INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2006; **52**:1175–1193
Published online 19 April 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1227

High-order stable interpolations for immersed boundary methods

Nikolaus Peller[1,*,†], Anne Le Duc[1], Frédéric Tremblay[2] and Michael Manhart[1]

[1]*Fachgebiet Hydromechanik, Technische Universität München, Arcisstr. 21, 80290 Munich, Germany*
[2]*Newmerical Technologies Int, 680 Sherbrooke Street West, Montreal, Canada H3A 2M7*

SUMMARY

The analysis and improvement of an immersed boundary method (IBM) for simulating turbulent flows over complex geometries are presented. Direct forcing is employed. It consists in interpolating boundary conditions from the solid body to the Cartesian mesh on which the computation is performed. Lagrange and least squares high-order interpolations are considered. The direct forcing IBM is implemented in an incompressible finite volume Navier–Stokes solver for direct numerical simulations (DNS) and large eddy simulations (LES) on staggered grids. An algorithm to identify the body and construct the interpolation schemes for arbitrarily complex geometries consisting of triangular elements is presented. A matrix stability analysis of both interpolation schemes demonstrates the superiority of least squares interpolation over Lagrange interpolation in terms of stability. Preservation of time and space accuracy of the original solver is proven with the laminar two-dimensional Taylor–Couette flow. Finally, practicability of the method for simulating complex flows is demonstrated with the computation of the fully turbulent three-dimensional flow in an air-conditioning exhaust pipe. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: immersed boundary (IB); Lagrange interpolation; least squares interpolation; large eddy simulation (LES); complex flow; matrix stability analysis

1. INTRODUCTION

With the development of memory and speed of computers, the computation of time-dependent flows around complex geometries has become a feasible task. A widely used method consists in representing the geometry with body-fitted coordinates. The boundaries of the computational mesh then coincide with the body surface. But solvers for curvilinear or unstructured

*Correspondence to: Nikolaus Peller, Fachgebiet Hydromechanik, Technische Universität München, Arcisstr. 21, 80290 Munich, Germany.
†E-mail: n.peller@bv.tum.de

Contract/grant sponsor: Publishing Arts Research Council; contract/grant number: 98–1846389

Received 20 July 2005
Revised 10 February 2006
Accepted 16 February 2006

meshes are less efficient than Cartesian solvers in terms of computational time and memory requirements [1]. Additionally, generating a body-fitted grid around an industrial geometry can be very time-consuming. The immersed boundary method (IBM) allows to take into account the effect of the body on a flow while using a Cartesian solver. The easy meshing and efficiency of Cartesian solvers is retained. An extensive review of the different IBMs along with their strengths and limitations can be found in References [2, 3]. Here we restrict ourselves to a brief review of force-field and direct-forcing modelling and focus on the precision and time-efficiency of the method.

The idea of modelling a solid boundary with a force field dates back to Peskin [4]. He computed two-dimensional biological flows within elastically deformable bodies. The method relies on a coupled time advancement of the equations for the position of the elastic wall and for the fluid velocity. It is originally first-order accurate and has been refurbished by Lai and Peskin [5] to be second-order accurate. The method is computationally cumbersome because of the coupling between a Lagrangian approach (for the boundaries) and an Eulerian approach (for the fluid).

Goldstein *et al.* [6] compute the flow around rigid bodies. The physical force can then be replaced with a source term in the momentum equation that brings the velocity at the solid boundary to a prefixed target value. Employing this feedback forcing within a Cartesian spectral code, Goldstein *et al.* compute the two-dimensional flow around an impulsively started circular cylinder and the flow in a plane and a ribbed channel. Saiki and Biringen [7] use the same forcing to compute the flow around a cylinder with a Reynolds number based on the diameter up to 400. Using a high-order finite difference solver instead of a spectral solver, and spreading the forcing over a few grid points about the solid boundary, they are able to reduce the spurious Gibbs oscillations appearing near the solid boundaries in Reference [6]. The stabilization is attributed to the use of finite difference schemes instead of spectral methods. The method of Saiki and Biringen [7] is at most first order at the boundaries.

A major disadvantage of the feedback forcing is the presence of two case-dependent constants in the forcing term. For flows with high frequencies (typically turbulent flows), the constants must take on high values. But the equations become stiffer with increasing magnitude of these parameters. Fadlun *et al.* [8] note that with feedback forcing, the CFL of the computation may drop 4 orders of magnitude compared to the CFL set by requirements within the core of the flow. Using a partly implicit time integration of the forcing, Fadlun *et al.* are able to raise the CFL, but it remains at least one order of magnitude smaller than without forcing.

Mohd-Yusof [9] realizes a breakthrough with a discrete-time IBM. Discretizing the Navier–Stokes equations temporally between time step n and $n+1$, Mohd-Yusof computes a force term such that the boundary condition at time $n+1$ is exactly enforced. This term is added to the momentum equation. The forcing is called direct because no dynamical process is involved: at each time step, the boundary condition holds regardless of the characteristic frequencies of the flow. The method is computationally time-efficient because the time-step requirement at the boundaries is the same as in the bulk of the flow. For a laminar ribbed channel, the results of Mohd-Yusof compare well with the ones obtained with a body-fitted grid.

In the rest of the paper, we consider only the direct-forcing implementation of IBM. It is practically equivalent with enforcing the boundary condition within the flow. When the boundary does not coincide with Cartesian grid points, an interpolation is needed. The

accuracy of the IBM thus depends on the interpolation type, order and direction. We will now focus on the interpolations.

We first discuss the problem of interpolation direction. Fadlun *et al.* [8] set the velocities 'that, in a linear interpolation approximation, the Cartesian point closest to the boundary would have if the boundary had the prescribed velocity'. With this second-order interpolation, the authors successfully compare their simulation of the flow around a sphere (with Reynolds number from 100 to 5000) with simulations performed with a body-fitted grid (both under assumption of axisymmetrical flow). Verzicco *et al.* [1] simulate the turbulent flow in a piston–cylinder assembly using the same interpolation. Note that the interpolation is performed along the direction where the distance between Cartesian control point and solid boundary is minimal. This one-dimensionality thus introduces arbitrariness when the body surface is not preferentially aligned with one direction of the grid. Kim *et al.* [10] remove it employing bilinear interpolations in their computations of a 2D cylinder up to Reynolds number 100 and of a sphere up to Reynolds number 300. Balaras [11] uses linear interpolation normal to the body surface. His results for a cylinder at Reynolds number 300 and for a wavy channel agree well with computations performed with body fitted grids. Tremblay *et al.* [12] employ weighting to extend one-dimensional Lagrange interpolations of higher order to three-dimensional interpolations. They compute the turbulent flow around a circular cylinder at subcritical Reynolds numbers (DNS at $Re = 3900$ [12] and LES at $Re = 140\,000$ [13]) and find good agreement with experimental data. The problem of preferred direction of interpolation is thus solved.

