Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXV, págs. 855-871 (artículo completo)
Martín I. Idiart, Ana E. Scarabino y Mario A. Storti (Eds.)
La Plata, 7-10 Noviembre 2017

HIGH-ORDER INTERPOLATION BETWEEN ADJACENT CARTESIAN FINITE DIFFERENCE GRIDS OF DIFFERENT SIZE

Alejandro Figueroa and Rainald Löhner

Center for Computational Fluid Dynamics, George Mason University, M.S. 6A2, Fairfax, VA 22030-4444, USA, afiguer7,rlohner@gmu.edu, https://cfd.gmu.edu/~rlohner

Keywords: Finite Difference Solvers, Interpolation, 2:1 Transition

Abstract. Nested cartesian grid systems by design require interpolation of solution fields from coarser to finer grid systems. While several choices are available, preserving accuracy, stability and efficiency at the same time require careful design of the interpolation schemes. Given this context, a series of interpolation algorithms for nested cartesian finite difference grids of different size were developed and tested. These algorithms are based on post-processing, on each local grid, the raw (bi/trilinear) information passed to the halo points from coarser grids. In this way modularity is maximized while preserving locality.

The results obtained indicate that the schemes improve markedly the convergence rates and the overall accuracy of finite difference codes with varying grid sizes.

1 INTRODUCTION

High-order cartesian finite difference (FD) solvers have shown considerable advantages in speed and simplicity for the solution of partial differential equations. Coupling with block-wise adaptive mesh refinement combined with prismatic meshing near the body or usage of immersed or embedded body techniques appear to remove the main obstacles to attaining geometrical flexibility and solution accuracy at the same time (Nakahashi, 2003; Sitaraman et al., 2008; Löhner et al., 2014; Sitaraman et al., 2017).
However, the transition between grids of different spatial resolution is still a source of difficulty. A recent study conducted by the authors for a 6th-order FD flow solver revealed that at the transition between grids the numerical diffusion and dispersion due to interpolation completely overwhelmed the physics. This prompted a search for better interpolation techniques and interface treatments for such problems.
If one considers the interpolation problem, the immediate inclination is to use standard classic high-order interpolation techniques. High-order interpolation in local refined Cartesian grids have been reported in literature (Ray et al., 2005; McCorquodale and Collella, 2011; Hittinger and Banks, 2013), where techniques such as Fourier basis, cubic least squares and limited monotone (WENO type) schemes were considered for designing the interpolations. In three dimensions, this often led to large stencil sizes of the order of 20 to 30 coefficients that need to be synthesized for each fine grid point. Moreover, interpolation schemes were fixed to maintain a particular order of accuracy and extending them to even higher orders often required revisiting the derivations and re-implementation of new schemes. Having a large stencil footprint often defeats the purpose of using 'fast and cheap' FD methods. This has led to the consideration of simpler interpolation techniques, utilizing immediately available data and constructing fast schemes that reuse results of interpolations performed previously. The aim of the present paper it to assess whether improvements in accuracy with a modest increase in complexity are possible using such schemes.

2 *H/2H* INTERPOLATION BETWEEN CARTESIAN GRIDS

The situation commonly encountered is shown in Fig. 1,2 for the 1-D and 2-D cases respectively. We denote the fine and coarse grids as *grid h* and *grid 2h* respectively. At the *h/2h* boundary, the Cartesian grids need to exchange information. The assumption is made that in order to maintain code modularity, halo points are used to transfer information between grids (and also for boundary conditions). In this way, the 'update' and 'boundary condition' stages are separated in a clean, modular fashion. At the beginning of each timestep, iteration, or Runge-Kutta stage the information required for the halo points is obtained from the appropriate neighbouring grids. Furthermore, it is assumed that the information given at gridpoints is the most accurate and should therefore not be changed. This implies that for points that coincide (labeled D in Fig. 2) a direct injection / transfer of information is desirable. On the other hand, for the points along edges or faces (labeled E,F in Fig. 2), one is at liberty to apply interpolation schemes of different order. Note that these will only be required for *grid h* on a *h/2h* boundary, i.e. only for the finer grid.

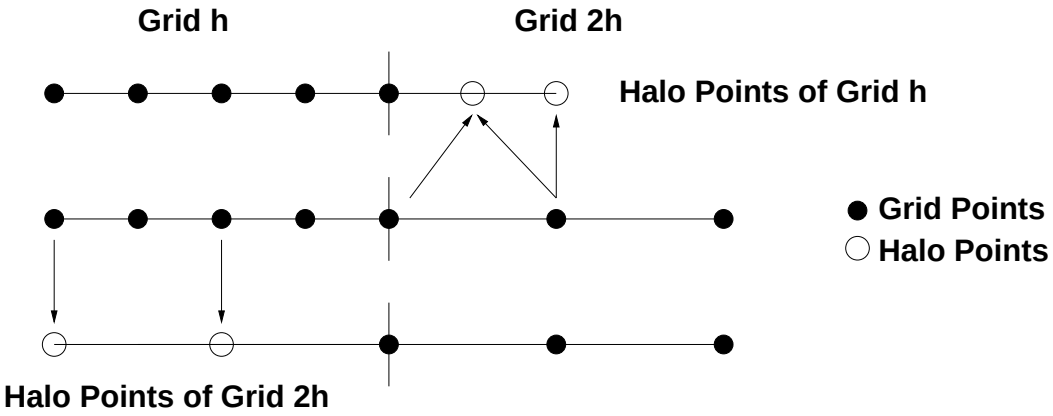

Figure 1: Interpolation Between FD Grids (1-D)

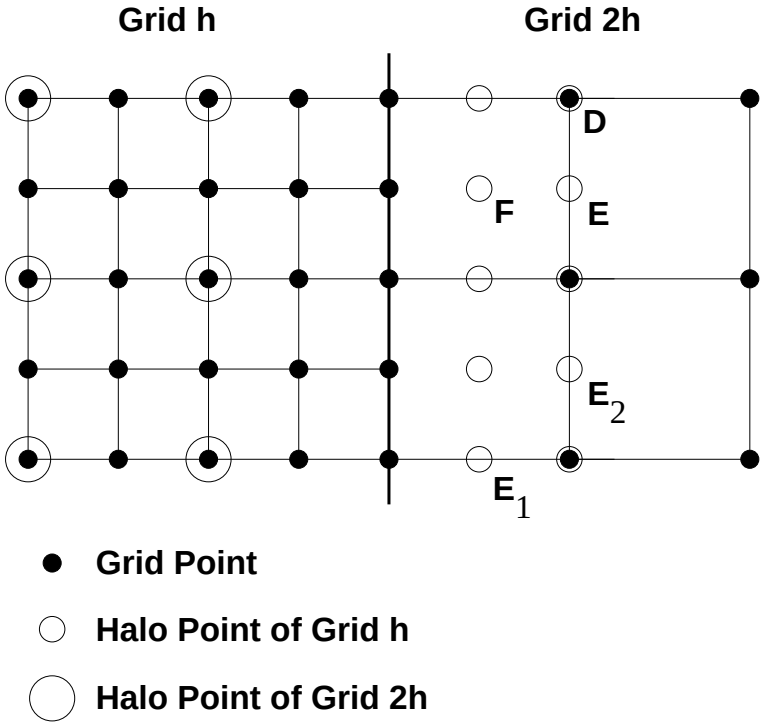

Figure 2: Interpolation Between FD Grids (2-D)

