Comput Mech
DOI 10.1007/s00466-006-0093-2

ORIGINAL PAPER

Computation of free-surface flows and fluid–object interactions with the CIP method based on adaptive meshless soroban grids

Kenji Takizawa · Takashi Yabe · Yumiko Tsugawa · Tayfun E. Tezduyar · Hiroki Mizoe

Received: 2 March 2006 / Accepted: 28 May 2006

Abstract The CIP Method [J comput phys 61:261–268, 1985; J comput phys 70:355–372, 1987; Comput phys commun 66:219–232, 1991; J comput phys 169:556–593, 2001] and adaptive Soroban grid [J comput phys 194:57–77, 2004] are combined for computation of three-dimensional fluid–object and fluid–structure interactions, while maintaining high-order accuracy. For the robust computation of free-surface and multi-fluid flows, we adopt the CCUP method [Phys Soc Japan J 60:2105–2108, 1991]. In most of the earlier computations, the CCUP method was used with a staggered-grid approach. Here, because of the meshless nature of the Soroban grid, we use the CCUP method with a collocated-grid approach. We propose an algorithm that is stable, robust and accurate even with such collocated grids. By adopting the CIP interpolation, the accuracy is largely enhanced compared to linear interpolation. Although this grid system is unstructured, it still has a very simple data structure.

K. Takizawa
National Maritime Research Institute,
6-38-1, Shinkawa, Mitaka-shi,
Tokyo 181-0004, Japan

T. Yabe(✉) · Y. Tsugawa
Tokyo Institute of Technology,
O-okayama, Meguro, Tokyo 152-8552, Japan
e-mail: yabe@mech.titech.ac.jp

T. E. Tezduyar
Mechanical Engineering, Rice University,
MS 321, Houston, Texas 77005, USA

H. Mizoe
Tohoku Electric Power Company, Inc.,
Sendai 981-0952, Japan

1 Introduction

Computation of fluid–object and fluid–structure interactions poses a number of numerical challenges. These challenges include the accuracy of the flow field near solid surfaces, accurate advection of the vortex fields, grid generation and motion, and computational efficiency. The numerical challenges become even more formidable when the problem, in addition to fluid–object and fluid–structure interactions, involves free-surface or multi-fluid flows. While the boundary layers near solid surfaces need to be accurately resolved, free surfaces and two-fluid interfaces need to be calculated in a robust, accurate and efficient fashion.

A number of very effective finite element interface-tracking (mesh moving) methods have been developed in recent decades for computation of fluid–object and fluid–structure interactions (see for example the methods and computations reported in [7–16]). The Mixed Interface-Tracking/Interface-Capturing Technique (MITICT) was introduced in [17] for computations involving both fluid–solid interfaces that need to be accurately tracked with a moving mesh method and fluid–fluid interfaces that are too complex or unsteady to be tracked and therefore require an interface-capturing technique. As far as these finite element approaches are concerned, there is a clear need for more freedom from mesh moving and distortion issues, and there is also still some room for improvements in accuracy and efficiency. On the other hand, Adaptive Mesh Refinement (AMR) [18] with Cartesian grids can be accurate only when the interpolation inside a cell is effective, but the AMR cannot place grid points on the boundaries and resolve the thin boundary layers. The cut-cell [19] approach is an alternative, but its application to moving

objects is not so easy because of the presence of extremely small-size cells. Besides these approaches, the use of the particles [20,21] in capturing interfaces would be among the challenging and interesting future techniques.

The Constrained Interpolation Profile/Cubic Interpolated Pseudo-particle (CIP) technique, which was developed by Yabe et al. [1–4] for solving hyperbolic equations, has attracted a great deal of attention. This technique would become even more powerful and practical if it is combined with a special mesh system, "Soroban grid", that allows accurate computations with unstructured grids. It would be a robust, accurate and efficient way of computing fluid–object and fluid–structure interactions in the presence of free surfaces and two-fluid interfaces. It would accurately resolve the boundary layers near solid surfaces and also calculate, in a robust fashion, even the most complex and unsteady free surfaces and two-fluid interfaces. In this paper we propose such a combined CIP/Adaptive Soroban grid method and apply it to a number of problems in multi dimensions.

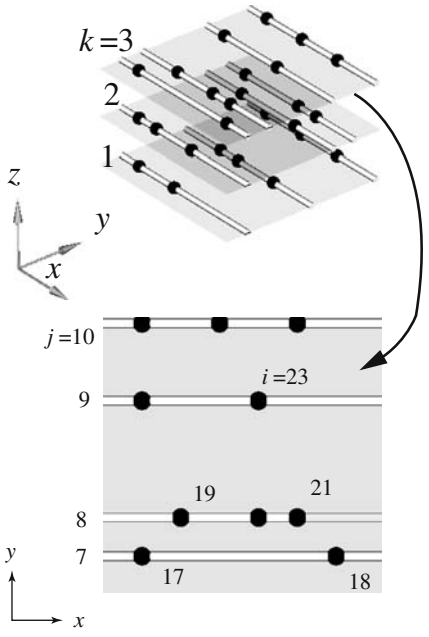

Fig. 1 (*Top*) Full view of a three dimensional Soroban grid. The black spheres represent the grid points. (*Bottom*) Line and grid arrangement on one of the planes. The symbols k, j, and i denote the indices for the planes, lines and grid points, respectively

2 Grid system

2.1 Structure of the Soroban grid system

The Soroban grid is composed of planes, lines and grid points as shown in Fig. 1. Parallel planes can freely move in the z direction. Each plane is composed of lines and grid points as shown in the bottom part of Fig. 1. The parallel lines can freely move in the y direction and the grid points on each line can move along each line which is directed in the x-direction. We note that the number of grid points on different lines are not necessarily equal. Therefore the number of grid points on each line can vary depending on the required resolution. Such flexibility of the Soroban grid could increase the possibility of grid arrangement. Most importantly, finding neighboring points becomes very easy and fast because of this structure, as will be shown in later sections.

Since the number of grid points may change on each line, it would be better to employ a one-dimensional array for storing the grid points and changing their number during a computation. For example, when the first line has 30 grid points and the second line has 20 grid points, we need 50 grid points. Even if both lines have 25 grid points in the next time step, the first and second lines can share the memory and we do not need memory re-allocation.

The index of the grid point number is unique for all the grid points on all the lines and planes. Therefore it starts from the first grid point along the first line on the first plane and increases along the first line, then continuously increases along the second line and so on. Similarly the index of the line number is also one-dimensional and starts from the first line on the first plane, and continuously increases to the next planes. In this way we can use only one-dimensional arrays and we can change the number of grid points on each line, number of lines on each plane, and the number of planes. The details of how the grid positions are specified are described in Appendix 6.1.

2.2 Grid generation

Using the Soroban grids, grid points can be placed adaptively around any complex shape, and can be changed each time step with little computational cost. Since we use a one-dimensional array for indexing, grid points are not inserted but all points are replaced. As shown in the next section, this will cause no problem for the calculations. We choose a simple method for the grid generation. This is an old-fashioned way, but it is easy to implement in multi dimensions. First, we explain the grid-generation method for a one-dimensional non-regular grid.

2.2.1 Non-regular grids in one dimension

We define a monitoring function $\mathrm{M}(x,t)$. This function should give the information on which part of the calculation domain should be refined. For this purpose, we propose a form:

$$\mathrm{M}(x,t) = \sqrt{1+\alpha\left(\frac{\partial\phi}{\partial x}\right)^2} + \beta\left|\frac{\partial^2\phi}{\partial x^2}\right|, \tag{1}$$

where α and β are some scaling coefficients, and ϕ represents the solution value, such as the density, velocity or pressure. Since the minimum of $\mathrm{M}(x,t)$ is 1, we define in advance the maximum of $\mathrm{M}(x,t), M_{\max}$, so that we can determine the grid interval ratio between the maximum and minimum intervals. Therefore the monitoring function is re-written as follows:

$$\mathrm{M}(x,t) = \min\left(\sqrt{1+\alpha\left(\frac{\partial\phi}{\partial x}\right)^2} + \beta\left|\frac{\partial^2\phi}{\partial x^2}\right|, M_{\max}\right). \tag{2}$$

Then we introduce the accumulated function I by integrating M as

$$\mathrm{I}(x,t) = \int_{x_S}^{x} \mathrm{M}(x,t)\mathrm{d}x, \tag{3}$$

where x_S is the start position of the system. If we divide the accumulated function into equal regions, the boundaries of each region give the grid points as shown in Fig. 2. Let L be the number of grid points. Then the position of each grid point is calculated as follows:

$$x_i^{n+1} = \mathrm{I}^{-1}\left(\frac{i\times \mathrm{I}(x_E)}{L^{n+1}-1}\right), \tag{4}$$

where x_E is the end position of the system and I^{-1} means the inverse function of I. Since I is a monotonically increasing function, the inverse of I can be easily estimated by linear interpolation as follows. First, we search i that satisfies $I_i \le I < I_{i+1}$, and then calculate as

$$x_i^{n+1} = \frac{\left(I_{i+1}^n - I\right)x_i^n + \left(I - I_i^n\right)x_{i+1}^n}{I_{i+1}^n - I_i^n}, \tag{5}$$

where I_i is the value of I at the old grid point x_i^n.

We allow the number of grid points to vary in time and hence L^{n+1} represents the number of the grid points at time step $n+1$. Although we can choose any L^{n+1}, we adopt the following rule.

$$L^{n+1} = \mathrm{int}\left(\frac{\mathrm{I}(x_E)}{\Delta x_{\max}}\right) + 1, \tag{6}$$

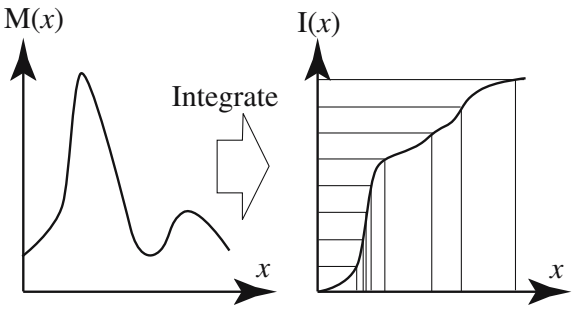

Fig. 2 (*Left*) Monitoring function. (*Right*) Accumulated function

where $\mathrm{int}(x)$ is the round-off function and $\Delta x_{\max}$ is the maximum grid interval. This means that all intervals become $\Delta x_{\max}$ when $M(x) = 1$ for all parts of calculation domain. Moreover, the minimum interval is $\Delta x_{\max}/M_{\max}$. In this way a non-regular grid is generated, then we apply it to the Soroban grid.