We now turn to the question of interpolation order. As noted by Fadlun *et al.* [8], linear interpolations require strong clustering in wall vicinity. This is important in particular for the proper representations of quantities known to be strongly non-linear [14]. The turbulent viscosity ν_T for example must be reconstructed in cells near the wall, because the evaluation of its test-filter operations would require information from blocked cells. But linear reconstruction of ν_T can lead to an overestimation of the turbulent viscosity in equilibrium flows where ν_T decreases with a y^{+3} slope near the wall [11]. For practical computations, higher-order interpolations are thus needed. They do not necessarily increase the formal accuracy of the solver (limited by the spatial accuracy in the bulk of the flow), but can allow for more practical point clustering in the vicinity of boundaries. As high-order interpolations are known to be sensitive to numerical instability, a trade-off between accuracy and stability must be found. Majumbar *et al.* [15] and Tseng and Ferziger [16] employ bilinear and quadratic (i.e. third order) boundary interpolation in a globally second-order Cartesian solver. Both use *ad hoc* image points to alleviate instabilities. Kim *et al.* [10] also notice the problem in their bilinear interpolation and introduce an *ad hoc* correction resembling a least squares interpolation. When trying to compute flows around practically relevant geometries using Lagrange high-order interpolations, we were also faced with numerical instabilities. Employing least squares interpolations, we derived stable high-order interpolations.

The goal of this paper is to demonstrate the robustness and accuracy of least squares interpolations within the direct-forcing IBM. Section 2 describes the Cartesian solver and the IBM implementation. Section 3 focuses on the different interpolation schemes. In Section 4, the interpolations are analysed with a matrix stability analysis. Numerical accuracy tests on Taylor–Couette flow are presented in Section 5. Section 6 is devoted to a practically relevant application: the flow exiting a car air-conditioning system is considered. Finally conclusions and perspectives are drawn in Section 7.

2. NUMERICAL METHODOLOGY

2.1. Basic numerical scheme

We solve the Navier–Stokes equations for incompressible flows:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} \tag{2}$$

At solid boundaries, we apply Dirichlet boundary condition for the velocity and Neumann boundary condition for the pressure.

The Navier–Stokes equations are integrated within the standard framework of finite volumes using staggered Cartesian grids [17, 18]. The spatial approximations are second-order accurate and use centred interpolations and differentiations. Time integration is performed via a fractional step method using a leapfrog scheme with time lagged diffusion [19] for the momentum equation (2):

$$u^{n+1} = u^{n-1} + 2\delta t(D(u^{n-1}) + C(u^n) + G(p^{n+1})) \tag{3}$$

where u is the velocity field, D are the diffusive terms, C the convective and G the pressure terms. The pressure p^{n+1} is obtained by the Poisson equation:

$$\mathrm{div}[G(\delta p^{n+1})] = -\frac{1}{2\delta t}\,\mathrm{div}(\tilde{u}^{n+1}) \tag{4}$$

where $\tilde{u}^{n+1}$ is an intermediate velocity field and $\delta p^{n+1} = p^{n+1} - p^n$. The resulting system is solved by Stone's strongly implicit procedure (SIP). See e.g. Reference [20] for discussion of these standard methods.

The solid boundaries are taken into account by the IBM which is the subject of the present paper. Their treatment during the time step is explained in detail in the following section.

2.2. Setting the boundary conditions

Our immersed boundary technique relies on direct forcing [12]. It consists in transforming the boundary condition known at the solid surfaces into internal boundary conditions at the nodes of the Cartesian grid. These internal boundary conditions are set using interpolation from the surrounding nodes. To maintain the order of the scheme and to avoid strong grid clustering at the boundaries, higher order interpolations are used. Before investigating different interpolation schemes in Section 3, we describe the general procedure here.

For setting the internal boundaries, grid cells intersected by the immersed boundaries are identified and blocked out of the computation. The velocities in these cells are set as Dirichlet boundary conditions determined by an interpolation algorithm from the neighbouring values within the flow field according to the following general stencil formulation:

$$\phi_0 = \left(\sum_{i=1}^{N} \alpha_i \cdot \phi_i\right) + \alpha_r \cdot \phi_r \tag{5}$$

where N is the number of neighbours involved in the interpolation. The situation in Figure 1 corresponds to the standard choice of boundary condition in a staggered grid. ϕ represents one of the velocity components. ϕ_0 is the internal Dirichlet boundary condition, ϕ_i the values at the neighbouring fluid points and ϕ_r the value at the physical boundary. The interpolation coefficients α_i and α_r solely depend on the geometry and are computed in a preprocessing step. They depend on the interpolation technique and are described in the following section. In the rest of this paragraph, we formally write boundary condition (5) as $u_0 = I(u)$.

We construct an iterative integration scheme to fulfil the following three conditions: (i) and (ii) are Equations (3) and (4) for the non-blocked cells and velocities and (iii) is the immersed boundary condition $u_0 = I(u)$ with desired interpolation accuracy order. This is accomplished by the iterative modified fractional step algorithm as given in Table I. We obtain an intermediate velocity field $\tilde{u}$ by using the pressure at the old time level p^n. The divergence of the velocity field is computed with boundary condition $\tilde{u}_0 = I(\tilde{u})$ (Step 2) and used as right-hand side of the Poisson equation (Step 3) for the pressure correction (Step 4). Within the Poisson equation, a Neumann condition for $\delta\tilde{p}$ is applied. After several iterations of Step 3 by a SIP solver, the velocity and pressure fields are updated by Step 4. After updating the boundary condition (Step 5), the divergence is checked. If it is below our desired threshold ε, the new velocity and pressure fields $u^{n+1} = \tilde{u}$ and $p^{n+1} = \tilde{p}$ are obtained. Else, Steps 3–5 are repeated until convergence.

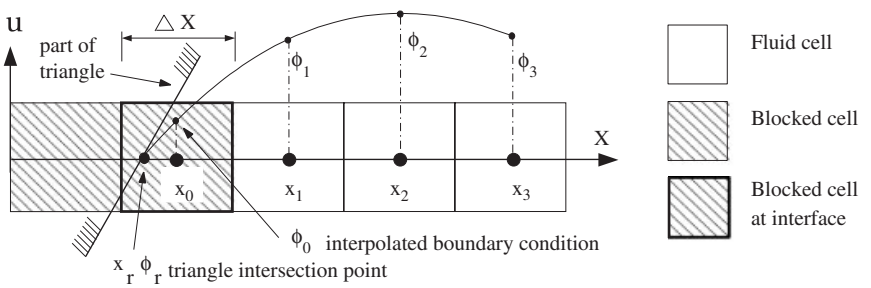

Figure 1. Typical 1D stencil configuration for interpolation in x direction. Blocked cells inside the body and neighbouring fluid cells are shown.

Table I. Time step algorithm.