3 POST-PROCESSING INTERPOLATION

An interesting option when trying to maximize modularity is to transfer all direct injection points first from *2h* to *h*, and then obtain the missing information by post-processing this data on *grid h*. In 2-D, the cases that need to be considered are:

- E_1: Edge-points aligned with grid-lines from grid h
- E_2: Edge-points not aligned with grid-lines from grid h
- F: Face-points

The distinguishing factor for points of type E_1 is that information from the interior of *grid h* is readily available and can be used to improve the interpolation order. Fig. 3 shows some of the

possibilities, together with the interpolation weights. For the points of type E_2 usual high order Lagrangian interpolation schemes are employed. Fig. 4 shows some of the possibilities, together with the interpolation weights. Points of type F may be interpolated either via a weighted average of the surrounding edge-points, or by treating them as points of type E_1 with the extra information required obtained previously for the points of type E_2 (see Fig. 5).

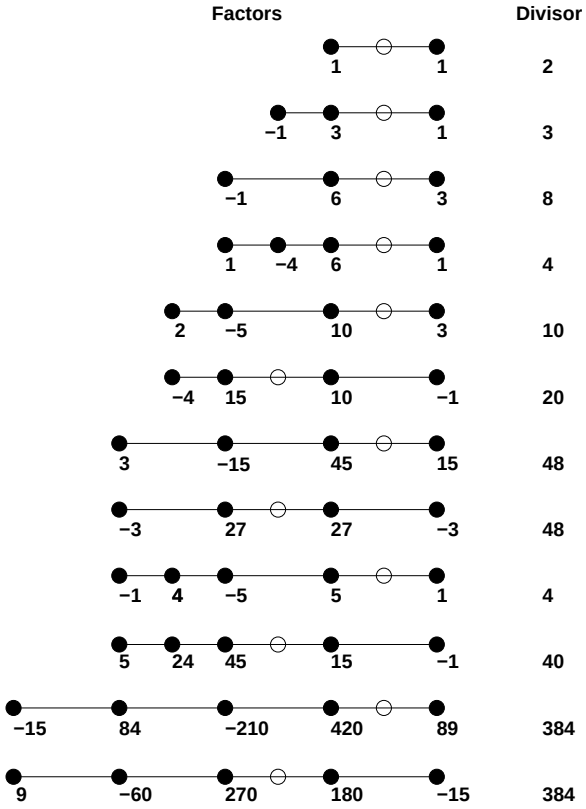

Figure 3: Interpolation Factors for Edges of Type E_1

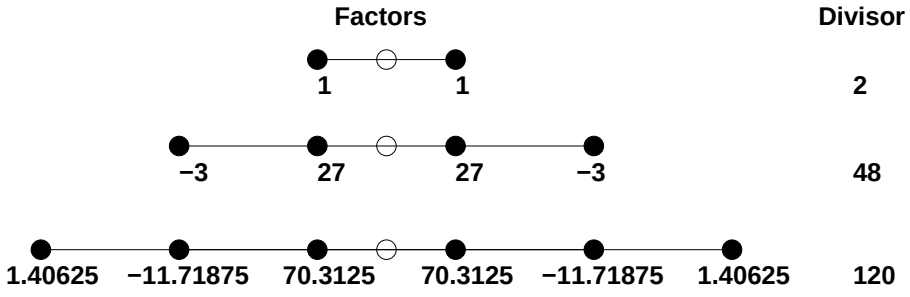

Figure 4: Interpolation Factors for Edges of Type E_2

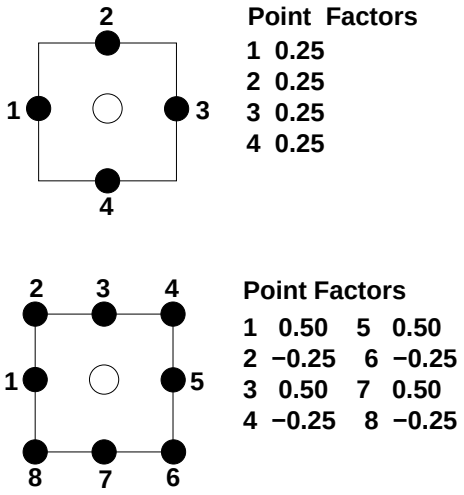

Figure 5: Interpolation Factors for Faces

4 FINITE DIFFERENCE NAVIER-STOKES SOLVER (FDFLO)

4.1 EQUATIONS SOLVED

FDFLO solves the weakly compressible Navier-Stokes equations. The temperature is included as an option, as well as the Boussinesq approximation for natural convection. The system of PDEs is given by:

$$\frac{1}{c^2}p_{,t} + \rho\nabla\cdot\mathbf{v} = 0 \ , \tag{1}$$

$$\rho\mathbf{v}_{,t} + \rho\mathbf{v}\nabla\mathbf{v} + \nabla p = \nabla\mu\nabla\mathbf{v} + \rho\mathbf{g} + \beta\rho\mathbf{g}(T - T_0) + S_v \ , \tag{2}$$

$$\rho c_p T_{,t} + \rho c_p\mathbf{v}\nabla T = \nabla\lambda\nabla T + S_T \ , \tag{3}$$

where $\rho, \mathbf{v} = (u, v, w), p, T, c, \mu, c_p, \lambda, \beta, T_0$ denote, respectively, the density, velocity, pressure, temperature, (constant) speed of sound, viscosity, heat capacitance, conductivity, thermal expansion and reference temperature for the fluid, and $\mathbf{g}, S_v, S_T$ the gravity vector and source terms for velocities and temperature.

4.2 NUMERICS

The numerics implemented may be summarized as follows:

- Explicit timestepping via low-storage Runge-Kutta schemes;
- Conservative formulation for advection and divergence;
- Easy extensions to high-order stencils;
- Ordered access to memory;
- Minimum access to memory;
- Long 1-D loops (for optimal vector, OMP and GPU performance);

- Use of halo points to impose boundary conditions and enable easy extension to massively parallel machines.

In the sequel, we describe in detail the discretizations employed.