2.2.2 Application to the Soroban grid

Since the Soroban grid is composed of the grid points along straight lines, it is easy to apply the above method to the Soroban grids in multi-dimensions by directional splitting. In two dimensions, we rearrange the line location on a plane at first according to the above-mentioned method, then rearrange the grid location on a line with the same method. However, we have to change the definition of the monitoring function as follows:

$$\mathrm{M}(x,y,t) = \sqrt{1+\alpha\left(\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2\right)} + \beta\left(\left|\frac{\partial^2\phi}{\partial x^2}\right| + \left|\frac{\partial^2\phi}{\partial y^2}\right|\right). \tag{7}$$

In order to determine the line location in the y direction, we must define $\mathrm{M}^y(y,t)$ like in the one-dimensional case. We propose two methods for calculating $\mathrm{M}^y(y,t)$ from $\mathrm{M}(x,y,t)$. One is to use the maximum of the monitoring function along the line. It means,

$$\mathrm{M}^y(y,t)_{\max} = \max\left\{\mathrm{M}(x,y,t) | x_S \le x \le x_E\right\}. \tag{8}$$

Another approach is to average the monitoring function as follows:

$$\mathrm{M}^y(y,t)_{\mathrm{ave}} = \frac{1}{x_E - x_S}\int_{x_S}^{x_E} \mathrm{M}(x,y,t)\mathrm{d}x. \tag{9}$$

Then we use a combination of these two definitions:

$$\mathrm{M}^y(y,t) = q\mathrm{M}^y(y,t)_{\max} + (1-q)\mathrm{M}^y(y,t)_{\mathrm{ave}}. \tag{10}$$

Here $0 \le q \le 1$ is the weighting function.

As is easily understood, $\int_{x_S}^{x_E} \mathrm{M}(x,y,t)dx$ is a function of y, and therefore the number of grid points on different lines do not need to be equal. In the second step, the grid points along a line are rearranged along the line using $\mathrm{M^{jL}(x,t)}$, which should be $\mathrm{M}(x, y[jL], t), y[jL]$ being the line location. In three dimensions, we first rearrange the planes, then the lines and grid points, and the monitoring function includes the spatial derivative in the z direction.

3 Computational strategy

3.1 Discretization

In this section, we discretize the three-dimensional fluid dynamics equations:

$$\frac{\partial \rho}{\partial t} + (\mathbf{u}\cdot\nabla)\,\rho = -\rho\nabla\cdot\mathbf{u}, \tag{11}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\mathbf{u} + \mathbf{F}, \tag{12}$$

$$\frac{\partial p}{\partial t} + (\mathbf{u}\cdot\nabla)\,p = -\rho C_s^2\nabla\cdot\mathbf{u}, \tag{13}$$

where C_s is the local acoustic speed, which is $\sqrt{\gamma p/\rho}$ for ideal gases. The symbols $\rho, \mathbf{u}, p$ denote the density, velocity and pressure, μ is the viscosity, and $\mathbf{F}$ is the external force such as gravity. Equations (11)–(13) are put into a common form as

$$\frac{\partial \phi}{\partial t} + (\mathbf{u}\cdot\nabla)\,\phi = G, \tag{14}$$

where ϕ is $\rho, \mathbf{u}$ and p. Then we use **a** fractional step method as follows:

$$\frac{\phi^* - \phi^n}{\Delta t} + (\mathbf{u}\cdot\nabla)\,\phi = 0, \tag{15}$$

$$\frac{\phi^{n+1} - \phi^*}{\Delta t} = G. \tag{16}$$

Equations (15) and (16) show, in the fractional-step treatment of a generic form represented by Eq. (14), how the spatial operator is split into two components. How each component is handled is explained in the remainder of this section. The advection part Eq. (15) is solved by the method of characteristics based on the CIP method [1–4]:

$$\phi^* = \phi^n\left(\mathbf{x}^* - \int_{t^n}^{t^{n+1}} \mathbf{u}\mathrm{d}t\right). \tag{17}$$

In Eq. (17), the form $\phi^n(\mathbf{x}^* - \int_{t^n}^{t^{n+1}} \mathbf{u}\mathrm{d}t)$ represents evaluation of ϕ^n at $\mathbf{x}^* - \int_{t^n}^{t^{n+1}} \mathbf{u}\mathrm{d}t$. What is important in Eq. (17) is the definition of the values on the Soroban grids. The values denoted by the superscript n are defined on the Soroban grid at the time step n, but the values denoted by the superscript $n+1$ and $*$ are defined on a new Soroban grid at the next time step. This new Soroban grid can be generated without any relation to the grid at step n. In this way, we can dynamically change the grid to adapt to the solution.

Since ϕ^n is defined on the grid at step n, $\phi(\mathbf{x})$ in between the grid points can be calculated by the CIP interpolation from ϕ defined on the grid point $\mathbf{x}^n$. It is important to note that we move the grid at the time of the advection process. Therefore the new grid point $\mathbf{x}^*$ in the argument of Eq. (17) is not necessarily the same as $\mathbf{x}^n$, and $\mathbf{x}^* - \mathbf{u}^{n+1}(\mathbf{x}^*)\Delta t$ becomes the upstream point. The advection needs the interpolated value at the upstream point in between grid points, and the value at the new grid points is also given by the interpolation from the values at $\mathbf{x^n}$. Therefore, by this procedure, the interpolation for both processes is performed by only a single interpolation procedure. The CIP method shows excellent performance for the advection process for fixed grids. This means that the CIP interpolation gives very accurate prediction of the value in between grid points. For example, as shown in Fig. 17, we calculate the value by using the CIP1D function given in Appendix 6.2. The bottom values are estimated by

$$\phi_{\mathrm{BS}} = \mathrm{CIP1D}\left(\phi\left[i_{00}^G\right], \phi\left[i_{00}^G + 1\right], x_0 - x\left[i_{00}^G\right], x\left[i_{00}^G + 1\right] - x\left[i_{00}^G\right]\right), \tag{18}$$

$$\phi_{\mathrm{BN}} = \mathrm{CIP1D}\left(\phi\left[i_{10}^G\right], \phi\left[i_{10}^G + 1\right], x_0 - x\left[i_{10}^G\right], x\left[i_{10}^G + 1\right] - x\left[i_{10}^G\right]\right), \tag{19}$$

where BS and BN are positions with coordinates $(x_0, y[j^L], z[k^P])$ and $(x_0, y[j^L+1], z[k^P])$, respectively. Then,

$$\phi_{\mathrm{B}} = \mathrm{CIP1D}\left(\phi_{BS}, \phi_{BN}, y_0 - y[j^L], y[j^L+1] - y[j^L]\right), \tag{20}$$

where B is the position with coordinates $(x_0, y_0, z[k^P])$. Similarly, ϕ_{T}, with coordinates $(x_0, y_0, z[k^P+1])$, is calculated. Then,

$$\phi_0 = \mathrm{CIP1D}\left(\phi_B, \phi_T, z_0 - z[k^P], z[k^P+1] - z[k^P]\right). \tag{21}$$

In addition to ϕ^* in Eq. (17), we need the updated derivatives for the CIP. Thus we differentiate Eq. (15) in the x-direction.

$$\frac{\partial_x\phi^* - \partial_x\phi^n}{\Delta t} + (\mathbf{u}\cdot\nabla)\,\partial_x\phi = -\left(\frac{\partial}{\partial x}\mathbf{u}\cdot\nabla\right)\phi^n, \tag{22}$$

where $\partial_x\phi$ is an independent variable and is the spatial derivative in the x-direction.

Then, with the same idea used in Eq. (17), the derivatives are:

$$\partial_x\phi^* = \partial_x\phi^n\left(\mathbf{x}^* - \int_{t^n}^{t^{n+1}} \mathbf{u}\mathrm{d}t\right) - \int_{t^n}^{t^{n+1}} \left(\frac{\partial}{\partial x}\mathbf{u}\cdot\nabla\right)\phi^n \mathrm{d}t. \quad (23)$$

The second term on the right-hand-side is calculated by the finite differences.

The non-advection parts of Eqs. (11)–(13) are expressed as follows:

$$\frac{\rho^{n+1}-\rho^*}{\Delta t} = -\rho^*\nabla\cdot\mathbf{u}^{\mathbf{n+1}}, \quad (24)$$

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^*}{\Delta t} = -\frac{1}{\rho^*}\nabla p^{n+1} + \frac{\mu}{\rho^*}\nabla^2\mathbf{u} + \mathbf{F}, \quad (25)$$

$$\frac{p^{n+1}-p^*}{\Delta t} = -\rho^* C_s^2 \nabla\cdot\mathbf{u}^{n+1}. \quad (26)$$

We use the same idea as the CCUP method [6] and ICE [22] to solve the part related to the acoustic wave as follows. From Eq. (25),

$$\nabla\cdot\mathbf{u}^{n+1} = -\nabla\cdot\left(\frac{1}{\rho^*}\nabla p^{n+1}\right)\Delta t + \nabla\cdot\mathbf{u}^* + \nabla\cdot\left(\frac{\mu}{\rho^*}\nabla^2\mathbf{u} + \mathbf{F}\right)\Delta t, \quad (27)$$

$$\frac{p^{n+1}-p^*}{\rho^* C_s^2 \Delta t^2} = \nabla\cdot\left(\frac{1}{\rho^*}\nabla p^{n+1}\right) - \frac{\nabla\cdot\mathbf{u}^*}{\Delta t} - \nabla\cdot\left(\frac{\mu}{\rho^*}\nabla^2\mathbf{u} + \mathbf{F}\right). \quad (28)$$

In **a** previous paper [23], Eq. (28) was re-written like the SMAC method,

$$\mathbf{u}^{(*+1)} = \mathbf{u}^* - \left(\frac{\nabla p^*}{\rho^*} - \frac{\mu}{\rho^*}\nabla^2\mathbf{u}^{(*+\theta)} - \mathbf{F}^*\right)\Delta t, \quad (29)$$

where $0 < \theta \leq 1$, and if θ equals 0, it becomes an explicit method,

$$\frac{\Delta p}{\rho^* C_s^2 \Delta t^2} = \nabla\cdot\left(\frac{1}{\rho^*}\nabla\Delta p\right) - \frac{\nabla\cdot\mathbf{u}^{(*+1)}}{\Delta t}, \quad (30)$$

where $\mathbf{u}^{(*+1)}$ is the predicted velocity and Δp **is** $p^{n+1} - p^*$. After the pressure is calculated, the divergence of the velocity is calculated from Eq. (26) as follows:

$$\nabla\cdot\mathbf{u}^{n+1} = -\frac{\Delta p}{\rho^* C_s^2 \Delta t}. \quad (31)$$

Using this expression in Eq. (24), we estimate the density evolution. Then the velocity is accelerated and the pressure is updated as follows:

$$\mathbf{u}^{n+1} = \mathbf{u}^{(*+1)} - \frac{\nabla\Delta p}{\rho^*}\Delta t, \quad (32)$$

$$p^{n+1} = p^* + \Delta p. \quad (33)$$

In summary, we use Eq. (24) and Eqs. (29)–(33). When we deal with incompressible flows, we use Eqs. (29) (without p^*), (30) (with $C_s \to \infty$) and (32), and in those equations Δp is replaced with p^{n+1}.