Step 1:	$\tilde{u} = u^{n-1} + 2\delta t(D(u^{n-1}) + C(u^n) + G(p^n))$
Step 2:	$\tilde{u}_0 = I(\tilde{u});\ \tilde{p} = p^n$
while $(\mathrm{div}(\tilde{u}) > \varepsilon)$	
Step 3:	$\mathrm{div}[G(\delta\tilde{p})] = -\dfrac{1}{2\delta t}\mathrm{div}(\tilde{u})$
Step 4:	$\tilde{u} = \tilde{u} + 2\delta t G(\delta\tilde{p});\quad \tilde{p} = \tilde{p} + \delta\tilde{p}$
Step 5:	$\tilde{u}_0 = I(\tilde{u})$
end	
Step 6:	$u^{n+1} = \tilde{u};\ u_0^{n+1} = \tilde{u}_0;\ p^{n+1} = \tilde{p}$

2.3. Cartesian blocking algorithm

Our Cartesian mesh consists of control volumes for the pressure (pressure cells) and velocities (velocity cells), respectively. Due to the staggered arrangement of the variables their boundaries do not coincide. A cell is called blocked when its corresponding variable (pressure or velocity) is determined by interpolation (5). We block cells according to the criteria described below.

The blocking strategy is pressure cell oriented. Starting from the cell centres, intersection points of the coordinate lines with the body surface are searched. If intersection points are found and lie within the boundaries of the pressure cell, this cell is blocked. Starting from the pressure cell the blocking is extended to the velocity cells: each velocity cell touched by the blocked pressure cell is also blocked.

Principally, the body surface can either be represented analytically or by an unstructured mesh consisting of triangles. We prefer the latter, because it offers great flexibility in importing arbitrary surfaces from an originally CAD-based description. One has to bear in mind, however, that this approach leads to a second-order determination of intersection points and consequently to a second-order representation of the whole geometry. Therefore the surface mesh has to be fine enough not to introduce unnecessary errors in the geometric representation when considering flows over analytically given geometries as e.g. cylinders.

As mentioned before, the primary criteria for blocking a pressure cell is based upon the intersection of the body surface with the coordinate lines. In a second sweep, all remaining pressure cells within the body are blocked by a filling algorithm. A third sweep blocks all remaining velocity cells which overlap with blocked pressure cells in order to achieve the configuration given in Figure 1.

The filling algorithm starts at an arbitrary cell known to lie within the fluid domain. All cells not belonging to the set of fluid-filled cells are either blocked or marked as inactive. These cells are excluded from certain steps in the algorithm as e.g. determining convergence criteria.

3. INTERPOLATION

The following section describes the determination of the interpolation coefficients α_i required in Equation (5). The interpolations are derived by using 1D approximations. The generalization to the three-dimensional situation is done by weighting coefficients depending on intersection distances as described in Section 3.3. Before that, two alternative interpolation strategies, Lagrange and least squares interpolation, respectively, are discussed.

The situation in 1D is sketched in Figure 1. x_0 is the position of the variable to be interpolated. In the following, we set $x_0 = 0$ for simplicity. x_r is the intersection point of the coordinate line with the wall. It can run in the interval $[-0.5\Delta x; 0.5\Delta x]$, whereby Δx is the grid spacing. The value ϕ_r at x_r is set to the velocity of the wall. x_1, x_2, x_3 are the positions of the neighbouring variables ϕ_1, ϕ_2, ϕ_3 which are known variables and used for the interpolation formulas.

3.1. Lagrange interpolation

The Lagrange interpolation consists in fitting a polynomial through the known fluid points and the wall. The coefficients α_i in Equation (5) are directly obtained for the 1D interpolation

by the Lagrange formula:

$$\alpha_i = \left(\prod_{j=1, j\neq i}^{N} \frac{(x_0 - x_j)}{(x_i - x_j)} \right) \frac{x_0 - x_r}{x_i - x_r} \tag{6}$$

$$\alpha_r = \left(\prod_{j=1}^{N} \frac{(x_0 - x_j)}{(x_r - x_j)} \right) \tag{7}$$

The degree of the polynomial is equal to the number N of fluid points used in addition to the wall intersection point. A third-order polynomial e.g. requires three fluid points x_i plus the wall point x_r. The accuracy of the interpolation is of order $O(\Delta x^{N+1})$, since the order of accuracy is one order higher than the degree of the interpolation polynomial used.

3.2. Least squares interpolation

For the least squares interpolation, a polynomial is chosen so that the sum of the squares of the distance from the polynomial to the values at the fluid points is minimal. In addition, the polynomial is constricted to match the value ϕ_r at the wall directly. In the least squares method the degree of the polynomial has to be smaller than (N). In the following, we show how the coefficients in Equation (8) are derived for a polynomial of degree 2. The value of the polynomial $p(x)$ is determined by

$$p(x) = a_0 + a_1 x + a_2 x^2 \tag{8}$$

The constraint for the wall value to match exactly ϕ_r reads:

$$p(x_r) = a_0 + a_1 x_r + a_2 x_r^2 = \phi_r \tag{9}$$

which leads to the following expression for the polynomial:

$$p(x) = \phi_r + a_1(x - x_r) + a_2(x^2 - x_r^2) \tag{10}$$

We seek the minimum of the squares of the errors:

$$\min[F(a_1, a_2)] = \min \left[\sum_{i=1}^{n} (p(x_i) - \phi_i)^2 \right] \tag{11}$$

Differentiation with respect to a_1 and a_2 leads to two equations for them

$$\begin{aligned} a_1 &= \frac{C_2 A_2 - C_1 A_4}{A_2 A_3 - A_1 A_4} \\ a_2 &= \frac{C_1 A_3 - C_2 A_1}{A_2 A_3 - A_1 A_4} \end{aligned} \tag{12}$$

$$A_1 = \sum_{i=1}^{n} (x_i - x_r)^2, \quad A_2 = \sum_{i=1}^{n} (x_i^2 - x_r^2)(x_i - x_r)$$

$$A_3 = A_2, \quad A_4 = \sum_{i=1}^{n} (x_i^2 - x_r^2)^2$$

$$C_1 = \sum_{i=1}^{n} (\phi_i - \phi_r)(x_i - x_r), \quad C_2 = \sum_{i=1}^{n} (\phi_i - \phi_r)(x_i^2 - x_r^2)$$

The sums A_i are only dependent on the geometry and remain constant. The sums C_1 and C_2 however are dependent on the values ϕ_i at the control points. After inserting the polynomial coefficients a_1 and a_2 into the polynomial expression and rearranging, we obtain an expression for ϕ_0

$$\phi_0 = p(x_0) = \alpha_r \phi_r + \alpha_1 \phi_1 + \alpha_2 \phi_2 + \cdots + \alpha_n \phi_n \tag{13}$$

The coefficients α_r and α_i are obtained as

$$\alpha_r = 1 + \frac{(-A_2 V_2 + A_4 V_1)(x_0 - x_r) + (-A_3 V_1 + A_1 V_2)(x_0^2 - x_r^2)}{A_2 A_3 - A_1 A_4} \tag{14}$$

$$\alpha_i = \frac{A_2(x_i^2 - x_r^2) - A_4(x_i - x_r)}{A_2 A_3 - A_1 A_4}(x_0 - x_r) + \frac{A_3(x_i - x_r) - A_1(x_i^2 - x_r^2)}{A_2 A_3 - A_1 A_4}(x_0^2 - x_r^2) \tag{15}$$

with

$$V_1 = \sum_{i=1}^{n} (x_i - x_r), \quad V_2 = \sum_{i=1}^{n} (x_i^2 - x_r^2) \tag{16}$$

All coefficients α_r and α_i can be computed in a preprocessing step, because they are solely dependent on the geometry.