4.3 DISCRETIZATION IN SPACE

The spatial discretization is carried out via Finite Differences on a cartesian grid with equal mesh size in all directions:

$$h_x = h_y = h_z = \Delta x \ . \tag{4}$$

All fluxes are written in conservative form as:

$$\mathbf{r}_i = \frac{1}{\Delta x}\left[\mathbf{f}^x_{i+1/2} - \mathbf{f}^x_{i-1/2} + \mathbf{f}^y_{i+1/2} - \mathbf{f}^y_{i-1/2} + \mathbf{f}^z_{i+1/2} - \mathbf{f}^z_{i-1/2}\right] \ . \tag{5}$$

Each flux is composed of the physical and the artificial dissipation flux, e.g.:

$$\mathbf{f}_{i+1/2} = \mathbf{f}^p_{i+1/2} - \mathbf{f}^d_{i+1/2} \ . \tag{6}$$

The advective physical fluxes are obtained from central difference operators of 2nd, 4th, 6th and 8th order:

$$\mathbf{f}^p_{i+1/2}\Big|^{II} = \frac{1}{2}\left(\mathbf{f}^p_{i+1} + \mathbf{f}^p_i\right) \tag{7.2}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{IV} = \frac{7}{12}\left(\mathbf{f}^p_{i+1} + \mathbf{f}^p_i\right) - \frac{1}{12}\left(\mathbf{f}^p_{i+2} + \mathbf{f}^p_{i-1}\right) \tag{7.4}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VI} = \frac{37}{60}\left(\mathbf{f}^p_{i+1} + \mathbf{f}^p_i\right) - \frac{8}{60}\left(\mathbf{f}^p_{i+2} + \mathbf{f}^p_{i-1}\right) + \frac{1}{60}\left(\mathbf{f}^p_{i+3} + \mathbf{f}^p_{i-2}\right) \tag{7.6}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VIII} = \frac{533}{840}\left(\mathbf{f}^p_{i+1} + \mathbf{f}^p_i\right) - \frac{139}{840}\left(\mathbf{f}^p_{i+2} + \mathbf{f}^p_{i-1}\right) + \frac{29}{840}\left(\mathbf{f}^p_{i+3} + \mathbf{f}^p_{i-2}\right) - \frac{3}{840}\left(\mathbf{f}^p_{i+4} + \mathbf{f}^p_{i-3}\right) \tag{7.8}$$

For the 2nd order (Laplacian) operators the physical fluxes of 2nd, 4th, 6th and 8th order are:

$$\mathbf{f}^p_{i+1/2}\Big|^{II} = \left(\mathbf{f}^p_{i+1} - \mathbf{f}^p_i\right) \tag{8.2}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{IV} = \frac{15}{12}\left(\mathbf{f}^p_{i+1} - \mathbf{f}^p_i\right) - \frac{1}{12}\left(\mathbf{f}^p_{i+2} - \mathbf{f}^p_{i-1}\right) \tag{8.4}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VI} = \frac{245}{180}\left(\mathbf{f}^p_{i+1} - \mathbf{f}^p_i\right) - \frac{25}{180}\left(\mathbf{f}^p_{i+2} - \mathbf{f}^p_{i-1}\right) + \frac{2}{180}\left(\mathbf{f}^p_{i+3} - \mathbf{f}^p_{i-2}\right) \tag{8.6}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VIII} = \frac{7175}{5040}\left(\mathbf{f}^p_{i+1} - \mathbf{f}^p_i\right) - \frac{889}{5040}\left(\mathbf{f}^p_{i+2} - \mathbf{f}^p_{i-1}\right) + \frac{119}{5040}\left(\mathbf{f}^p_{i+3} - \mathbf{f}^p_{i-2}\right) - \frac{9}{5040}\left(\mathbf{f}^p_{i+4} - \mathbf{f}^p_{i-3}\right) \tag{8.8}$$

These (unstable) approximations are stabilized by adding an appropriate artificial viscosity / damping (Hirsch, 1991; Kallinderis and Chen, 1996; Löhner, 2008; Sitaraman et al., 2008) of the form:

$$\mathbf{f}^d_{i+1/2}\Big|^{II} = c_d\lambda_{i+1/2}\left[(\mathbf{u}_{i+1} - \mathbf{u}_i)\right] \tag{9.2}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{IV} = c_d\lambda_{i+1/2}\left[3\left(\mathbf{u}_{i+1} - \mathbf{u}_i\right) - \left(\mathbf{u}_{i+2} - \mathbf{u}_{i-1}\right)\right] \tag{9.4}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VI} = c_d\lambda_{i+1/2}\left[10\left(\mathbf{u}_{i+1} - \mathbf{u}_i\right) - 5\left(\mathbf{u}_{i+2} - \mathbf{u}_{i-1}\right) + \left(\mathbf{u}_{i+3} - \mathbf{u}_{i-2}\right)\right] \tag{9.6}$$

$$\mathbf{f}^p_{i+1/2}\Big|^{VIII} = c_d\lambda_{i+1/2}\left[35\left(\mathbf{u}_{i+1} - \mathbf{u}_i\right) - 21\left(\mathbf{u}_{i+2} - \mathbf{u}_{i-1}\right) + 7\left(\mathbf{u}_{i+3} - \mathbf{u}_{i-2}\right) - \left(\mathbf{u}_{i+4} - \mathbf{u}_{i-3}\right)\right] \tag{9.8}$$

where λ is the maximum eigenvalue of the system

$$\lambda = |\mathbf{v}| + c \tag{10}$$

and c_d the artificial viscosity / damping coefficient. Typical values are: $c_d^{II} = 0.2, c_d^{IV} = 0.10, c_d^{VI} = 0.02, c_d^{VIII} = 0.02$. For viscous cases, the artificial viscosity / damping coefficient of the momentum equations is reduced:

$$c_d^* = c_d \cdot r(u) \;\;,\;\; r(u) = max(0, min(1, Re_h - 1)) \;\;,\;\; Re_h = \frac{\rho u \Delta x}{\mu} \;\;. \tag{11}$$

Note that as the mesh is refined and the cell Reynolds-number Re_h falls below $Re_h = 1$, the artificial viscosity vanishes. The same type of advection and artificial viscosity is also used for the temperature equation, but limiting with the local Peclet-number.
Finally, for the **Boussinesq terms** the approximation taken is simply:

$$\mathbf{r}_i = \mathbf{g}\beta\left(T_B - T_i\right) \;\;. \tag{12}$$

The same is done for the source-terms S_v, S_T.

In order to achieve long vector loops the formation of right-hand sides (RHSs) is carried out by forming a single array of point data. Given `nx, ny, nz`, and defining `nxny = nx * ny`, the points are traversed as `ip = nxny*(iz-1) + ny*(iy-1) + ix`. In order to minimize the use of registers, the RHSs are formed dimension by dimension.