For the time evolution of the spatial derivatives, we already treated the term related to advection in Eq. (23). The other terms are estimated by taking the spatial derivative of Eq. (16) in x, as

$$\frac{\partial_x\phi^{n+1} - \partial_x\phi^*}{\Delta t} = \frac{\partial G}{\partial x}. \quad (34)$$

Instead of using finite differences for $\partial G/\partial x$ on the right-hand-side, we use Eq. (16) for this calculation, because the effect of G is already included in the change of ϕ:

$$\frac{\partial G}{\partial x}\Delta t = \frac{\partial\phi^{n+1}}{\partial x} - \frac{\partial\phi^*}{\partial x}. \quad (35)$$

Thus Eq. (34) is written as

$$\partial_x\phi^{n+1} = \partial_x\phi^* + \frac{\partial}{\partial x}(\phi^{n+1} - \phi^*). \quad (36)$$

Although Eq. (36) looks like a tautology, the actual calculation is different. The symbol $\partial_x\phi$ denotes the independent variable but $\partial\phi/\partial x$ is estimated by the finite difference of ϕ [2]. All in all, the time evolution of $\partial_x\phi$ in the non-advection part is estimated by the time evolution of finite differences approximation of ϕ expected from the G term.

3.2 Finite differences on the Soroban grid

The spatial discretization is based on finite difference approximations on the Soroban grid. The preliminary concepts of the finite differences on the Soroban grid are given in Appendix 6.3.

3.2.1 Time evolution of the spatial derivatives

When we update the spatial derivatives according to Eq. (36), we already know $\phi^*, \partial_x\phi^*, \partial_y\phi^*$ and ϕ^{n+1} and therefore we obtain $\partial_x\phi^{n+1}$ and $\partial_y\phi^{n+1}$ by the following procedure.

(1) The calculations of $\partial\phi^{n+1}/\partial x$ and $\partial\phi^*/\partial x$ are done by just using finite differences along the line, which is the same as Eq. (90).
(2) In the calculation of $\partial\phi^{n+1}/\partial y$ and $\partial\phi^*/\partial y$, as explained before, there exist no corresponding points with the same x in the y-direction. We need to calculate these values as accurately as possible. Using ϕ and $\partial_x\phi$, the values at S and N shown in Fig. 19 are estimated by the CIP method.

If we use the CIP Type-C [24] method, we can calculate $\partial\phi_{xy}^{n+1}$ as the 3rd step. Since we already know the $\partial_y\phi^{n+1}$ and $\partial_y\phi^*$, we can apply the CIP method to the x-derivative of $\partial_y\phi$ instead of ϕ along the line to get $\partial_{xy}\phi$. See refs. [24] and [5] for detailed procedure.

3.2.2 Implicit solution of the equations

In Sect. 3.1, we omitted the details of how we treated the viscous term and **F**. These terms are

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\mu}{\rho}\nabla^2\mathbf{u} + \mathbf{F}. \tag{37}$$

Frequently, the viscous term needs to be treated implicitly. In such cases, the above equation is very similar to Eq. (30). These are put into a common form:

$$\frac{\Delta\phi}{\Delta t} = \kappa\nabla\cdot\left(H\nabla\left(\theta\Delta\phi + \phi^*\right)\right) + \Gamma, \tag{38}$$

where $\Delta\phi$ is $\phi^{n+1} - \phi^*$, and ϕ is $\mathbf{u}$ or p. Here $0 < \theta \leq 1$, and if θ equals 0, it becomes an explicit method. To solve the equation for $\theta \neq 0$, we have to solve the matrix equation in terms of $\Delta\phi$.

In a conventional method, we need to solve a matrix system only for $\Delta\phi$. In the proposed method in Sect. 3.2.1, however, we need the spatial derivatives of $\Delta\phi$ for interpolation. In order to avoid solving the augmented matrix both for ϕ and its derivatives, we introduce the following iteration process by separating Eq. (38) as follows:

$$\begin{aligned}\frac{\Delta\phi^{(m-1)} + \delta\phi^{(m)}}{\Delta t} &= \kappa\nabla\cdot\left(H\nabla\left(\theta\Delta\phi^{(m-1)} + \phi^*\right)\right)\\ &\quad+\kappa\nabla\cdot\left(H\nabla\left(\theta\delta\phi^{(m)}\right)\right) + \Gamma,\end{aligned} \tag{39}$$

where (m) denotes the number of iteration step and

$$\Delta\phi^{(m)} = \sum_{k=0}^{(m)}\delta\phi^k, \tag{40}$$

which is the predicted solution at the (m)-th iteration. The above equation is implicitly solved in terms of $\delta\phi^{(m)}$. Since $\partial_x\Delta\phi^{(m-1)}$,$\partial_y\Delta\phi^{(m-1)}$, $\partial_{xy}\Delta\phi^{(m-1)}$ are already known explicitly, $\kappa\nabla\cdot\left(H\nabla\left(\theta\Delta\phi^{(m-1)} + \phi^*\right)\right)$ is calculated by the finite difference with the values interpolated by the CIP method as shown in Sect. 3.2. In most of the examples in Sect. 4, the maximum m for convergence is about 3.

In contrast, $\delta\phi^{(m)}$ is solved by the linear interpolation. Suppose the $\nabla\cdot(H\nabla)$ operator is written as follows:

$$\nabla\cdot(H\nabla)\delta\phi^{(m)} = \sum\chi_l\delta\phi_l^{(m)}, \tag{41}$$

where l symbolically represents C or N or S and so on shown in Fig. 19, and χ_l are the coefficients stemming from finite difference approximation. In two dimensions, the number of points related to the grid point C are 6 and in three dimensions it is 14. In the x-direction, it is simply written as

$$\begin{aligned}\frac{\partial}{\partial x}\frac{H\partial\delta\phi(m)}{\partial x} &= \frac{1}{\Delta x}\left(\frac{H_{CCm}}{\Delta x_{CCm}}\left(\delta\phi_{Cm}^{(m)} - \delta\phi_C^{(m)}\right)\right.\\ &\quad\left.+\frac{H_{CCp}}{\Delta x_{CCp}}\left(\delta\phi_{Cp}^{(m)} - \delta\phi_C^{(m)}\right)\right),\end{aligned} \tag{42}$$

where Δx, Δx_{CCm} and Δx_{CCp} are the distance between the points Cm and Cp, Cm and C, and C and Cp in Fig. 19, respectively. H_{CCm} stands for the value of H in the middle of C and Cm. Then,

$$\begin{aligned}\frac{\partial}{\partial x}\frac{H\partial\delta\phi(m)}{\partial x} &= \chi_{Cm}^x\delta\phi_{Cm}^{(m)} + \chi_{Cp}^x\delta\phi_{Cp}^{(m)}\\ &\quad-\left(\chi_{Cm}^x + \chi_{Cp}^x\right)\delta\phi_C^{(m)},\end{aligned} \tag{43}$$

where,

$$\chi_{Cm}^x = \frac{H_{CCm}}{\Delta x\Delta x_{CCm}}, \tag{44}$$

$$\chi_{Cp}^x = \frac{H_{CCp}}{\Delta x\Delta x_{CCp}}. \tag{45}$$

Similarly, in the y direction,

$$\begin{aligned}\frac{\partial}{\partial y}\frac{H\partial\delta\phi(m)}{\partial y} &= \chi_S^y\delta\phi_S^{(m)} + \chi_N^y\delta\phi_N^{(m)}\\ &\quad-\left(\chi_S^y + \chi_N^y\right)\delta\phi_C^{(m)},\end{aligned} \tag{46}$$

$$\chi_S^y = \frac{H_{CS}}{\Delta y\Delta y_{CS}}, \tag{47}$$

$$\chi_N^y = \frac{H_{CN}}{\Delta y\Delta y_{CN}}, \tag{48}$$

where Δy, Δy_{CS} and Δy_{CN} are the distances between the points S and N, C and S, and C and N, respectively. H_{CN} is the value of H in the middle of C and N. Since we do not have grid points at N and S, we must estimate these values. If these values were estimated by the CIP method, derivatives would be required to be solved at the same time. Although it is possible, we avoid it for efficient computation by introducing the iteration process defined by Eq. (39). Therefore these values at this stage are estimated by the linear interpolation. It will be corrected later in the form of $\nabla(H\nabla(\Delta\phi^{(m-1)}))$ in Eq. (39).

In the linear interpolation, for example, $\delta\phi_S^{(m)}$ are estimated as follows:

$$\delta\phi_S^{(m)} = (1-\xi_S)\delta\phi_{Sm}^{(m)} + \xi_S\delta\phi_{Sp}^{(m)}, \tag{49}$$

where $\xi_S = (x_S - x_{Sm})/(x_{Sp} - x_{Sm})$. Similarly,

$$\delta\phi_N^{(m)} = (1-\xi_N)\delta\phi_{Nm}^{(m)} + \xi_N\delta\phi_{Np}^{(m)}, \tag{50}$$

where $\xi_N = (x_N - x_{Nm})/(x_{Np} - x_{Nm})$. These equations are substituted into Eq. (46). For the calculation of $\nabla \cdot (H\nabla(\theta\Delta\phi^{(m-1)} + \phi^*))$, we use the technique proposed in Sect. 3.2, and thus $\Delta\phi_S^{(m-1)}$ and $\Delta\phi_N^{(m-1)}$ are estimated by the CIP interpolation defined by Eqs. (92) and (93).

