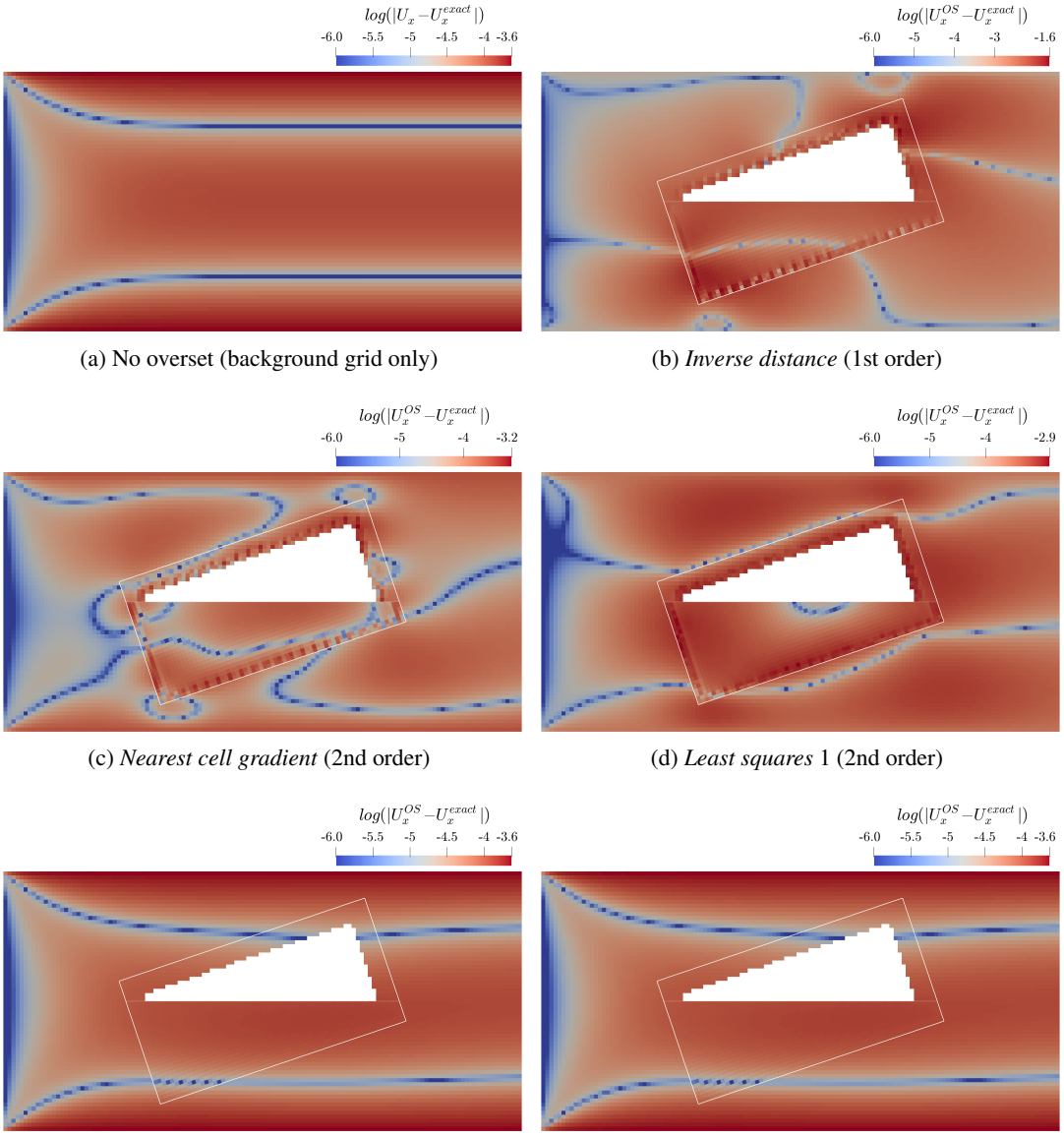
Finally, the error field for the 3rd and 4th order interpolation schemes are very similar to the computation done without overset. Computing the actual difference between the non overset computation and the 3rd or 4th order *least squares* interpolation on the background grid shows that the maximum difference is bellow $10^{-4.4}$ for the grid G3, which is 1 order of magnitude lower than the difference with the analytical solution.