4.4 BOUNDARY CONDITIONS

A variety of boundary conditions are required for practical computations. An easy way to implement these is via halo points. In order to be able to cast all operations in terms of large loops over all points, two (for 4th order stabilized fluxes), three (for 6th order stabilized fluxes) or four (for 8th order stabilized fluxes) halo points are added at the minimum and maximum extent of each dimension. This increases the total point count, but allows for maximal vector length. The boundary conditions are then imposed as follows:

- No-Slip (NavSto) Wall: Same $p, \mathbf{v}, T$ as wall;
- Symmetry/Euler Wall: same $p, \mathbf{v}\cdot\mathbf{t}, T$, opposite $\mathbf{v}\cdot\mathbf{n}$;
- Inflow to Field: $\mathbf{v}, T$ imposed, p free;
- Outflow of Field: p imposed, $\mathbf{v}, T$ free.

4.5 IMMERSED BODY OPTION

A solver based on Cartesian Finite Differences may be very fast, but its use is very limited when considering geometrically complex objects. There are two options of treating these: either via immersed body methods, or via embedded surface techniques. We have implemented both.

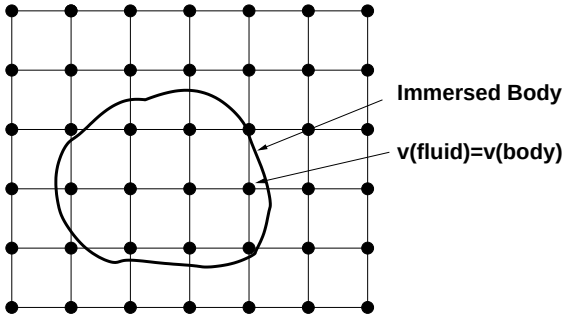

Figure 1 Immersed Body Approach

The **immersed body methods** (Mohd-Yusof, 1997; Ye et al., 1999; Fadlun et al., 2000; Gilmanov et al., 2003; Balaras, 2004; Gilmanov and Sotiropoulos, 2005; Mittal and Iaccarino, 2005; Yang and Balaras, 2006; Storti et al., 2013) (see Fig. 1) may be classified by the way they apply the boundary conditions into kinematic or kinetic (Löhner, 2008).
The **kinematic** approaches simply set the velocity of the flow to the velocity of the body:

$$\mathbf{v}_{flow} = \mathbf{v}_{body} \quad . \tag{13}$$

This 1st order scheme may be improved via interpolation or weighting functions, i.e. taking into account neighbour information (Balaras, 2004).
The **kinetic** (i.e. force-based) approaches add a force to the momentum equations such that Eqn.(13) is fulfilled at the end of the timestep, or a penalty term that imposes Eqn.(13) weakly. In either case, the immersed body options described require the following steps:

- Read in the immersed bodies as a mesh (elements, coordinates, velocities, temperature);
- Determine the cartesian points inside the bodies; this is done by performing a loop over the elements of the immersed bodies; for each of these the cartesian points inside the element are obtained and marked;
- Store the points marked in a separate boundary point array.

In order to be as general as possible, the immersed bodies are defined via tetrahedral meshes.

4.6 EMBEDDED SURFACE OPTION

The second way to treat geometrically complex objects within a Cartesian Finite Difference code is via embedded surface techniques (Clarke et al., 1985; Melton et al., 1993; Pember et al., 1995; Aftosmis et al., 2000; Dadone and Grossman, 2002; Nakahashi and Kim, 2004; Peller et al., 2006) (see Fig. 2).

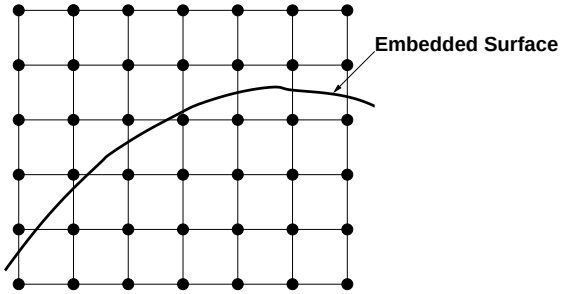

Figure 2 Embedded Surface Approach

The key idea is to 'extend' the stencil past the edges crossed by the embedded surfaces, mirror these points back into the proper computational domain, and then use mirroring or other boundary conditions to impose the presence of the embedded surfaces (see Fig. 3). This way of imposing the boundary conditions for embedded surfaces is theoretically 2nd order accurate. However, in many cases (particularly if the points are very close to the embedded surface, or in narrow passages) the available information for interpolation is not sufficient to guarantee this order.

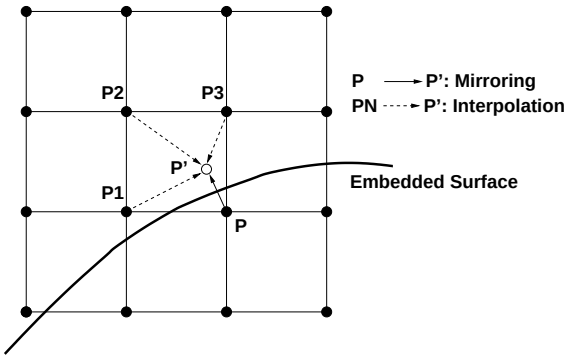

Figure 3 Embedded Surface Approach: Boundary Conditions

The embedded surface option requires the following steps:

- Read in the embedded surfaces as a triangulation (elements, coordinates, velocities, temperature);
- Determine the cartesian edges crossed by the surface; this is done by performing a loop over the triangles of the embedded surfaces; for each of these the cartesian edges crossed are obtained and marked;
- Determine the interpolation conditions for the points 'on the other side' of the crossed edges.
- Store the points marked in a separate boundary point array.

There are several possible ways of incorporating the values of the unknowns required to imposed embedded surfaces. We have pursued a dual-loop approach: The first loop over the points is the usual one, i.e. it ignores the embedded surface boundary conditions. The second loop, over the points whose surrounding edges are crossed by embedded surfaces, substracts the right hand of the 1st loop, and adds the right hand side replacing the unknowns 'on the

other side' with the proper values. In this way, the extra burden of imposing embedded surface boundary conditions is minimized.

4.7 DISCRETIZATION IN TIME

After spatial discretization, the original PDEs given by Eqns.(1-3) form a coupled system of ordinary differential equations (ODEs) of the form:

$$\mathbf{u}_{,t} = \mathbf{r}(t, \mathbf{u}) \ . \tag{14}$$

This system is solved using explicit, low-storage **Runge-Kutta** methods of the form:

$$\Delta \mathbf{u}^{n+i} = \alpha_i \, \Delta t \, \mathbf{r}(\mathbf{u}^n + \Delta \mathbf{u}^{n+i-1}) \ , \ \ i = 1, s \ , \ \ \Delta \mathbf{u}^0 = 0 \ , \tag{15}$$

or via the usual 4-stage Runge-Kutta scheme (Butcher, 2003). We remark that for linear ODEs the choice

$$\alpha_i = \frac{1}{s+1-i} \ , \ \ i = 1, s \tag{16}$$

in combination with Eqn.(15) leads to a scheme that is s-th order accurate in time. Like all explicit schemes, the allowable timestep is bounded by the condition:

$$\Delta t < CFL \cdot min \left(\frac{h}{|\mathbf{v}| + c} \, , \, \frac{\rho h^2}{\mu} \, , \, \frac{\rho c_p h^2}{\lambda} \right) \ , \tag{17}$$

where the allowable CFL factor is proportional (i.e. increasing) with the number of stages s. One complete timestep is given by the following steps:

- Apply BC / Transfer Info from Domain to Halo Points
- Get Allowable Timestep
- Set Timestep $\Delta t = 0$ for Halo Points
- Loop Over the Stages:
  - Set $\mathbf{r} = 0$
  - Compute $\mathbf{r}$
  - Obtain $\Delta \mathbf{u} = \alpha_i \Delta t \mathbf{r}(\mathbf{u})$
  - Apply Boundary Conditions
  - Update $\mathbf{u}$

4.8 MULTIBLOCK OPTION

A solver based on Cartesian Finite Differences may be very fast, and may be made applicable to complex geometries via immersed or embedded techniques. However, its use is still very limited when considering problems with varying spatial lengthscales. The best way of addressing this problem while keeping the speed advantages of the basic solver is via multiblocking (Berger and Oliger, 1984; Pember et al., 1995). The key idea is to consider each cartesian grid or block independently, and to combine these by interpolating the unknowns of the halo points from the adjacent blocks.

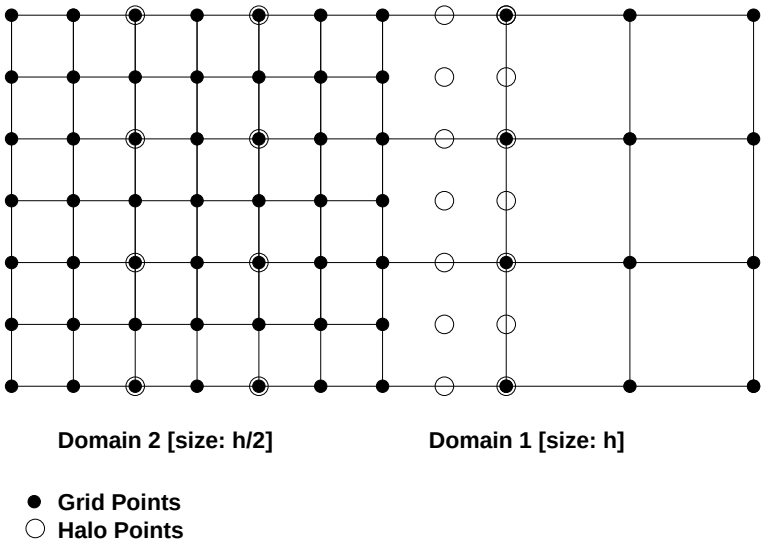

Figure 4 Multiblock Option

5 EXAMPLES

The interpolation schemes developed were implemented and tested in FDFLO. In the sequel, we show the performance of the difference options implemented.

5.1 Lamb-Vortex

The so-called Lamb-vortex, centered at $x, y = 0$, was chosen to assess the accuracy of the interpolation schemes developed. The unknowns are given by:

$$u = u_0 - \frac{\alpha}{2\pi} y e^{\phi(1-x^2-y^2)} \;\; ; \;\; v = \frac{\alpha}{2\pi} x e^{\phi(1-x^2-y^2)} \;\; ; \;\; p = -\left(\frac{\alpha}{2\pi}\right)^2 \frac{1}{4\phi} e^{2\phi(1-x^2-y^2)} \;\; , \qquad (18)$$

and $\mu = 0$. For the particular case tested, the domain was given by $-5 \leq x, y \leq 5$, and $\alpha = 1$, $\phi = 0.5$, $c = 1$, $\rho = 1$, and the grids were of size $h = 0.125$ and $2h = 0.250$. The vortex was propagated for $T = 200$ time units. Given that the domain is doubly periodic, the vortex should reappear in the exact location as at time $T = 0$ after traversing the mesh twice.
Fig. 5.1 shows the initial conditions for the *mesh h2h*, where the discretizations used are clearly visible. Fig. 5.2-3 show the pressure and velocity obtained using an 8th order spatial discretization and a 5th order low-storage Runge-Kutta timestepping scheme. From left to right, the cases are: *mesh 2h*,*mesh h2h* (i.e. mesh h inside mesh 2h) with usual bilinear interpolation, *mesh h2h* with cubic interpolation, and *mesh h*. One can see that for the case with bilinear interpolation, running the case on the *2h mesh* yields better results than the *h2h mesh*, defeating the purpose of mesh refinement. This should be a cause of alarm if one considers complex flow problems where vortices and other flow structures will traverse grids with different mesh sizes. On the other hand, the cubic interpolation does yield results on the *h2h mesh* that are demonstrably better than those on the *2h mesh*, indicating the potential of the procedures developed.

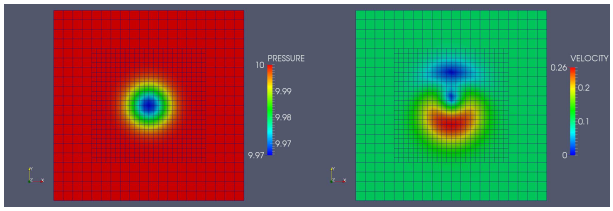

Figure 5.1: Lamb Vortex: Initial Conditions, *h2h Mesh*

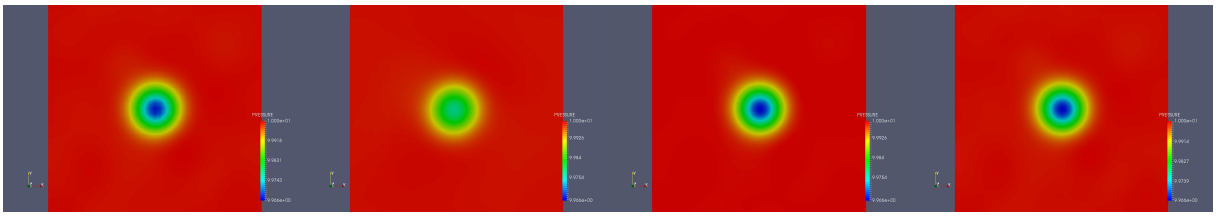

Figure 5.2: Lamb Vortex: Comparison of Pressures at T_f:
2h, h2h with Bilinear interpolation, h2h with Cubic interpolation, h

