AN IMMERSED BOUNDARY METHOD WITH FEEDBACK FORCING FOR SIMULATION OF FLOW AROUND AN ARBITRARILY MOVING BODY

S. J. Shin¹, W.-X. Huang¹, H. J. Sung¹
¹ Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, KOREA

ABSTRACT

We present an improved method for computing incompressible viscous flow around an arbitrarily moving body on a fixed computational grid. The main idea is to incorporate feedback forcing scheme of Peskin’s virtual boundary method using momentum forcing in the Navier-Stokes equations has received much attention because it can handle easily viscous flow over or inside complex geometries with Cartesian grids which generally do not coincide with the body surface. The IB method can be classified into two categories, depending on how momentum forcing is applied [1]. One is discrete forcing approach and the other is continuous forcing approach. Compare with discrete forcing approach, continuous forcing approach is easy to be expanded in three dimensional cases and straightforward in flows with moving boundaries due to its simple formation. Details about comparison of two approach can be found in Mittal et al. [1]

Continuous forcing approach can be divided in two methods: Peskin’s IB method and the virtual boundary method. Peskin’s IB method was first proposed by Peskin [2] to simulate flows inside a moving heart. The basic idea is to determine a singular force distribution at arbitrary Lagrangian positions and to apply it to the flow equations in the fixed reference frame via a regularized delta function. The careful design of Peskin’s delta function is vital to the efficiency of the method.

Goldstein et al. [3] developed a virtual boundary method that employs a feedback forcing to enforce the no-slip condition at immersed boundaries embedded in the fluid domain. Saiki and Biringer [4] modified the virtual boundary formulation and proposed the so-called ‘area-weighted’ virtual boundary method. Several papers using this method have reported that this method suffers from a very strict time-step restriction because the amplitude of the feedback forcing needs to be large for proper operation, resulting in a very stiff system. But Lee [5] relieves time restriction by investigating the stability characteristics of the virtual boundary method. He simulated turbulent flows with complex boundaries using an order-one CFL number.

In the present study, we sought to develop a new IB method that combines the merits of the two previous methods, i.e., the feedback forcing scheme of the virtual boundary method and Peskin’s delta function transformation. To this end, we made a detailed comparison of Peskin’s IB method and the virtual boundary method. And we analyzed the stability of the proposed IB method to relieve time restriction. The present method is applied to flow around several different moving problems.

NUMERICAL APPROACH

Spatial discretization of Eulerian and Lagrangian variables

The computational domain is divided in two parts. The fluid is an Eulerian domain and fluid-solid interface is a Lagrangian domain. The Eulerian domain is defined as a fixed Cartesian grid.

Fig.1 shows a circular solid object embeded in fluid. We do not consider the interior of the solid object as Lagrangian domain for reasons of efficiency [6]. The fluid-solid interface $\partial S$ is evenly distributed by $N_L$ points which are denoted by

$$X_i \in \partial S \quad 1 \leq i \leq N_L,$$  \hspace{1cm} (1)

Each Lagrangian point has a discrete volume $\Delta V_i$ with thickness equal to the mesh width $h$. And $N_L$ is selected to hold that a discrete volume is comparable to a finite volume of the Eulerian grid, i.e. $\Delta V_i^{(N)} = h^D$, where $D$ is the number of space-dimensions. Although we illustrate only circular solid object, the present method can be applied to arbitrarily shaped objects.

Regularized delta function

The regularized delta function is used to transfer quantities between Lagrangian and Eulerian locations,

$$U(x) = \sum_{x \in \partial S} u(x) \delta_h (x - X_l) h^3, \quad 1 \leq l \leq N_L,$$ \hspace{1cm} (2)

$$f(x) = \sum_{l=1}^{N_L} F(X_l) \delta_h (x - X_l) \Delta V_l,$$ \hspace{1cm} (3)
where capital letters mean Lagrangian quantities and small letters mean Eulerian quantities. We apply regularized delta functions in Eq.(4).

\[ \delta_r(x) = \frac{1}{h} \, \phi \left( \frac{x - X}{h} \right) - \frac{1}{h} \, \phi \left( \frac{x - X}{h} \right) \, , \tag{4} \]