3.3. From 1D to 3D interpolation

One-dimensional interpolations have been introduced in the preceding sections. In the general case, the interpolation of the interface cell values will be possible in two or three dimensions and depends on the slope of the geometry. Figure 2 shows a possible 2D situation.

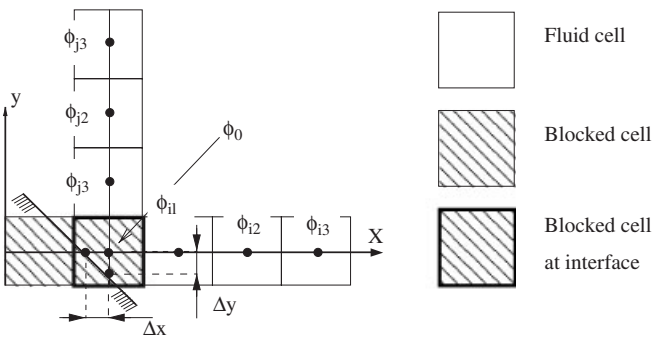

Figure 2. Two-dimensional stencil configuration.

The stencil in each direction is one-dimensional and the interpolation procedure (5) can be applied. In order to account for the three dimensionality, a weighting of the individual interpolation directions takes place. It must assure that the interpolated value equals the exact value when the boundary condition ϕ_r reaches a grid position. This can be achieved in the following way. The weighting factors γ_{dir} are defined by the equations:

$$\gamma_{\text{dir}} = \frac{\beta_{\text{dir}}}{\sum_{k=1}^{3} \beta_k} \tag{17}$$

with

$$\beta_{\text{dir}} = \frac{\prod_{k=1, k\neq \text{dir}}^{3} \Delta_k}{\Delta_{\text{dir}}} \tag{18}$$

The distance Δk is the distance between the intersection point and the interface cell centre in the corresponding direction. The sum of the γ_{dir} equals unity. This finally leads to a sum of interpolated values from all three directions.

$$\phi_{\text{mean}} = \sum_{\text{dir}=1}^{3} \gamma_{\text{dir}} * \phi_{\text{dir}} \tag{19}$$

4. STABILITY ANALYSIS

Stability is a crucial issue of the IBM. In order to gain insight into the stability properties of the various interpolation types, we perform a stability analysis. As noted by Carpenter *et al.* [21], the stability of a boundary scheme cannot be examined separately from the inner scheme. We follow their analysis and investigate boundary schemes in combination with the inner spatial approximation using matrix stability analysis.

Since the complete Navier–Stokes equations are too complex to be analysed analytically, we consider the linear convection equation in order to model the hyperbolic part of the Navier–Stokes equations:

$$\frac{\partial \phi}{\partial t} = c \frac{\partial \phi}{\partial x} \tag{20}$$

According to Gottlieb *et al.* [22], the stability of the scalar equation also yields the stability of the system of equations. Therefore it is sufficient to analyse the scalar equation (20). The convection speed c will be set to $c = -1.0$ in the following analysis.

For any explicit spatial scheme the semi-discretized equation is then

$$\frac{\partial \phi^+}{\partial t} = \frac{c}{\Delta x} \cdot M^+ \phi^+ \tag{21}$$

where Δx is the (constant) grid spacing. The stability properties of the scheme are determined by the properties of matrix M^+. When using a spatial second-order discretization with upwind boundary stencil at left border and artificial boundary closure at right border

[21, Equation (76)], the system matrix and solution vector read

$$M^+ = \begin{bmatrix} -1.0 & 1.0 & & & & & & \\ -0.5 & 0 & 0.5 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -0.5 & 0 & 0.5 & \\ & & & & & -0.5 & 0.5 & \end{bmatrix}, \quad \phi^+ = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_N \end{bmatrix} \tag{22}$$

The stability could be influenced by the choice of condition at the right boundary. We have chosen the same artificial boundary condition as introduced by Carpenter *et al.* [21, Equation (76)] because of two reasons: (i) In case of the one-dimensional convection equation there is only one physical boundary condition. In order to assess sufficient conditions for the stability of the scheme, the other boundary should not inhibit instabilities. (ii) Using an upwind stencil at the right boundary is not a proper choice as it could be too dissipative. The stability analysis would then be less sensitive to the left boundary closure under consideration.

At the left border, the IBM is applied: the immersed boundary stencil is used in order to interpolate ϕ_0 with the information at the boundary location x_r and at the next grid points within the fluid domain. When setting $\phi_r = 0$ in Equation (3), this can be expressed as follows:

$$\phi_0 = \gamma_1 \cdot \phi_1 + \gamma_2 \cdot \phi_2 + \gamma_3 \cdot \phi_3 \tag{23}$$

Here, we have restricted ourselves to three fluid control points. The coefficients $\gamma_{1\ldots3}$ are set according to Lagrange or least squares interpolation. With three control points, a polynomial of maximum degree three can be described by Lagrange interpolation, giving rise to a fourth-order interpolation. The least squares polynomial is restricted to degree two, which gives an accuracy of third order. Also note that coefficients $\gamma_{1\ldots3}$ depend on the location of the actual boundary. Therefore the stability analysis must be carried out for several boundary locations.

Substituting ϕ_0 in Equations (21) and (22) by Equation (23) imposes the left boundary condition. Note that only (N) points are strictly numerically motivated [21]. The physical boundary condition is a prescribed value and will not be considered in the stability analysis. Therefore the solution vector ϕ is reduced by the element ϕ_0. Consequently, the matrix M^+ is also reduced by the first line and first column in order to remain consistent. In the following, Equation (24) shall denote the reduced system

$$\frac{\partial \phi}{\partial t} = \frac{c}{\Delta x} \cdot M\phi = \hat{M}\phi \tag{24}$$

with

$$M = \begin{bmatrix} [-0.5\cdot\gamma_1] & [0.5\cdot(1-\gamma_2)] & [-0.5\cdot\gamma_3] & & & & \\ -0.5 & 0 & 0.5 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -0.5 & 0 & 0.5 & \\ & & & & -0.5 & 0.5 & \end{bmatrix} \tag{25}$$

and

$$\phi = [\phi_1 \quad \cdots \quad \phi_N]^{\mathrm{T}} \tag{26}$$

A similarity transformation can be applied to Equation (24) by using the eigenvalue matrix (diagonal matrix S) and eigenvectors (P) of the matrix $\hat{M}$:

$$P^{-1}\hat{M}P = S, \quad P^{-1}\phi = \tilde{\phi} \tag{27}$$

which yields

$$\frac{\partial\tilde{\phi}}{\partial t} = S\tilde{\phi} \tag{28}$$

The solution of the transformed equation finally reads:

$$\tilde{\phi}(x,t) = \mathrm{e}^{(St)}\cdot\tilde{f}(x) \tag{29}$$

It shows exponential dependence [21] on the eigenvalues of the matrix $\hat{M}$, i.e. the temporal stability of the numerical scheme is determined by the properties of the eigenvalues $\lambda_{1\ldots N}$ in matrix S. Spatial discretizations yielding eigenvalues with a positive real part will exhibit exponential divergence of the solution. In the following, we analyse the eigenvalues for Lagrange and least squares boundary conditions.