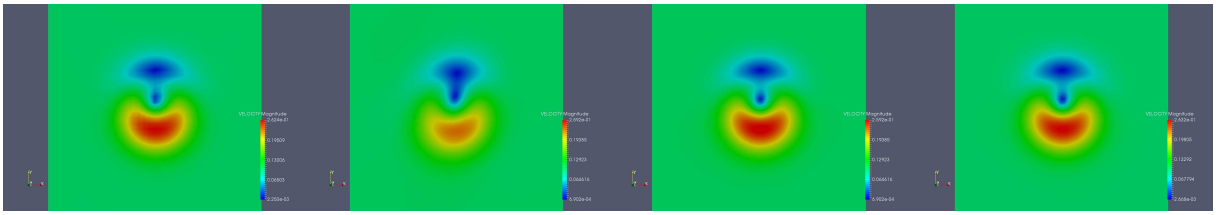

Figure 5.3: Lamb Vortex: Comparison of Velocity Magnitudes at T_f:
2h, h2h with Bilinear interpolation, h2h with Cubic interpolation, h

5.2 Convergence Study With Stationary Lamb-Vortex

In order to quantify the relative merit of the different interpolation schemes, a series of convergence studies were carried out. The same domain as before was used, but the boundary conditions were changed from periodic to gliding wall. The right half of the domain was of size h, the left of size $2h$. At the same time, uniform grids of size h and $2h$ were run for comparison. A stationary Lamb-Vortex was set as the initial condition. This is an exact steady solution, so the initial residual can be used to measure the convergence of the schemes. A typical configuration is shown in Fig. 6.1.

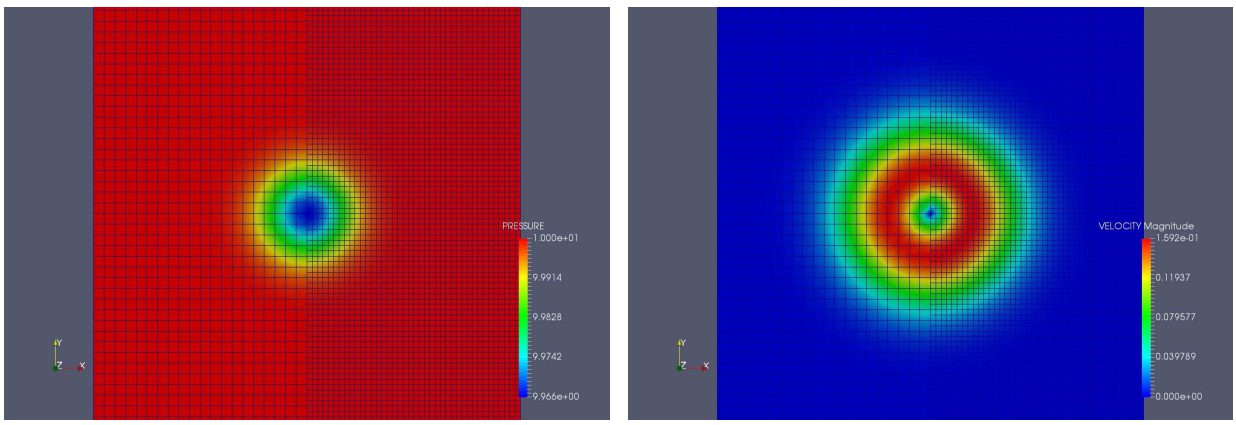

Figure 6.1: Stationary Lamb Vortex: Typical Initial Configuration

Fig. 6.2-5 show the convergence obtained for the 6th and 8th order solvers, together with different interpolation schemes. The notation is as follows: L2 denotes the $L2$ norm, LI the L_∞ norm, P the pressure, V the velocity, UNH and U2H the convergence on uniform grids of size h and $2h$, and M00, M33, M43 the cases of mixed $h, 2h$ grids with simple bi/trilinear interpolation, cubic interpolation and quartic interpolation. As expected, for the $L2$ norm the errors of the high order interpolation schemes fall between the values for uniform grids of size h and $2h$. This is not always the case for the L_∞ norm. Furthermore, one can see the serious negative effect on convergence of the bi/trilinear interpolation. The results show that the aim of interpolation schemes that are balanced and appropriate to the spatial discretization while being local and fast has been achieved.