Similarly, in the z direction:

$$\frac{\partial}{\partial z}\frac{H\partial\delta\phi(m)}{\partial z} = \chi_{CB}^z \delta\phi_B^{(m)} + \chi_T^z \delta\phi_T^{(m)} - \left(\chi_B^z + \chi_T^z\right)\delta\phi_C^{(m)}, \tag{51}$$

$$\chi_B^z = \frac{H_{CB}}{\Delta z \Delta z_{CB}}, \tag{52}$$

$$\chi_T^z = \frac{H_{CT}}{\Delta z \Delta z_{CT}}, \tag{53}$$

where $\Delta z, \Delta z_{CB}$ and Δz_{CT} are the distances between the points B and T, C and B, and C and T. By the same reason, B and T are estimated by using neighboring points. In this case, however, we need two steps to determine the values. First, these points are estimated by BS and BN as follows:

$$\delta\phi_B^{(m)} = (1 - \eta_B)\delta\phi_{BS}^{(m)} + \eta_B \delta\phi_{BN}^{(m)}, \tag{54}$$

where $\eta_B = (z_B - z_{BS})/(z_{BN} - z_{BS})$. Next, $\delta\phi_{BS}^{(m)}$ and $\delta\phi_{BN}^{(m)}$ are linearly interpolated in the x direction:

$$\delta\phi_{BS}^{(m)} = (1 - \xi_{BS})\delta\phi_{BSm}^{(m)} + \xi_{BS}\delta\phi_{BSp}^{(m)}, \tag{55}$$

where $\xi_{BS} = (x_{BS} - x_{BSm})/(x_{BSp} - x_{BSm})$. Similarly,

$$\delta\phi_{BN}^{(m)} = (1 - \xi_{BN})\delta\phi_{BNm}^{(m)} + \xi_{BN}\delta\phi_{BNp}^{(m)}. \tag{56}$$

These equations are substituted into Eq. (51).

In this way Eq. (39) is modified as follows:

$$\frac{1}{\Delta t}\delta\phi_C^{(m)} - \kappa\theta\sum\chi_l\delta\phi_l^{(m)} = -\frac{1}{\Delta t}\Delta\phi_C^{(m-1)} + \kappa\nabla\cdot\left(H\nabla\left(\theta\Delta\phi^{(m-1)} + \phi^*\right)\right) + \Gamma. \tag{57}$$

This equation is solved in terms of $\delta\phi^m$ by using the BiCGSTAB matrix solver [28–30].

By using this notation, Eq. (30) is calculated as follows:

$$\frac{\delta p_C^{(m)}}{\rho C_s^2 \Delta t^2} - \sum\chi_l\delta p_l^{(m)} = -\frac{\Delta p_C^{(m-1)}}{\rho C_s^2\Delta t^2} + \nabla\cdot\left(\frac{1}{\rho}\nabla(\Delta p^{(m-1)} + p^*)\right) + \frac{\nabla\cdot\mathbf{u}^{*+1}}{\Delta t}. \tag{58}$$

Here $\nabla\cdot\mathbf{u}^{*+\mathbf{1}}$ is estimated in a form of average flux of the control volume. This means that $\partial u/\partial x$ is estimated as follows:

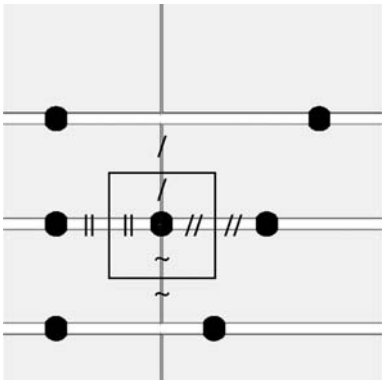

Fig. 3 Control volume around a central point in pressure calculations

$$\frac{\partial u}{\partial x} = \frac{2}{x_{Cp} - x_{Cm}}\left(u\left(\frac{x_C + x_{Cp}}{2}, y_C, z_C\right) - u\left(\frac{x_C + x_{Cm}}{2}, y_C, z_C\right)\right), \tag{59}$$

where $u(x, y, z)$ is estimated by the CIP. In addition, this control volume does not overlap each other and these volumes cover the entire space. Figure 3 shows a two-dimensional control volume. Then y and z directions are calculated as follows:

$$\frac{\partial v}{\partial y} = \frac{2}{y_N - y_S}\left(v\left(x_C, \frac{y_C + y_N}{2}, z_C\right) - v\left(x_C, \frac{y_C + y_S}{2}, z_C\right)\right), \tag{60}$$

$$\frac{\partial w}{\partial z} = \frac{2}{z_T - z_B}\left(w\left(x_C, y_C, \frac{z_C + z_T}{2}\right) - w\left(x_C, y_C, \frac{z_C + z_B}{2}\right)\right). \tag{61}$$

3.2.3 Velocity update

The non-advection part of the momentum equation also includes the pressure gradient (see Eq. (32)). For stable calculations, these need to be estimated like it is done with staggered grids. Thus, the two gradients of the pressure are calculated in the middle of C and Cm and Cp and C, and the pressure gradient at C is estimated by the linear combination of these two gradients as follows:

$$\frac{\partial u}{\partial t} = -\left(\frac{\Delta x_{CCp}}{\Delta x}\frac{(p_C - p_{Cm})}{\rho_{CCm}\Delta x_{CCm}} + \frac{\Delta x_{CCm}}{\Delta x}\frac{(p_{Cp} - p_C)}{\rho_{CCp}\Delta x_{CCp}}\right). \tag{62}$$

Similarly,

$$\frac{\partial v}{\partial t} = -\left(\frac{\Delta y_{CN}}{\Delta y}\frac{(p_C - p(x_C, y_S, z_C))}{\rho_{CS}\Delta y_{CS}} + \frac{\Delta y_{CS}}{\Delta y}\frac{(p(x_C, y_N, z_C) - p_C)}{\rho_{CN}\Delta y_{CN}}\right), \tag{63}$$

$$\frac{\partial w}{\partial t} = -\left(\frac{\Delta z_{CT}}{\Delta z}\frac{(p_C - p(x_C, y_C, z_B))}{\rho_{CB}\Delta z_{CB}} + \frac{\Delta z_{CB}}{\Delta z}\frac{(p(x_C, y_C, z_T) - p_C)}{\rho_{CT}\Delta z_{CT}}\right), \tag{64}$$

where $p(x, y, z)$ is estimated by the CIP interpolation.

3.2.4 Improvement of the temporal accuracy

Using a semi-Lagrangian method [25] and an implicit method, we can choose any time-step size. The problem that we have to consider is the accuracy in time. We usually assume that there is no velocity change during the advection process in Eq. (17). If the velocity changes in time, then it should be estimated as follows:

$$\mathbf{u}(\mathbf{x}, t) = \frac{(t - t^n)\mathbf{u}^{n+1}(\mathbf{x}) + (t^{n+1} - t)\mathbf{u}^n(\mathbf{x})}{t^{n+1} - t^n}. \tag{65}$$

Unfortunately, however, the velocity at the next time step is not yet determined, and thus we propose to approximate it by the following iteration process. At first, we define:

$$l = 0, \tag{66}$$

$$\mathbf{u}^{n+1,(l)} = \mathbf{u}^n, \tag{67}$$

$$\phi^{n+1,(l)} = \phi^n, \tag{68}$$

where l is the iteration counter. Here ϕ represents various physical quantities that vary in time. Thus the advection velocity is estimated as follows:

$$\mathbf{u}(\mathbf{x}, t) = \frac{(t - t^n)\mathbf{u}^{n+1,(l)}(\mathbf{x}) + (t^{n+1} - t)\mathbf{u}^n(\mathbf{x})}{t^{n+1} - t^n}. \tag{69}$$

At every iteration, we regenerate a new grid because the surface of the moving structure will be in a different position during the iterations within a time step. Therefore, $\mathbf{x}$ in Eq. (69) is the location of the new grid points. Since $\mathbf{u}^{n+1,(l)}$ and $\mathbf{u}^n$ are on the old grids, the CIP interpolation is used to estimate them.

Using Eq. (69), Eqs. (15) and (16) are calculated, and the calculated values are stored as data with the superscript $n+1, (l+1)$. Then (l) is incremented by one and we return to the procedure for generating a new grid. As shown in a later section, only one iteration is sufficient. In summary, $\phi^{n+1,(0)} = \phi^n$ is used as the initial guess. At every iteration counted by the iteration counter (l), (a) we generate a new grid, (b) calculate ϕ^* by using Eqs. (17) and (69), (c) solve for pressure by using Eq. (30), and (d) solve for other $\phi^{n+1,(l+1)}$.

4 Test calculations

4.1 Two dimensions

4.1.1 Compressible flow

We first test the proposed scheme on a compressible flow problem. The purpose of this calculation is to confirm the directional independence of the calculation because the lines in the Soroban grid is directed to a specific direction and thus the directional dependence might be expected. A high density and high pressure fluid is placed at the center like a circular cylinder, then this fluid should radially explode away. The calculation area is $0 \le x \le 1$ and $0 \le y \le 1$, with $\rho = 0.1, P = 0.1$ for $R > 0.08$ and $\rho = 1.0, P = 3.0$ for $R < 0.08$, where R is the distance from the center. In this calculation, we set $\gamma = 1.4$. We adopt the scalar numerical viscosity proposed by Ogata and Yabe [26] to avoid the directional dependence of the viscosity for variable grid size. If $\nabla \cdot \mathbf{u} \le 0$,

$$q = -\rho c_v\left(C_s - \frac{\gamma + 1}{2}\lambda\nabla \cdot \mathbf{u}\right)\lambda\nabla \cdot \mathbf{u}, \tag{70}$$

otherwise, $q = 0$, where $c_v = 0.55$, C_s is the local acoustic speed and λ is $(\Delta x \Delta y)^{1/2}$. Here Δx and Δy are the local grid spacings.

Figure 4 is the result after 150 steps ($\Delta t = 5.0 \times 10^{-5}$). Figure 4(a) is the result with the Cartesian grid and 4(b) is the result with the Soroban grid, where the physical quantity used for the monitor is the density, and $\alpha = 7.1$ and $\beta = 1.1$. Although the Soroban grid is not symmetrical, the cylindrical symmetry is not deteriorated. The grid was controlled so that the maximum grid spacing is 0.04 and the minimum spacing is 0.0027.

The number of grid points varies with time as shown in Fig. 5, and the grid location at the final step is shown in Fig. 6. The Soroban grid describes the shock wave more sharply with a number of grid points comparable to the number of grid points for the Cartesian grid. Furthermore, the profiles at $x = y = 0.3$ or 0.7 is more symmetrical. It is important to note that the CPU time for the Soroban grid and the Cartesian grid are 195 [s] and 290 [s] , respectively. The Soroban grid is proven to be faster than non-Soroban grids despite the extra calculation related to the grid generation. The reason is partly because the computational cost is small for grid generation and partly because the number of grid points is smaller for most of the calculation as shown in Fig. 5.