We computed the eigenvalues for Lagrange interpolation of second to fourth order as well as for least squares interpolation of third-order accuracy in connection with a standard second-order scheme in the domain. The boundary location is varied from $-0.5\Delta x$ to $+0.5\Delta x$ with the origin of the coordinate system at x_0 (Figure 1). A value of $|x_r| = 0.5\Delta x$ corresponds to a position in the middle of two cells. In Figure 3, the eigenvalues of all investigated schemes are plotted for $x_r = 0.0$. It is clear that the eigenvalues of the various schemes have to fall on one curve, since with $x_r = 0.0$, the interpolated value is set directly to the value of the boundary condition (in our case zero). This situation is stable.

When moving the boundary position towards positive x_r, we can investigate the stability for extrapolation of ϕ_0. At $x_r = 0.2\Delta x$ (Figure 4, left), the real parts of the eigenvalues move away from the imaginary axis and all eigenvalues are still within the left half plane (LHP). This trend continues for $x_r = 0.5\Delta x$ (Figure 4, right), from which we can conclude that, for extrapolation, all investigated schemes remain stable.

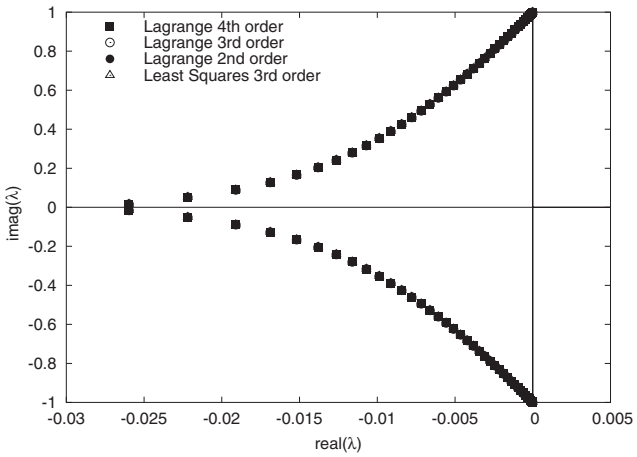

Figure 3. Eigenvalues of matrix M for $x_r = 0.0$

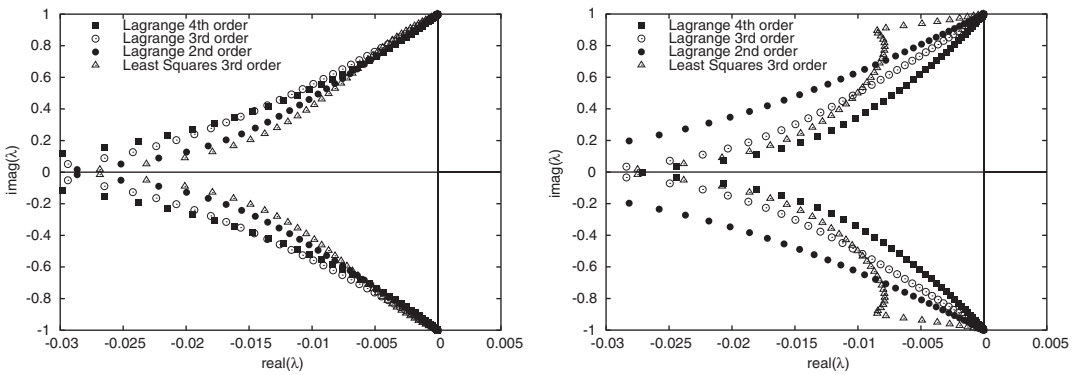

Figure 4. Eigenvalues of matrix M for $x_r = 0.2\Delta x$ (left) and $x_r = 0.5\Delta x$ (right).

For interpolation of ϕ_0, however, the various schemes behave differently. At position $x_r = -0.2\Delta x$ (Figure 5, left), it can be seen that all schemes tend towards the imaginary axis. The first boundary scheme with positive real eigenvalues is the fourth-order Lagrange interpolation which is slightly unstable for this boundary position. The third-order Lagrange interpolation as well as the linear (second-order) Lagrange interpolation and the least squares interpolation still have all eigenvalues in the LHP and are consequently stable.

At the location $x_r = -0.5\Delta x$, however, (Figure 5, right) the fourth order as well as the third-order Lagrange interpolations have eigenvalues with positive real part. This means instability for Lagrange interpolations of order higher than two. The least squares interpolation of third order still behaves well and only shows eigenvalues in the LHP.

We can conclude that with the least squares interpolation method, stable interpolation schemes for immersed boundaries can be constructed at higher accuracy orders than with Lagrange interpolation polynomials.

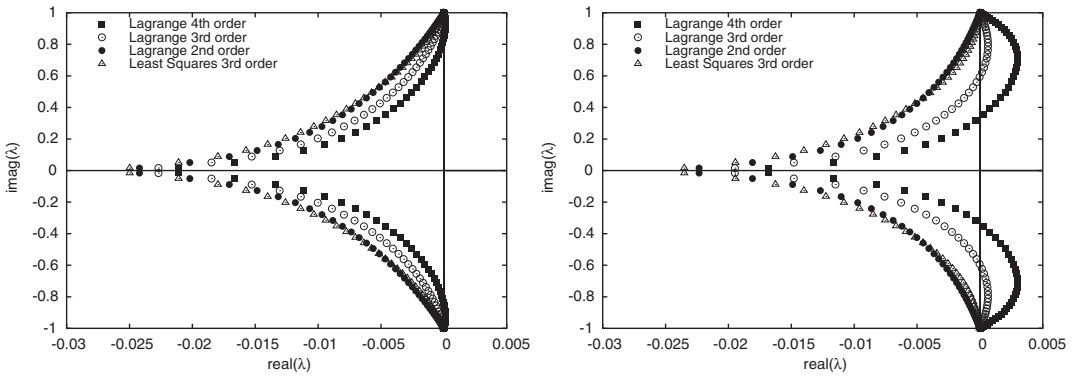

Figure 5. Eigenvalues of matrix M for $x_r = -0.2\Delta x$ (left) and $x_r = -0.5\Delta x$ (right).