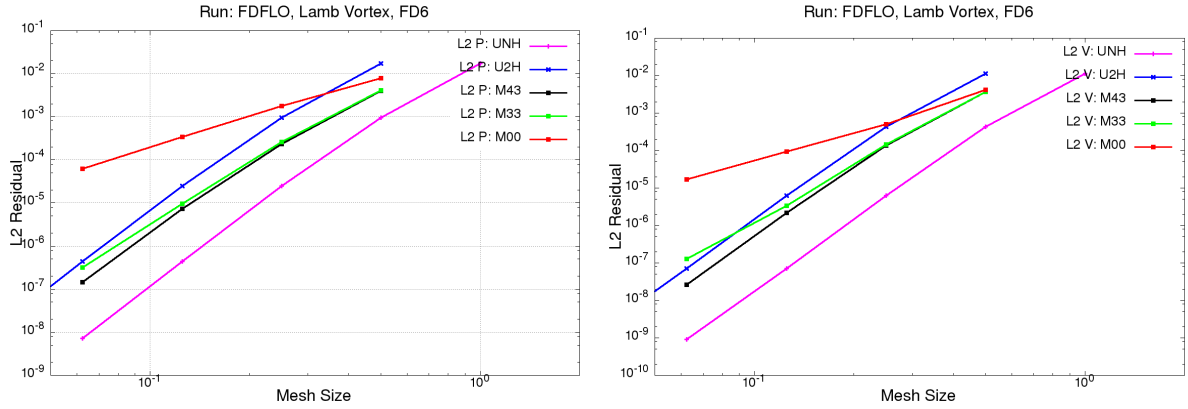

Figure 6.2: Stationary Lamb Vortex: $L2$ Convergence for 6th Order

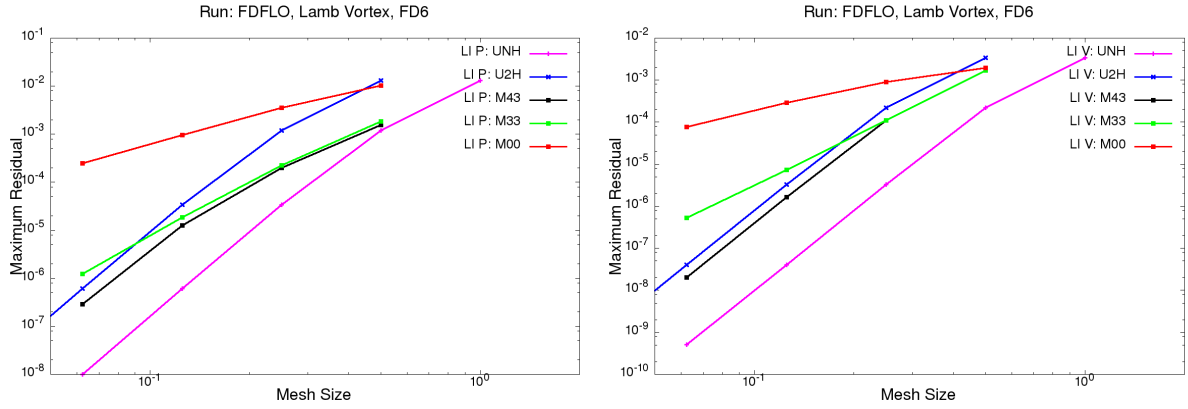

Figure 6.3: Stationary Lamb Vortex: L_∞ Convergence for 6th Order

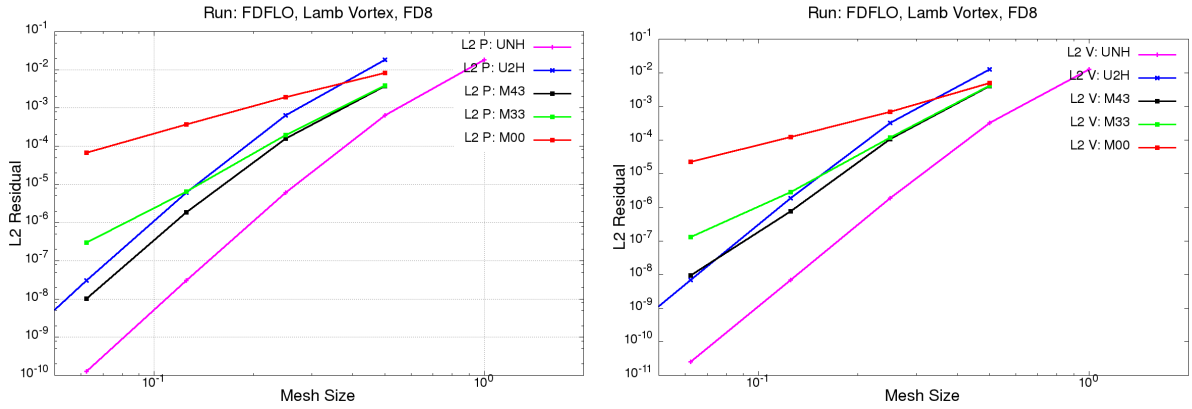

Figure 6.4: Stationary Lamb Vortex: $L2$ Convergence for 8th Order

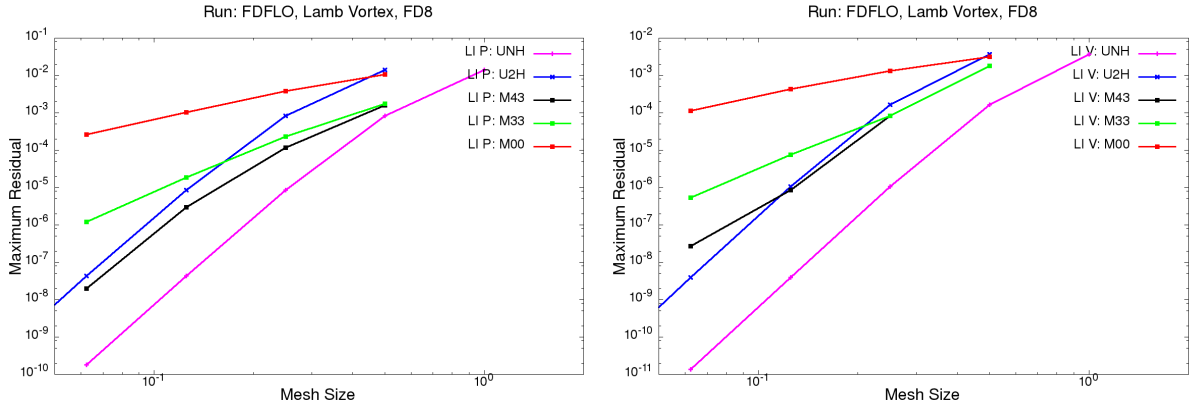

Figure 6.5: Stationary Lamb Vortex: L_∞ Convergence for 8th Order

5.3 Cylinder

This classic testcase was added in order to show the effect of high-order interpolation for wake flows. The domain considered was $-4 \leq x \leq 10$, $-2 \leq y \leq 2$, with gliding wall boundary conditions at y_{min}, y_{max}, prescribed uniform inflow and prescribed pressure at outflow. As can be seen from Fig. 6.1, the mesh consisted of 10 domains, with three levels of refinement: $\Delta x = 0.100, 0.050, 0.025$. The physical parameters were set as follows: $\rho = 1.0, \mathbf{v}_\infty = (1, 0, 0), \mu = 0.01, c = 5$, and the diameter of the cylinder was $d = 1.0$, yielding a Reynolds number of $Re = 100$. A 6-stage, low-storage RK scheme was used to integrate in time with a Courant-number of $C = 0.4$. The immersed boundary option was used with a spatial discretization of 6th order. The case was run with the usual, low-order bi/trilinear interpolation, and also with the high-order interpolation. The results obtained for the latter one at $T = 100$ are shown in Fig. 7.2. A number of station time history points were placed in the flow and the results recorded. Fig. 7.3-4 show the values for the pressure, x- and y-velocities for two stations. As expected, the most pronounced difference can be observed in the y-velocities. However, even the pressures show larger variations in time for the high-order interpolation, indicating less dissipation. Note also that a slight change of frequency is incurred when changing interpolation order.