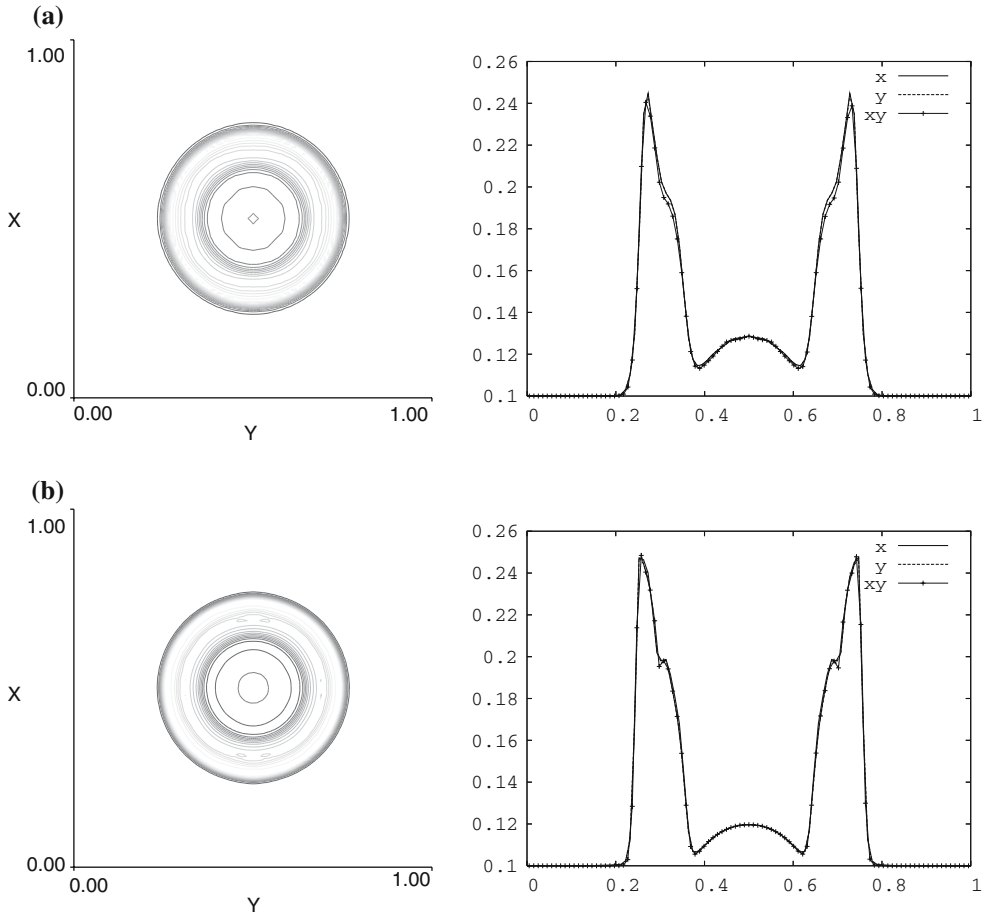

Fig. 4 Explosion of a point. (*Left*) Density contours at $t = 7.5 \times 10^{-2}$, after 150 steps. (*Right*) Cross-section along $x = 0.5, y = 0.5$ and $y = x$. The grid system for (**a**) is a Cartesian grid with $\Delta x = \Delta y = 0.01$ and for (**b**) is the Soroban grid with the number of grid points less than the number grid points of the Cartesian grid, which is 101×101

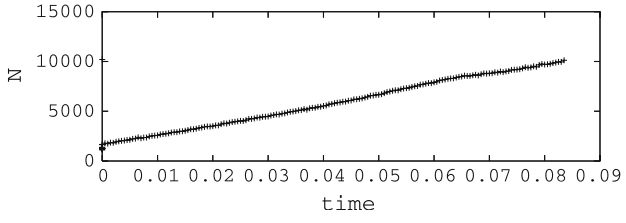

Fig. 5 Time evolution of the number of grid points

4.1.2 Incompressible flow

The next example is incompressible flow past a circular cylinder. The dimension of the computational domain is 60×16, which is normalized by the diameter of the cylinder. The cylinder is located at $(8, 8)$, and the location of the cylinder and the lateral dimension of the domain are the same as those reported in [27]. The initial Soroban grid is shown in Fig. 7. The Soroban lines are parallel to the vertical (y) axis, and the grid points are placed on each line in a way to refine the grid where it is needed.

We also put grid points inside the cylinder, otherwise the interpolations become more complicated. The grids in this case are generated from the monitoring functions that depend on the distance from the surface and the vorticity:

$$M_1(x, y) = \frac{1}{0.005 + 0.1 \times \min(5, D(x, y))}, \tag{71}$$

$$D(x, y) = \sqrt{(x - 8)^2 + (y - 8)^2} - 0.5, \tag{72}$$

$$M_2(x, y, t) = 1 + 50 \times \left| \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right|, \tag{73}$$

$$M(x, y, t) = \min\left(\frac{\sum_{k=1}^{N} M_k(x, y)}{N}, \frac{\Delta x_{\max}}{\Delta x_{\min}} \right). \tag{74}$$

The no-slip boundary condition is imposed on the cylinder surface, and inside the cylinder the velocities are set to $u = v = 0$. We employed the second-order implicit

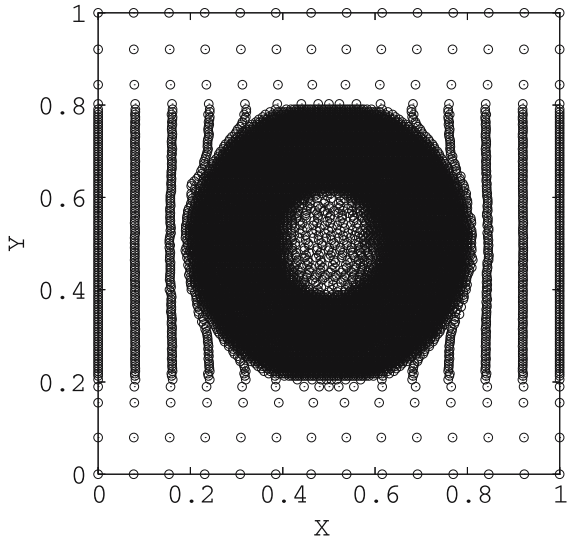

Fig. 6 Grid arrangement for the Soroban grid

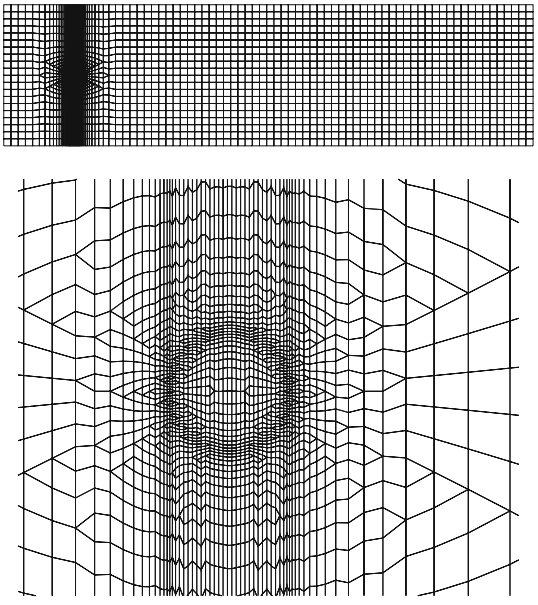

Fig. 7 Initial grid arrangement. The Soroban lines are parallel to the vertical (y) axis. All other lines are for visualization purpose only. (*Top*) Entire grid image. (*Bottom*) Local view near the cylinder

method (i.e. $\theta = 1/2$ in Eq. (38)) for the viscous term. The Reynolds number is 100.

The drag coefficient C_D and lift coefficient C_L are defined as follows:

$$C_D = \frac{F_D}{\frac{1}{2}\rho U_\infty^2 D}, \tag{75}$$

$$C_L = \frac{F_L}{\frac{1}{2}\rho U_\infty^2 D}, \tag{76}$$

where F_D and F_L are the drag and lift forces, respectively. Initially the number grid points is about 4,000,

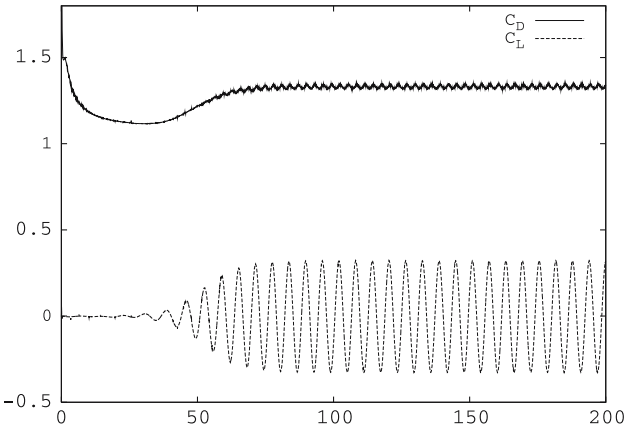

Fig. 8 Time history of the drag coefficient C_D and lift coefficient C_L for the flow past a cylinder at $Re = 100$

and it is eventually increased up to 9,000 points. Figure 8 shows the time history of the drag and lift coefficients. The drag coefficient is 1.375 ± 0.009 and the lift coefficient is ± 0.27. The computed Strouhal number is about 0.16. These values are in good agreement with those reported in [27].

Figure 9 is a snapshot of the adaptive grids. The number of grid points changes in time as shown in Fig. 10. After $t = 80$, the number of grid points stays at the same value, because the number of wake vortices remains almost constant in the computational domain. We compared the calculation with the fine grids as shown in the bottom of Fig. 9. The grid spacing is the same as it is in the finest part of the Soroban grids. The number of grid points is 45,000. Figure 11 shows the maximum vorticity on each line(y direction). This figure indicates that Soroban grids can work very well without introducing any additional diffusion.

Next, we tested the dependence of the solution on the number of iterations used in Eq. (69). Figure 12 shows the lift coefficient in which 'Iteration 1' means $l = 1$. We cannot see a difference between $l = 1$ and $l = 2$. Therefore, it is sufficient to use Eq. (69) only once.

In Fig. 12, we also depict the results from linear interpolation. As explained in Sect. 3.2.2, we normally use the CIP method for interpolation. The figure shows that the use of linear interpolation instead of the CIP gives rise to a large dissipation error as well as error in Strouhal number, which is 0.14 for linear interpolation, while it is 0.164, 0.165, 0.165 for $l = 0, 1, 2$. Thus, accurate interpolation is a key issue in the Soroban grid system.