5. ACCURACY INVESTIGATION

In what follows, we will investigate the effect of the boundary treatment on the numerical accuracy. In order to do so, we consider the cylindrical Taylor–Couette flow. We represent the cylinders in a Cartesian mesh applying the IBM and compare to the analytical solution. We investigate Lagrange interpolations of second and fourth order in comparison with the least squares interpolation of third-order accuracy. Note that for Lagrange third and fourth order, our simplified stability analysis predicts unstable behaviour. This does not necessarily mean that in actual cases, this instability is observed as already documented by Tremblay *et al.* [12, 13].

5.1. Testcase set-up

The cylindrical Taylor–Couette flow establishes between an inner and an outer cylinder, rotating at (possibly different) constant angular speed. A top view is shown in Figure 6. For the numerical investigations, the outer cylinder is maintained at rest while the inner cylinder rotates with unit angular speed. The outer cylinder radius is six times as big as the inner cylinder. The Reynolds number built with angular speed of the inner cylinder and radius of the inner cylinder is 30. Note that the simulation is strictly plane, which inhibits the appearance of three-dimensional Taylor vortices. Thus the flow is forced to remain laminar and an analytical solution can be obtained.

In cylindrical coordinates, the flow is one dimensional and fully described by its azimuthal velocity u_φ and pressure distribution p. Both variables solely depend on the radius r. u_φ satisfies the no-slip boundary condition at inner and outer cylinders. The one-dimensional analytical solution is written as (see e.g. Reference [23]):

$$u_\varphi(r) = \frac{A}{r} + Br \tag{30}$$

$$\frac{p(r)}{\rho} = \frac{-A^2}{2}\frac{1}{r^2} + 2AB\log r + \frac{B^2}{2}r^2 + D \tag{31}$$

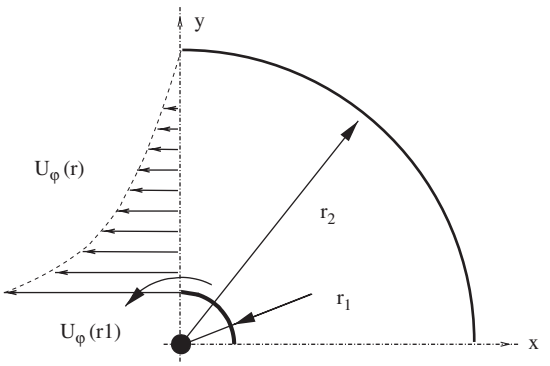

Figure 6. Top view of the cylindrical Taylor–Couette flow. For visibility only one-quarter of the cylinder is shown.

Table II. Grid dimensions.

	#1	#2	#3	#4
$\Delta x/r_1$	0.2	0.1	0.05	0.025
Total number of cells	6400	25 600	102 400	409 600
Number of cells per inner radius	5	10	20	40

with

$$A = \frac{-U_\varphi(r_1) \cdot r_1 r_2^2}{(r_1^2 - r_2^2)}, \quad B = \frac{U_\varphi(r_1) \cdot r_1}{(r_1^2 - r_2^2)} \tag{32}$$

The constant D in the equation for the pressure is chosen in such a way that the pressure at the outer cylinder is zero $p(r_2) = 0$. Note that the analytical solution is independent of Reynolds or Taylor number. We solve the flow in a 2D Cartesian coordinate system spanning the plane perpendicular to the axis.

5.2. *Numerical solution*

The two cylinders are immersed in a 2D Cartesian equidistant grid of size $16r_1 \times 16r_1$. For all tested interpolation algorithms, four runs with different grid spacings are investigated as displayed in Table II.

Figure 7 shows the azimuthal velocity and pressure profiles of the numerical simulation plotted along with the analytical profile for the grid spacings, #1 and #2, as given in Table II. Both use the least squares interpolation for the representation of the solid wall. Grid #1 is the coarsest grid used for the Couette-flow simulations. Only minor deviations from the analytical solution can be recognized. For the finer grid #2, no deviations can be distinguished anymore.

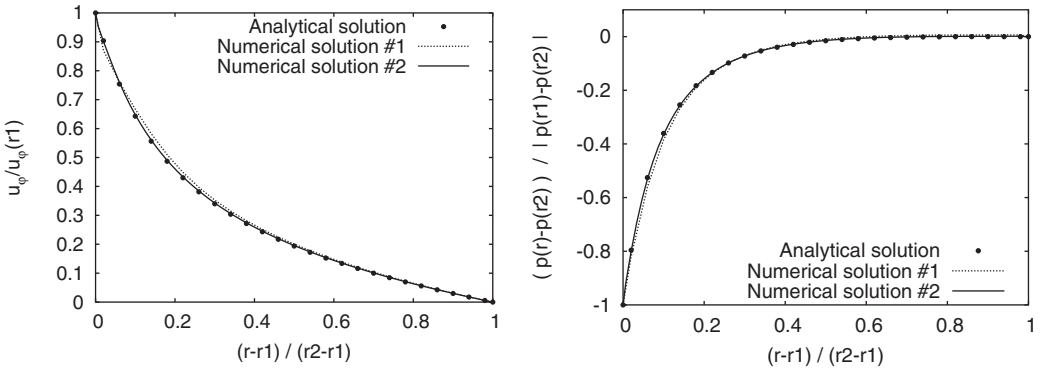

Figure 7. Profiles for least squares boundary condition for grid #1 and #2.

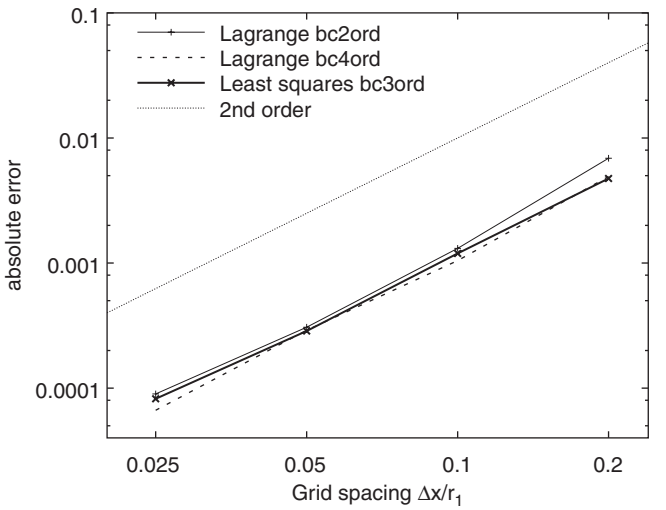

Figure 8. Absolute error of tangential velocity.