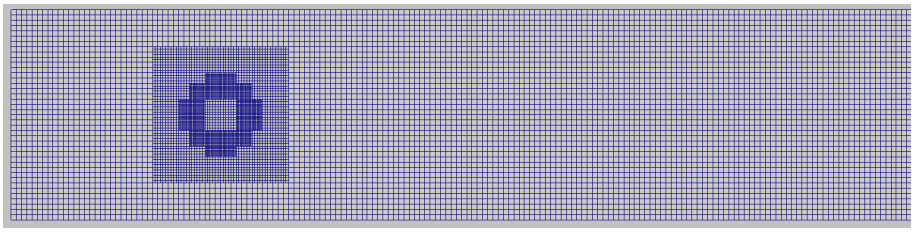

Figure 7.1: Cylinder: Grid System Used

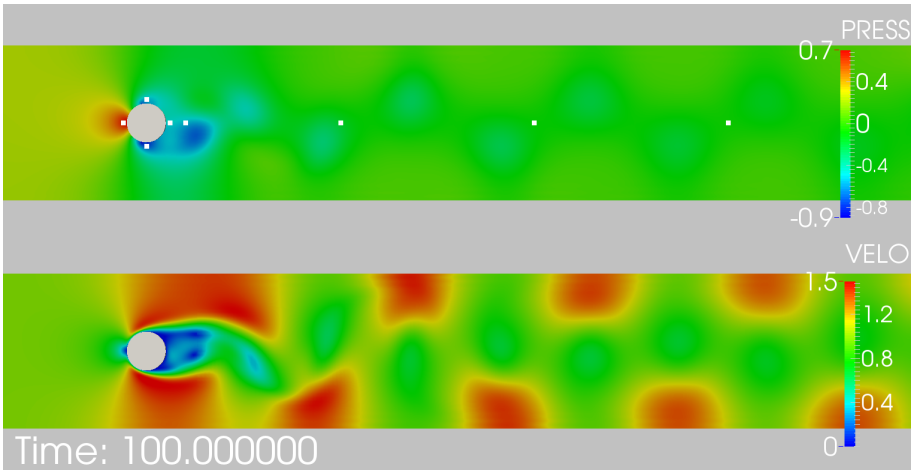

Figure 7.2: Cylinder: Results at Time $T = 200$

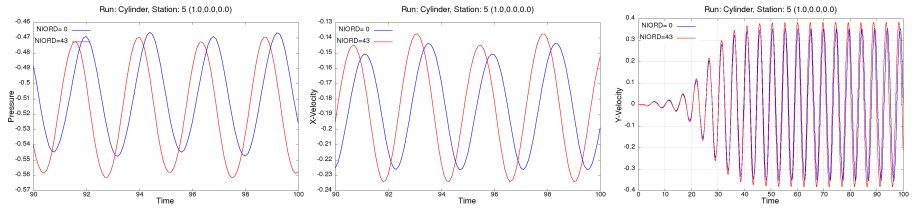

Figure 7.3: Cylinder: Station Time History for Station 5: $\mathbf{x} = (1, 0, 0)$

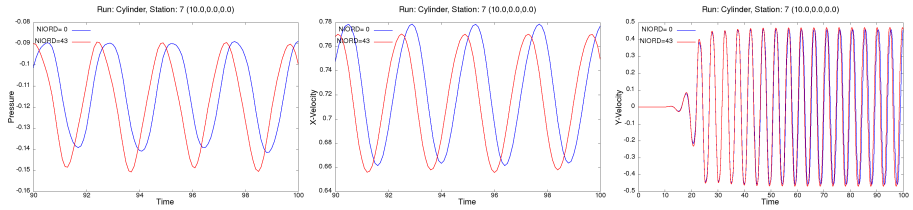

Figure 7.4: Cylinder: Station Time History for Station 7: $\mathbf{x} = (10, 0, 0)$

6 CONCLUSIONS AND OUTLOOK

A series of interpolation algorithms for nested cartesian grids of different size were developed and tested. These algorithms are based on post-processing, on each local grid, the raw (bi/trilinear) information passed to the halo points from coarser grids. In this way modularity is maximized while preserving locality.

The results obtained indicate that the schemes improve markedly the convergence rates and the overall accuracy of finite difference codes with varying grid sizes.

Current work is centered on quantifying these results for more realistic cases, including some

large-scale complex 3-D flows around cars and helicopters.

REFERENCES

Aftosmis M., Berger M., and Adomavicius G. A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries. *AIAA-00-0808*, 2000.

Balaras E. Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. *Comp. Fluids*, 33:375–404, 2004.

Berger M. and Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53(3):484–512, 1984.

Butcher J. *Numerical Methods for Ordinary Differential Equations*. J. Wiley, 2003.

Clarke D., Hassan H., and Salas M. Euler calculations for multielement airfoils using cartesian grids. *AIAA-85-0291*, 1985.

Dadone A. and Grossman B. An immersed boundary methodology for inviscid flows on cartesian grids. *AIAA-02-1059*, 2002.

Fadlun E., Verzicco R., Orlandi P., and Mohd-Yusof J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comp. Phys.*, 161:35–60, 2000.

Gilmanov A. and Sotiropoulos F. A hybrid cartesian/immersed boundary method for simulating flows with 3-d, geometrically complex moving objects. *J. Comp. Phys.*, 207:457–492, 2005.

Gilmanov A., Sotiropoulos F., and Balaras E. A general reconstruction algorithm for simulating flows with complex 3d immersed boundaries on cartesian grids. *J. Comp. Phys.*, 191, 2:660–669, 2003.

Hirsch C. *Numerical Computation of Internal and External Flow*. J. Wiley & Sons, 1991.

Hittinger J. and Banks J.W. Block-structured adaptive mesh refinement algorithms for vlasov simulation. *J. Comp. Phys.*, 241:118–140, 2013.

Kallinderis Y. and Chen A. An incompressible 3-d navier-stokes method with adaptive hybrid grids. *AIAA-96-0293*, 1996.

Löhner R. *Applied CFD Techniques, Second Edition*. J. Wiley & Sons, 2008.

Löhner R., Corrigan A., Wichmann K.R., and Wall W. Comparison of lattice-boltzmann and finite difference solvers. *AIAA-2014-1439*, 2014.

McCorquodale P. and Collella P. A high-order finite-volume method for conservation laws on locally refined grids. *Comm. in Applied Mathematics and Computational Science*, 6:1, 2011.

Melton J., Berger M., and Aftosmis M. 3-d applications of a cartesian grid euler method. *AIAA-93-0853-CP*, 1993.

Mittal R. and Iaccarino G. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

Mohd-Yusof J. Combined immersed-boundary/b-spline methods for simulations of flow in complex geometries. *CTR Annual Research Briefs, NASA Ames Research Center/ Stanford Univ.*, pages 317–327, 1997.

Nakahashi K. *Building-Cube Method for Flow Problems with Broadband Characteristic Length*. Computational Fluid Dynamics, (S.W. Armfield, P. Morgan and K. Srinivas eds), Springer, Berlin, 2003.

Nakahashi K. and Kim L.S. Building-cube method for large-scale, high-resolution flow computations. *AIAA-04-0434*, 2004.

Peller N., LeDuc A., Tremblay F., and Manhart M. High-order stable interpolations for immersed boundary methods. *Int. J. Num. Meth. Fluids*, 252:1175–1193, 2006.

Pember R., Bell J., Colella P., Crutchfield W., and Welcome M. An adaptive cartesian grid

method for unsteady compressible flow in irregular regions. *J. Comp. Phys.*, 120:278, 1995.

Ray J., Kennedy C., Lefantzi S., and Najm H. Using high-order methods on adaptively refined block structured meshes – discretizations, interpolations and filters. *SAND2005-7981, Sandia National Laboratories, CA.*, 2005.

Sitaraman J., Katz A., Jayaraman B., Wissink A., and Sankaran V. Evaluation of a multi-solver paradigm for cfd using overset unstructured and structured adaptive cartesian grids. *AIAA-2008-0660*, 2008.

Sitaraman J., Lakhminarayan V., Roget B., and Wissink A. Progress in strand mesh generation and domain connectivity for dual-mesh cfd simulations. *AIAA-2017-0288*, 2017.

Storti M., Paz R., Dalcin L., Costarelli S., and Idelsohn S. A fft preconditioning technique for the solution of incompressible flow on gpuâĂŹs. *Computers and Fluids*, 74:44–57, 2013.

Yang J. and Balaras E. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *J. Comp. Phys.*, 215:12–40, 2006.

Ye T., Mittal R., Udaykumar H., and Shyy W. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. *J. Comp. Phys.*, 156:209–240, 1999.