Since both the semi-Lagrangian technique for the advection terms and the implicit procedure for the pressure and viscous terms make the scheme free from CFL constraints, we can use any time-step size. This becomes very important for locally refined grids that limit the time-step size in conventional schemes. Actually, the

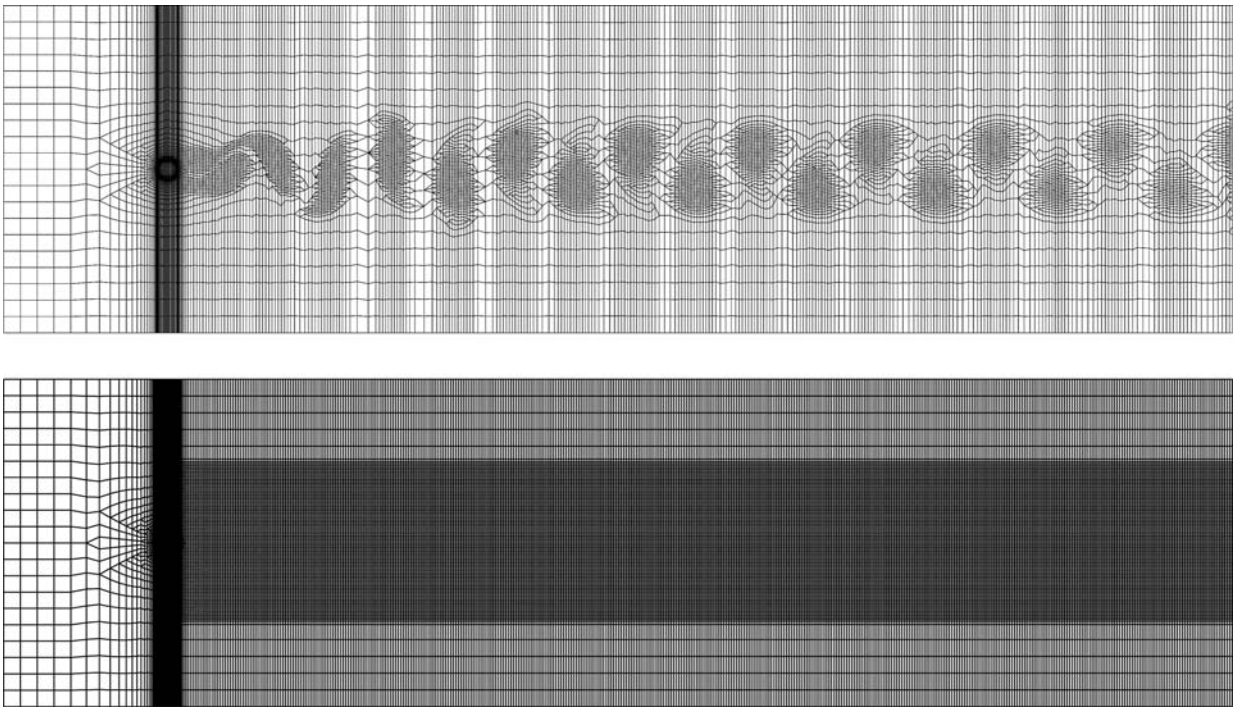

Fig. 9 (*Top*) Snapshot of the adaptive grid. (*Bottom*) Fixed fine grid

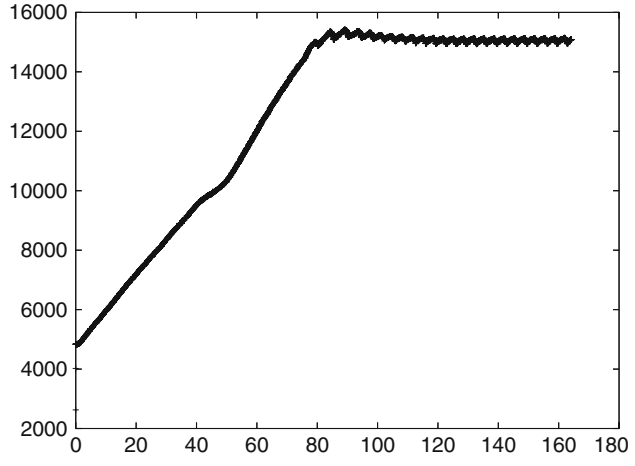

Fig. 10 Time history of the number of grid points in the adaptive calculation

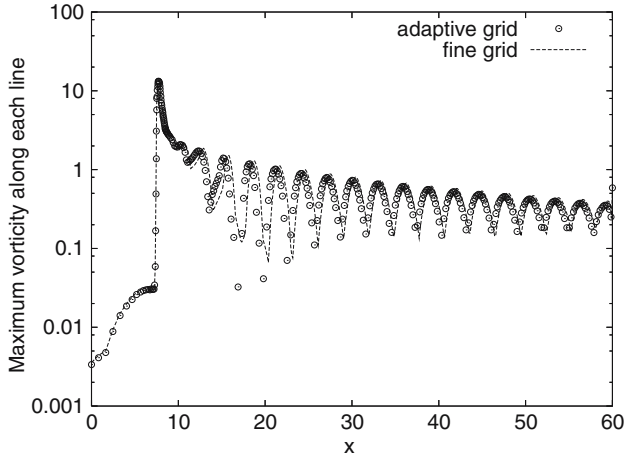

Fig. 11 Maximum magnitude of the vorticity along each Soroban line, given by the adaptive and fixed fine grids

time-step size was set to $\Delta t = 0.05$ regardless of the grid size, and the maximum CFL number is about 10.

4.2 Three dimensions

In this section, we examine a three dimensional case. The CCUP method can calculate compressible flow and incompressible flow at the same time. Then, we added a color function ψ, where $\psi = 1$ for water and $\psi = 0$ for air. The advection of ψ is solved by the tangent-function CIP [31]. Then this ψ is used for the calculation of the physical values at grid points as follows:

$$R_{\text{grid}} = R_{\text{water}}\psi + R_{\text{air}}(1 - \psi), \tag{77}$$

where $R_{\text{grid}}, R_{\text{water}}$ and R_{air} are the physical values at the grid point, and for water and air. In the calculation, the acoustic speed and viscosity of the mixed states are similarly calculated by using this color function.

As a test calculation, we adopt a skimmer (stone-skipping) simulation. The coupling technique we use in dealing with the interaction between the solid and fluid is explained in [32]. At every iteration, (a) we move the object with fluid forces based on the pressure values calculated by using the pressure versions of Eqs. (17)

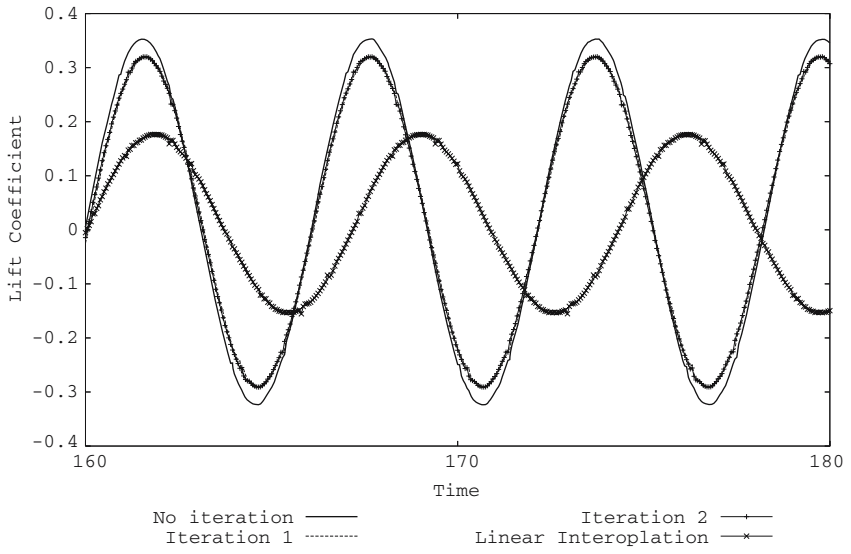

Fig. 12 Time history of the lift coefficient at $Re = 100$

and (69), (b) generate a new grid, (c) calculate ϕ^* by using Eqs. (17) and (69), (d) solve for pressure by using Eq. (30), and (e) solve for other $\phi^{n+1,(l+1)}$. We have already performed experiments, and the detailed movement was visualized by a high-speed camera as shown in Fig. 13 [32]. In the experiment, the disk made of aluminum was used instead of a stone. The feature of this phenomenon is that the disk goes through thin water film, which is generated by the collision of the disk and water surface.

In order to capture such phenomena, the Soroban grid points were collected near the interface of the water and air and also, to a lesser extent, near the solid surfaces. Figures 14 and 15 show the computed results in which the initial velocity of the aluminum disk is set to $(u, v, w) = (5, -1.2, 0)$ m/s, and the angle-of-attack is 10 degrees. The disk is 5 cm in diameter and 1 cm in thickness, rotating with a speed of 20 rad/s. The domain size in the simulation is $0.4 \times 0.14 \times 0.14$ m, and the water depth is 0.05 m. The number of grid points is not constant but is about 120,000.

The calculation was efficiently advanced by automatically repeating grid generation. Figure 16 shows the grid arrangement. A vertical plane shows the grid arrangement where we connected the grid points, like it is in a finite element method, but the connectivity is for the purpose of visualization only. Also for visualization purpose, grid points in three dimensions are depicted by spheres whose size depends on the density and hence the grid points in the air are not visible.

5 Concluding remarks

We have proposed, for fluid–object and fluid–structure interactions, the Soroban CCUP method, which uses a collocated-grid approach. Multi-fluid flows with density ratios reaching 1000 and fluid–object interactions were computed robustly and accurately. The use of the CIP interpolation in the estimation of the values from the neighboring points enhances the accuracy compared to using linear interpolation. In solving the matrix systems, we avoided the inclusion of the derivatives and proposed an efficient iteration procedure.

By using a semi-Lagrangian CIP technique and by treating the viscous terms implicitly, we removed the time-step size limitations to a large extent, and in some cases we were able to compute with the CFL number reaching 10. Therefore, we are able to use large time-step sizes even with very small mesh sizes in refined-mesh areas.

Because of somewhat limited movement of the Soroban grid and the resulting ease in memory management, searching for the neighboring points is very fast and hence the time required for dynamical grid generation is negligibly small. The Soroban grid technique we currently have imposes, for the sake of computational efficiency, fine grid resolution (in directions perpendicular to the Soroban grid lines and planes) even in locations far from the solid objects. This is one of the aspects of the method that we plan to improve in the future, without significantly compromising the efficiency.