5.3. *Grid study*

The numerical error is computed by subtracting the analytical solution from the numerical solution at $(x, y) = (0, 4r_1)$ with the rotation axis of the cylinder at $(x, y) = (0, 0)$. In Figure 8, the error is plotted versus the grid spacing for all three interpolation algorithms in comparison to the second-order slope expected for the solver in the core of the domain. All three interpolation algorithms show similar convergence behaviour. The only difference is in the absolute value of the error. The fourth-order Lagrange interpolation predicts the boundary value a little bit more accurately than the third-order least squares interpolation which in turn is a little more accurate than the second-order Lagrange interpolation. It turns out that the higher stability of the least squares compared to the Lagrange interpolation is obtained without loss of accuracy.

6. APPLICATION TO LARGE-EDDY SIMULATION OF GEOMETRICALLY COMPLEX FLOWS

The practical advantages of the IBM with respect to boundary-fitted methods, among others easy meshing, increase with geometrical complexity. This paragraph provides a brief illustration of using the IBM to compute the flow in geometries of high practical relevance. Lagrange and least squares interpolations are compared.

We consider a car air-conditioning jet exiting into the passenger compartment. The nozzle is a standard industrial exhaust pipe provided by Audi A.G. It is fitted with two sets of small rectangular blades meant to deflect the flow (Figure 9). We perform an LES of the flow using a constant coefficient Smagorinsky model. In the present simulation, no near-wall (Van-Driest like) damping of the subgrid viscosity is used, because the grid is too coarse to resolve the buffer regions. A suitable wall model would be desirable. However, for such a complex flow situation the available models based on the logarithmic law or on thin boundary layer approximations are questionable. So, near wall errors are inevitable, but since the flow and separations are mainly driven by pressure gradients, we believe that the errors introduced are tolerable.

A full analysis of the flow can be found in Reference [24]. Here, we focus on computational aspects of the study and particularly on the immersed boundary treatment. We consider results obtained on a grid totalizing 40 millions points, which corresponds to 12 points in the channel space between two horizontal deflection blades. Although the total number of grid points is large, the interchannel resolution remains somewhat coarse. High-order stable IBM interpolations are thus particularly desirable for such a computation.

Both Lagrange and least squares IBM interpolations have been tested. The flow could be successfully computed with least squares quadratic (third-order) IBM interpolation. On the other hand, all non-trivial Lagrangian interpolations (order $\geqslant 2$) proved unstable for this flow. With Lagrange interpolations, we were only able to compute the flow in a long-term stable way with first-order interpolation. This is a simple displacement of the boundary condition resulting in a smeared stepwise body because of the staggered variable arrangement. The third-order

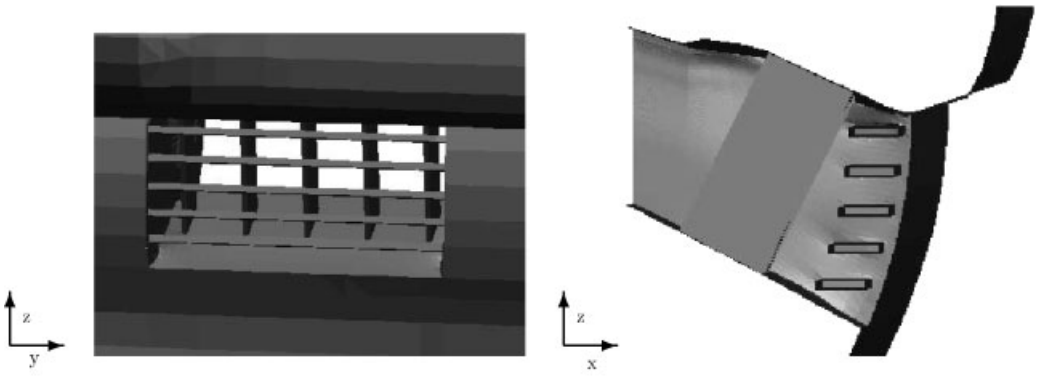

Figure 9. Front view (left) and side view (right) of the rectangular exhaust pipe. Two sets of rectangular thin blades are fitted at high angle of attack within the pipe.

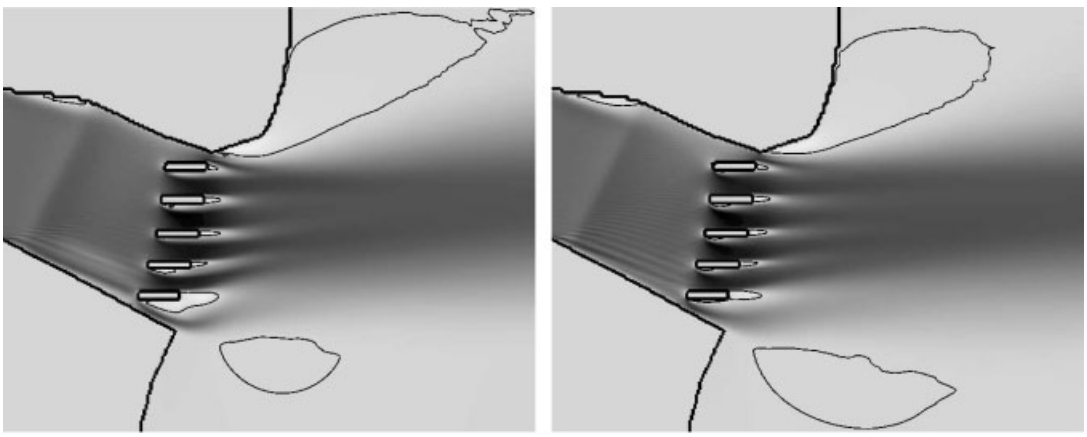

Figure 10. Time-averaged streamwise velocity in a (x,z) plane close to the middle spanwise position of the exhaust pipe. Thick black lines indicate the geometry of the nozzle, thin black lines are $u = 0$. Left: smooth body (least squares third order). Right: stepwise body.

least squares interpolation however allows for a smooth, accurate and stable representation of the wall.

Figure 10 shows time-averaged flows obtained with the smooth body (left) and the stepwise body (right). The black lines indicate the limits of recirculation zones. Both boundary treatments lead to similar qualitative features in the time-averaged flows. The flow is dominated by strong time-pulsating recirculation zones, particularly on the leeward side of blades. The jet spreads in both cases with nearly the same opening angle (4.8° for the third-order IBM versus 4.6° for the stepwise body) while it is deflected upwards. For the third-order interpolation, the deflection angle is 7.4°, in good agreement with experimental flow visualizations. For the stepwise body the deflection angle is smaller (6.6°). This comes from the under-prediction of the recirculation regions by the stepwise body representation. This result might be linked with a remark of Tseng and Ferziger [16] who found that the flow behind a Gaussian bump did not converge to the proper solution if a stepwise representation of the body was used. They attribute this effect to fine-scale noise introduced by the artificial roughness of the steps. We observe the same in our application where the stepwise representation leads to strong spurious oscillations with a wave length equal to the grid spacing. On the other hand, the third-order interpolation allows to get smooth unsteady vortical structures, convected with local flow velocity [24]. This feature illustrates the need for high-order boundary treatment when computing the flow around complex geometries with the IBM.