(a) Infinity norm of the error for the velocity field

(b) Mass Imbalance between inlet and outlet

Fig. 3: Infinity norm of the error and mass imbalance against the grid refinement for every scheme tested. The *least squares* 2 and 3 lines are overlapping each other on both plots.



(a) No overset (background grid only)

(b) *Inverse distance* (1st order)

(c) *Nearest cell gradient* (2nd order)

(d) *Least squares* 1 (2nd order)

(e) *Least squares* 2 (3rd order)

(f) *Least squares* 3 (4th order)

Fig. 4: Log of the error for the velocity field computed with the analytical solution for the grid assembly G3. Each plot has its own color bar with the maximum corresponding to the maximum error of the field and a minimum at 10^{-6}. No overset related artefacts are visible for the schemes of order 3 or higher.

6 Concluding remarks

From this study performed on interpolation for overset meshes several conclusions can be drawn. On a CFD solver that uses second order schemes for its discretisation, at least a 3rd order interpolation scheme is needed for the overset coupling in order for the interpolation error to be bellow the transport equation discretisation errors. Moreover, no gain is achieved when using interpolation schemes of order higher than 3 (same mass imbalance and error). Some mass imbalance is introduced with the use of interpolation, it however reduces with the increase of interpolation order and grid refinement. Minimising the mass imbalance is particularly relevant on unsteady cases since it can lead to pressure fluctuations being propagated in the entire domain. In order to perform a 3rd order interpolation, one can use the *least squares* approach. When using locally structured or Cartesian grids however, more donor cells are needed to prevent high errors. The computational cost is however higher when increasing the number of donor cells since the search and the gathering of field data need more parallel communications. On the other hand, the *nearest cell gradient* scheme only needs one donor cell, is a second order scheme and has error levels lower than the *inverse distance* scheme, it should hence be preferred over the later. Future works are now focusing on the performance and parallelisation of the interpolation and donor cell searching methods in order to make 3rd order *least squares* interpolation more affordable.

Acknowledgements

The authors would like to thank Dr. Ralph Noack and Prof. Pablo Carrica for their help with the coupling with Suggar++. This work was supported by the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling (EP/L015382/1). The authors acknowledge the use of the IRIDIS High Performance Computing Facility at the University of Southampton, in the completion of this work.

References

Benek, J., Buning, P. G., and Steger, J. (1985). A 3-D chimera grid embedding technique. In *7th Computational Physics Conference*, page 10, Reston, Virigina. American Institute of Aeronautics and Astronautics.

Carrica, P. M., Ismail, F., Hyman, M., Bhushan, S., and Stern, F. (2013). Turn and zigzag maneuvers of a surface combatant using a URANS approach with dynamic overset grids. *Journal of Marine Science and Technology*, 18(2):166–181.

Chandar, D. D. (2019). On overset interpolation strategies and conservation on unstructured grids in OpenFOAM. *Computer Physics Communications*, pages 1–12.

Chandar, D. D., Boppana, V. B. L., and Kumar, V. (2018). A Comparative Study of Different Overset Grid Solvers Between OpenFOAM, StarCCM+ and Ansys-Fluent. In *AIAA SciTech Forum*, number January, Kissimmee, Florida.

Gatin, I., Vukcevic, V., Jasak, H., and Lalovic, I. (2018). Manoeuvring simulations using the overset grid technology in foam-extend. In *32nd Symposium on Naval Hydrodynamics*, number August, page 10, Hamburg, Germany.

Lemaire, S., Vaz, G., and Turnock, S. R. (2018). Implementation and Verification of an Explicit Overset Grid Method. In *21th Numerical Towing Tank Symposium (NuTTS)*, Cortona, Italy.

Noack, R. W. and Boger, D. A. (2009). Improvements to SUGGAR and DiRTlib for Overset Store Separation Simulations. In *47th AIAA Aerospace Sciences Meeting*, number January, pages 5–9, Orlando, Florida.

Schreck, E. and Peric, M. (2012). Overset Grids in STAR-CCM+: Methodology, Applications and Future Developments. In *STAR Japanese Conference*.

Völkner, S., Brunswig, J., and Rung, T. (2017). Analysis of non-conservative interpolation techniques in overset grid finite-volume methods. *Computers and Fluids*, 148:39–55.