6 Appendix

6.1 Specifying the grid point locations

The location of a grid point is specified by the Cartesian coordinate (x, y, z) in the three-dimensional space. However, the grid points along a line has the same value

Fig. 13 Skimmer experiment

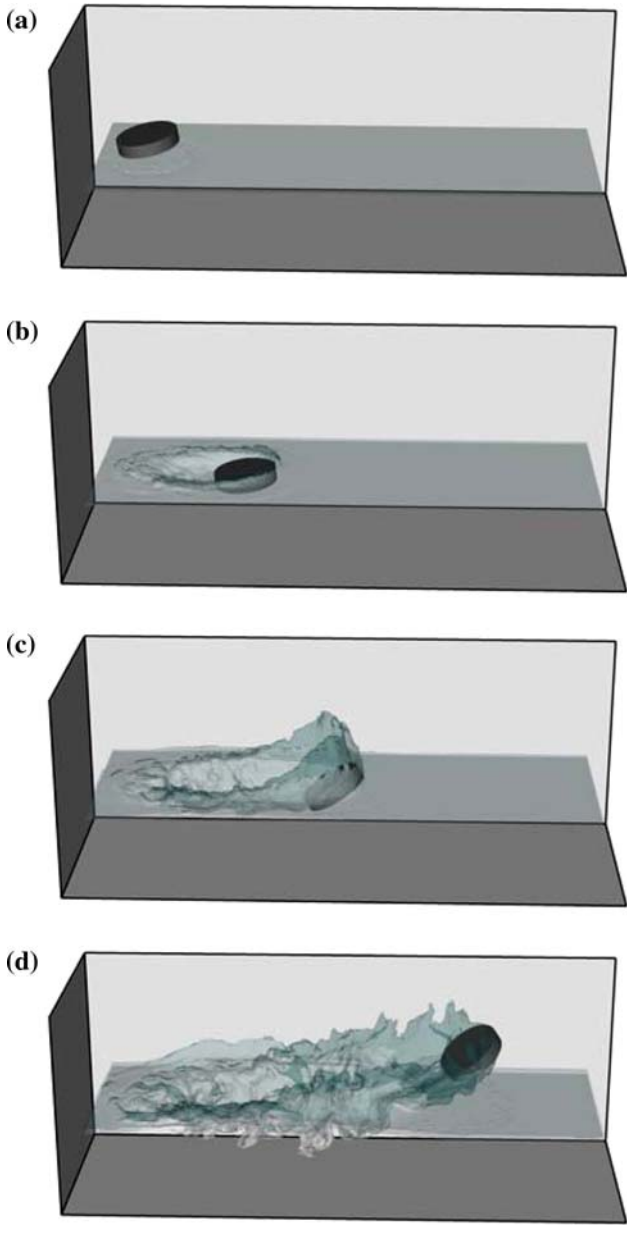

Fig. 14 Skimmer simulation (side view)

of y and z. Therefore, we specify the plane by the index k^P and call it the k-th plane. Similarly the line is specified by the index j^L. Since j^L is a one-dimensional index and continuously increases over all the planes, we need to introduce the starting index $JLS[k^P]$, which means that the line index j^L starts from $JLS[k^P]$ on the k^P plane.

Similarly, the grid point is specified by the index i^G, which is also a one-dimensional index and continuously increases over all the lines, and we introduce the starting index $IGS[j^L]$ on the j^L line. The grid points are stored in a one-dimensional array in the order shown in Table 1.

In the calculation described in the next section, we need to know where the point (x_0, y_0, z_0) is located in this system. First, z_0 is specified by two planes as follows:

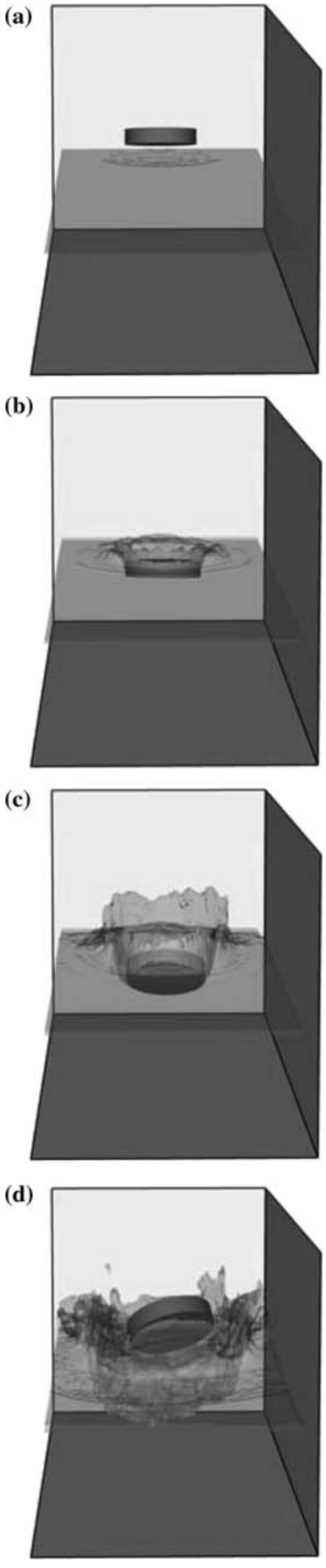

Fig. 15 Skimmer simulation (front view)

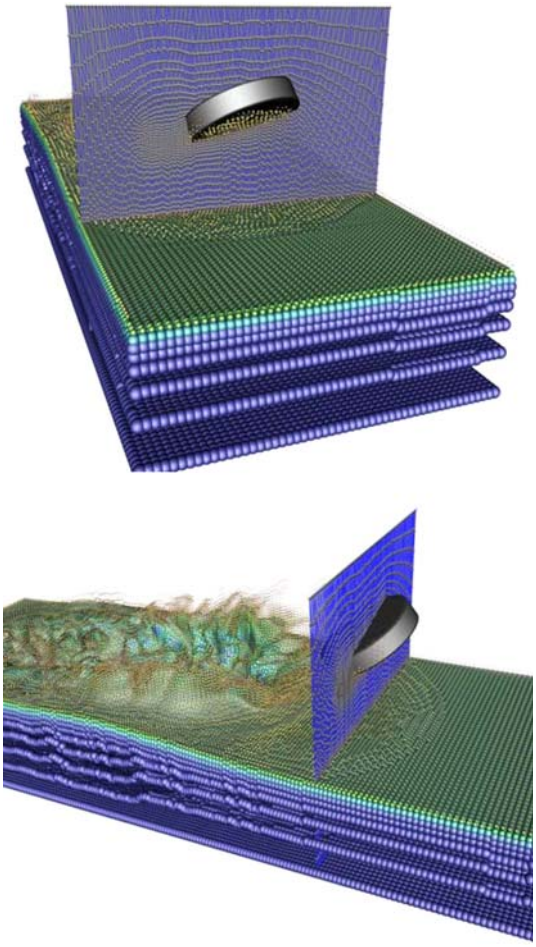

Fig. 16 Grid points used in the simulation. The vertical plane shows the grid arrangement. For visualization, grid points in three dimensions are depicted by *spheres* whose sizes represent the density

Table 1 JLS denotes the start index for the lines on each plane. IGS denotes the start index for the grid points on each line. For example, the number of grid points is 2 and 3 for the 7-th and 8-th lines. See also Fig. 1

Plane number	1	2	3	...		
JLS	1	4	7	...		
Line number	...	7	8	9	10	...
IGS	...	17	19	22	24	...

$$z[k^P] \leq z_0 < z[k^P + 1], \tag{78}$$

as shown in Fig. 17. There would be several methods of searching the k^P as proposed in our previous paper [5], but in this paper, we use the binary search tree algorithm since the cost of computation is very small in a one-dimensional search. Since the system is arranged as

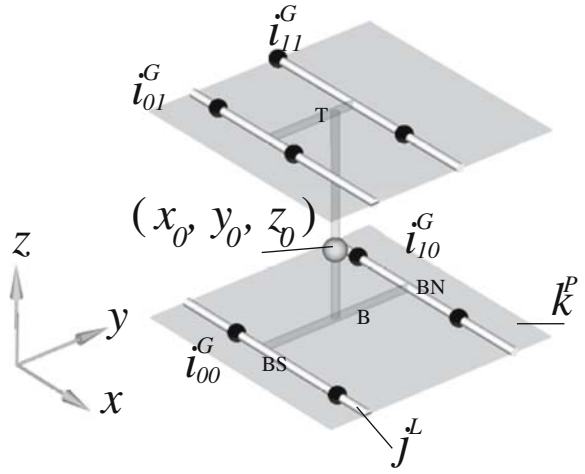

Fig. 17 Location of a point (x_0, y_0, z_0) in the Soroban grid. The first and second subscripts of i^G specify the line and plane numbers, respectively

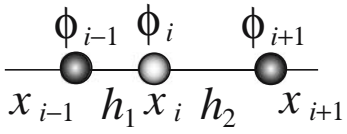

Fig. 18 One-dimensional non-regular grid

in Fig. 17, we need to find two lines on the k^P plane and another two lines on the $k^P + 1$ plane so that

$$y[j^L] \le y_0 < y[j^L + 1] \tag{79}$$

is satisfied. We note that there are two j^Ls; one is for the k^P plane and the other is for the $k^P + 1$ plane.

Similarly, we search for x_0 on the j^L-th line:

$$x[i^G] \le x_0 < x[i^G + 1]. \tag{80}$$

We note that there are four i^Gs shown in Fig. 17.

6.2 One-dimensional CIP Interpolation

CIP1D is to estimate the function by the CIP method. In fact we use the spatial derivatives as follows,

$$\mathrm{CIP1D}\,(\phi_0, \phi_1, X, D) = \mathrm{CIP}(\phi_0, \partial_x\phi_0, \phi_1, \partial x\phi_1, X, D) \tag{81}$$

$$\mathrm{CIP}(\phi_0, \partial_x\phi_0, \phi_1, \partial_x\phi_1, X, D) = \Sigma_{k=0}^{3} C_k X^k \tag{82}$$

$$\partial_x \mathrm{CIP}(\phi_0, \partial_x\phi_0, \phi_1, \partial_x\phi_1, X, D) = \Sigma_{k=1}^{3} k \cdot C_k X^{k-1} \tag{83}$$

$$C_0 = \phi_0, \tag{84}$$

$$C_1 = \partial_x\phi_0, \tag{85}$$

$$C_2 = \frac{3(\phi_0 + \phi_1)}{D^2} - \frac{2\partial_x\phi_0 - \partial_x\phi_1}{D^3}, \tag{86}$$

$$C_3 = \frac{\partial_x\phi_0 + \partial\phi_1}{D^2} + \frac{2(\phi_0 - \phi_1)}{D^3}. \tag{87}$$

This is x directional estimation, when we calculate along the y direction, spatial derivatives of y direction are required.