7. CONCLUSIONS

High-order IBM for simulating complex flows have been presented. The boundary treatment of the embedded geometry has been analysed for least squares and Lagrange interpolation schemes. Special attention has been drawn towards the least squares interpolation. It provides a high-order boundary treatment in combination with stable numerical properties.

The advantage of the IBM lies within the arbitrariness of geometrical complexity. Though being restricted to reasonable mesh resolution at the body surface, the mesh is completely independent of the geometry. Cumbersome mesh generation as in body-fitted grids is circumvented. The IBM is thus a practical method to compute utterly complex geometries as can be found in industrial applications.

The immersed boundary stencil formulation is created automatically in a preprocessing step and limits the numerical effort and cost to a minimum. Nevertheless, when using the IBM, its efficiency is restricted by the number of cells which are blocked out of the computation. Though these cells do not contribute to the numerical solution, the algorithm has to treat them as if they were part of it. Therefore, the computational efficiency is dependent on the number of blocked cells. Depending on the grid resolution the laminar test case example of the Couette flow has a ratio of blocked cells to non-blocked cells between 30 and 60%. Nevertheless the laminar Couette flow could be computed on a normal PC. The grid for the turbulent exhaust jet has approximately 40 Million nodes and the ratio of blocked nodes to non-blocked nodes is approximately 30%. By this the computational time for one time step is 5 s on 4 nodes of the Hitachi SR8000, LRZ Munich. The additional effort for the interpolation remains below 10% of the overall computing time. Together, this gives an overhead of approximately 40%, due to the IBM for this particular case.

In addition to the applicability of efficient numerical algorithms, investigations on the accuracy of the IBM have proven that the method preserves the accuracy of the numerical solver. Both least squares and Lagrange interpolations follow the same convergence rate. On the other hand, only least squares interpolation gives overall good results regarding the stability analysis. Thus least squares interpolation should be preferred for the computation of geometrically complex flows.

On the example of the turbulent exhaust jet, the applicability of the high-order method is demonstrated. In this case, none of the Lagrange interpolations, except the first order, gave stable solutions. We therefore used the least squares interpolation. It has the advantage to level out the strong near wall fluctuations occurring in this flow when the grid resolution, especially close to the wall, is limited. The urge for higher order representation of the geometry is documented by the fact that, with first-order interpolation, none of the fine-scale turbulent structures could be resolved and the solution was contaminated by unphysical oscillations. Thus, the least squares method renders a useful improvement with respect to Lagrange interpolation, because it extends the range of high-order boundary treatment considerably.

ACKNOWLEDGEMENTS

This work was supported by Publishing Arts Research Council through Grant No. 98–1846389.

REFERENCES

1. Verzicco R, Mohd-Yusof J, Orlandi P, Harworth D. Les in complex geometries using boundary body forces. *Proceedings of the Summer Program*, Center for Turbulence Research, 1998; 171–186.
2. Iaccarino G, Verzicco R. Immersed boundary technique for turbulent flow simulations. *Applied Mechanics Review* 2003; **56**(3):331–347.
3. Mittal R, Iaccarino G. Immersed boundary methods. *Annual Review of Fluid Mechanics* 2005; **37**:239–261.
4. Peskin CS. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics* 1972; **10**:252–271.

5. Lai M-C, Peskin CS. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics* 2000; **160**:705–719.
6. Goldstein D, Handler R, Sirovich L. Modeling a no-slip boundary with an external force field. *Journal of Computational Physics* 1993; **105**:354–366.
7. Saiki EM, Biringen S. Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method. *Journal of Computational Physics* 1996; **123**:450–465.
8. Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics* 2000; **161**:35–60.
9. Mohd-Yusof J. Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries. In *Annual Research Briefs*. NASA Ames Research Center/Stanford University Center of Turbulence Research: Stanford, 1997; 317–327.
10. Kim J, Kim D, Choi H. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics* 2001; **171**:132–150.
11. Balaras E. Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. *Computers and Fluids* 2004; **33**:375–404.
12. Tremblay F, Manhart M, Friedrich R. DNS and LES of flow around a circular cylinder at a subcritical Reynolds number with Cartesian grids. In *LES of Complex Transitional and Turbulent Flows*, Friedrich R, Rodi W (eds). Kluwer Academic Publishers: Dordrecht, 2001; 133–150.
13. Tremblay F, Manhart M, Friedrich R. LES of flow around a circular cylinder at a high subcritical Reynolds number. In *Direct and Large-Eddy Simulation IV*, Geurts B, Friedrich R, Metais O (eds). Kluwer Academic Publishers: Dordrecht, 2001.
14. Tessicini F, Iaccarino G, Fatica M, Wang M, Verzicco R. Wall modelling for large-eddy simulation using an immersed boundary method. In *Annual Research Briefs*. NASA Ames Research Center/Stanford University Center of Turbulence Research: Stanford, 2002; 181–187.
15. Majumbar S, Iaccarino G, Durbin P. RANS solvers with adaptive structured boundary non-conforming grids. In *Annual Research Briefs*. NASA Ames Research Center/Stanford University Center of Turbulence Research: Stanford, 2001; 353–366.
16. Tseng Y-H, Ferziger JH. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics* 2003; **192**:593–623.
17. Manhart M, Tremblay F, Friedrich R. MGLET: a parallel code for efficient DNS and LES of complex geometries. In *Parallel Computational Fluid Dynamics 2000*, Jensen *et al.* (eds). Elsevier Science B.V.: Amsterdam, 2001; 449–456.
18. Manhart M. A zonal grid algorithm for DNS of turbulent boundary layers. *Computers and Fluids* 2004; **33**(3):435–461.
19. Schumann U. Linear stability of finite difference equations for three-dimensional flow problems. *Journal of Computational Physics* 1975; **18**:465–470.
20. Ferziger JH, Peric M. *Computational Methods for Fluid Dynamics*. Springer: Berlin, 1996.
21. Carpenter M, Gottlieb D, Abarbanel S. The stability of numerical boundary treatments for compact high-order finite-difference schemes. *Journal of Computational Physics* 1993; **108**:272–295.
22. Gottlieb D, Gunzburger M, Turkel E. On numerical boundary treatment of hyperbolic systems for finite difference and finite element methods. *SINUM*, *Journal of the Society for Industrial and Applied Mathematics on Numerical Analysis* 1982; **19**(4):671–682.
23. Tritton DJ. *Physical Fluid Dynamics*. Oxford University Press: Oxford, 1988.
24. Le Duc A, Peller N, Manhart M, Wachsmann E-P. Aerodynamics and acoustic sources of the exhaust jet in a car air-conditioning system. In *Proceedings of the ERCOFTAC International Symposium on Engineering Turbulence Modelling and Measurements*, *ETMM6*, Rodi W, Mulas M (eds). Elsevier: Amsterdam, 2005; 709–718.