6.3 Preliminary concepts of finite differences on the Soroban grid

In estimating the finite difference of $\partial\phi/\partial x$ and the terms in G , we need special consideration because for example as shown in Fig. 17, all the points in general are not always on a straight line. In this paper, we interpolate the values needed for finite differences by using the grid points. Usually the interpolation deteriorates the accuracy. As already demonstrated in a previous paper [5], the CIP method can very accurately retrieve a profile between the grid points.

We should just consider two derivatives. One is the first spatial derivative, the other is the second derivative. In one dimension, it is just using finite differences over non-regular grids.

For example,

$$h_1 = x_i - x_{i-1}, \tag{88}$$

$$h_2 = x_{i+1} - x_i, \tag{89}$$

$$\frac{\partial\phi_i}{\partial x} = \frac{1}{h_1 + h_2}\left(h_1\frac{\phi_{i+1} - \phi_i}{h2} - h_2\frac{\phi_i - \phi_{i-1}}{h1}\right), \tag{90}$$

$$\frac{\partial^2\phi_i}{\partial x^2} = \frac{2}{h_1 + h_2}\left(\frac{\phi_i - \phi_{i-1}}{h_1} + \frac{\phi_{i+1} - \phi_i}{h_2}\right). \tag{91}$$

In two dimensions, shown in the bottom-left part of Fig. 19, estimation of the derivative at C needs the values at N and S along the vertical line, but these two points do not in general coincide with the grid points. Then we use CIP interpolation like we do in the advection part.

The value at S can be estimated by the values at Sm and Sp. This is symbolically written as

$$\phi_S = \mathrm{CIP1D}\left(\phi_{Sm}, \phi_{Sp}, x_S - x_{Sm}, x_{Sp} - x_{Sm}\right). \tag{92}$$

Similarly at N,

$$\phi_N = \mathrm{CIP1D}\left(\phi_{Nm}, \phi_{Np}, x_N - x_{Nm}, x_{Np} - x_{Nm}\right). \tag{93}$$

In this way, we get $\phi_S, \phi_N, h_1 = y_j - y_{j-1}, h_2 = y_{j+1} - y_j$. Then, we can calculate the spatial derivative in the y-direction at C.

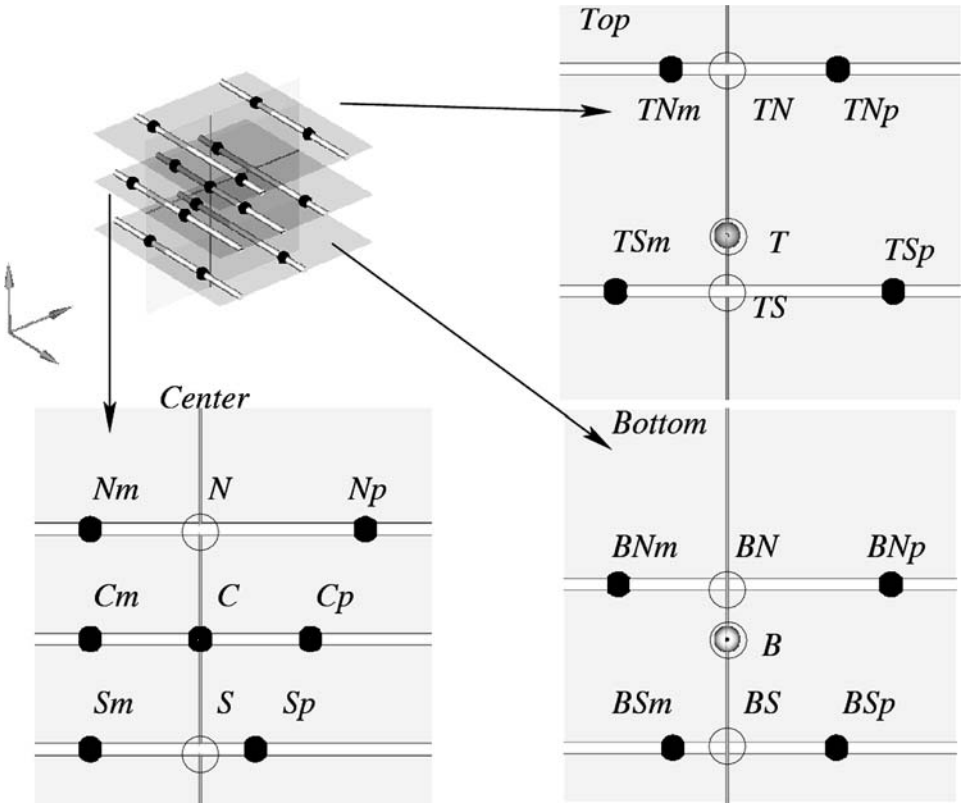

Fig. 19 *Black points* are the grid points. *Circles* are the points that must be estimated from the grid points on a line or plane. The *shaded circles* inside the circles at T and B show the location of the point C in other planes. The symbol C denotes center, N and S denote north and south, and B and T denote bottom and top. Lower case m and p are the minus and plus sides along a line

The derivatives in the z direction are also estimated by ϕ_B and ϕ_T on the neighboring planes. ϕ_B are estimated by ϕ_{BS} and ϕ_{BN} as follows:

$$\phi_B = \mathrm{CIP1D}\left(\phi_{BS}, \phi_{BN}, y_B - y_{BS}, y_{BN} - y_{BS}\right), \tag{94}$$

where ϕ_{BN} and ϕ_{BS} are estimated like N and S. For example,

$$\phi_{BS} = \mathrm{CIP1D}\left(\phi_{BSm}, \phi_{BSp}, x_{BS} - x_{BSm}, x_{BSp} - x_{BSm}\right). \tag{95}$$

References

1. Takewaki H, Nishiguchi A, Yabe T (1985) The cubic-interpolated pseudo-particle (CIP) method for solving hyperbolic-type equations. J Comput Phys 61:261–268
2. Takewaki H, Yabe T (1987) Cubic-interpolated pseudo particle (CIP) method – Application to nonlinear or multi-dimensional problems. J Comput Phys 70:355–372
3. Yabe T, Aoki T (1991) A universal solver for hyperbolic-equations by cubic-polynomial interpolation. I. One-dimensional solver. Comput Phys Commun 66:219–232
4. Yabe T, Xiao F, Utsumi T (2001) Constrained interpolation profile method for multiphase analysis. J Comput Phys 169:556–593
5. Yabe T, Mizoe H, Takizawa K, Moriki H, Im H, Ogata Y (2004) Higher-order schemes with CIP method and adaptive Soroban grid towards mesh-free scheme. J Comput Phys 194:57–77
6. Yabe T, Wang PY (1991) Unified numerical procedure for compressible and incompressible fluid. Phys Soc Jpn J 60:2105–2108
7. Tezduyar TE, Behr M, Liou J (1992) A new strategy for finite element computations involving moving boundaries and interfaces – The deforming-spatial-domain/space–time procedure: I. The concept and the preliminary numerical tests. Comput Methods Appl Mech Eng 94:339–351
8. Tezduyar TE, Behr M, Mittal S, Liou J (1992) A new strategy for finite element computations involving moving boundaries and interfaces – The deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. Comput Methods Appl Mech Eng 94:353–371
9. Johnson AA, Tezduyar TE (1996) Simulation of multiple spheres falling in a liquid-filled tube. Comput Methods Appl Mech Eng 134:351–373
10. Johnson AA, Tezduyar TE (1997) 3D simulation of fluid-particle interactions with the number of particles reaching 100. Comput Methods Appl Mech Eng 145:301–321
11. Mittal S, Tezduyar TE (1995) Parallel finite element simulation of 3D incompressible flows – Fluid-structure interactions. Int J Numerical Methods Fluids 21:933–953
12. Stein K, Benney R, Kalro V, Tezduyar TE, Leonard J, Accorsi M (2000) Parachute fluid–structure interactions: 3-D Computation. Comput Methods Appl Mech Eng 190:373–386
13. Wall W (1999) Fluid–Structure Interaction with Stabilized Finite Elements. Ph.D. thesis, University of Stuttgart
14. Heil M (2004) An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. Comput Methods Appl Mech Eng 193:1–23

15. Hubner B, Walhorn E, Dinkler D (2004) A monolithic approach to fluid–structure interaction using space–time finite elements. Comput Methods Appl Mech Eng 193:2087–2104
16. Dettmer W (2004) Finite element modeling of fluid flow with moving free surfaces and interfaces including fluid–solid interaction. Ph.D. thesis, University of Wales Swansea
17. Tezduyar TE (2001) Finite element methods for flow problems with moving boundaries and interfaces. Arch Comput Methods Eng 8:83–130
18. Berger MJ, Oliger J (1984) Adaptive mesh refinement for hyperbolic partial differential equations. J Comput Phys 53:484–512
19. Kirkpatrick MP, Armfield SW, Kent JH (2003) A representation of curved boundaries for the solution of the Navier-Stokes equations on a staggered three-dimensional Cartesian grid. J Comput Phys 184:1–36
20. Unverdi SO, Tryggvasson GA (1992) A front-tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 100:25–37
21. Enright D, Fedkiw R, Ferziger J, Mitchell I (2002) A hybrid particle level set method for improved interface capturing. J Comput Phys 183:83–116
22. Harlow FH, Amsden AA (1968) Numerical calculation of almost incompressible flow. J Comput Phys 3:80–93
23. Yoon SY, Yabe T (1999) The unified simulation for incompressible and compressible flow by the predictor-corrector scheme based on the CIP method. Comput Phys Commun 119:149–158
24. Aoki T (1995) Multi-dimensional advection of CIP (cubic-interpolate propagation) scheme. CFD J 4:279–291
25. Staniforth A, Côté J (1991) Semi-Lagrangian integration scheme for atmospheric models - A review. Mon Weather Rev 119:2206–2223
26. Ogata Y, Yabe T (1999) Shock capturing with improved numerical viscosity in primitive Euler representation. Comput Phys Commun 119:1799–193
27. Tezduyar TE, Liou J, Ganjo DK (1990) Incompressible flow computations based on the vorticity-stream function and velocity-pressure formulations. Comput Struct 35:445–472
28. van der Vorst HA (1992) Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non symmetric linear systems. SIAM J Sci Stat Comput 13:631–644
29. Lee J, Zhang J, Lu C (2003) Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems. J Comput Phys 185:158–175
30. Show E, Saad Y (1997) Experimental study of ILU preconditioners for indefinite matrices. J Comput Appl Math 86:387–414
31. Yabe T, Xiao F (1993) Description of complex and sharp interface during shock wave interaction with liquid drop. Phys Soc Jpn J 62:2537–2540
32. Takizawa K, Yabe T, Chino M, Kawai T, Wataji K, Hoshino H, Watanabe T (2005) Simulation and experiment on swimming fish and skimmer by CIP method. Comput Struct 83:397–408