In Fig.2, 2-point regularized delta function is similar with the ‘area-weighted’ virtual boundary method of Saiki and Biringer [4]. And 3-point regularized delta function is introduced by Roma et al. [7] and 4-point regularized delta function is introduced by Peskin [8], respectively. It is noted that all three type delta function have property as below

\[ \sum \delta_r(x - X)h^3 = 1 \, , \tag{5} \]

which is discrete analogue to basic property of the Dirac delta function. The value of \( \phi(r) \) is maximum at \( r = 0 \) in three types of regularized delta functions and the value of \( \phi(0) \) decreases, as points of delta function increase. These properties are the kernels of stability analysis shown in the next section.

![Fig.2. Three types of regularized delta functions](image)

**The flow solver**

The present method is based on the N–S solver adopting the fractional step method and a staggered Cartesian grid system. A fully implicit time-advancement is employed where both the convection and diffusion terms are advanced with the second-order Crank–Nicholson scheme. The governing equations are discretized as below,

\[ \frac{u^{n+1} - u^n}{\Delta t} + N u^{n+1} = -G p^{n+1/2} + \frac{1}{2 \, \text{Re}} \left( L u^{n+1} + L u^n \right) + f^n \, , \tag{6} \]

\[ D u^{n+1} = 0 \, , \tag{7} \]

where \( N \) is the linearized discrete convective operator, \( G \) is the discrete gradient operator, \( L \) is the discrete Laplacian operator and \( D \) is the discrete divergence operator, respectively. Here \( n \) denotes the \( n \) th time step and \( \Delta t \) denotes the time increment. The spatial discrete operators \( N,G,L \) and \( D \) are evaluated using the second-order central finite-difference scheme.

The discretized equations, including the fluid-solid coupling term are shown as follows,

\[ F^n = \alpha \int (U^n(X_i) - U^n_j(X_j)) \, dt \, + \beta (U^n(X_i) - U^n_j(X_j)) \, , \tag{8} \]

\[ f^n = \int F^n \delta_r(x - X_i) \, ds \, , \tag{9} \]

\[ \frac{1}{\Delta t} u^* + N u^* = -G p^{n+1/2} + \frac{1}{2 \, \text{Re}} \left( L u^* + L u^* \right) + f^* \, , \tag{10} \]

\[ \Delta t D G \delta p = D u^* \, , \tag{11} \]

\[ u^{n+1} = u^* - \Delta t G \delta p \, , \tag{12} \]

\[ p^{n+1/2} = p^{n-1/2} + \delta p \, , \tag{13} \]

\[ U^{n+1} = \int u^{n+1} \delta_r(x - X^{n+1}_m) \, dx \, , \tag{14} \]

where \( U^n_j(X_j) \) is the desired velocity of Lagrangian point and \( u^* \) is the intermediate fluid velocity. In the present method, the Lagrangian point is desired to move along the rigid-body motion of solid object. So the feedback forcing in Eq.(8) is calculated to make the velocity at the Lagrangian point to equal the desired velocity. It is noted that feedback forcing points are located in a staggered fashion like the velocity components defined on a staggered grid. Next, we transfer feedback forcing in the Lagrangian domain to the Eulerian domain by using the regularized delta function (Eq.(9)). We can add the Eulerian forcing \( f^n \) as a momentum forcing to N-S equations. If the momentum forcing is calculated implicitly [9], a large sparse matrix is introduced for a complicate interpolation scheme which gives a significant increase of computing cost. So the momentum forcing is calculated explicitly for reason of efficiency. In Eqs.(10)-(13), we solve the \( u^{n+1} \) and \( p^{n+1/2} \) using the fractional step method. Details regarding the present N–S solver can be found in Kim et al. [10]. We interpolate the Eulerian velocity to the Lagrangian point by using the regularized delta function as kernels in the tranfer step. We use Lagrangian velocity obtained in Eq.(14) to calculate feedback forcing in next time step again. This procedure is repeated.

**STABILITY ANALYSIS**

First, we consider the simple case in which the Lagrangian domain is a point located at \( x_1 + r(x_2 - x_1) \) between two Eulerian points, \( x_1 \) and \( x_2 \) in one-dimensional space as shown in Fig.3(a). The velocity at the Lagrangian point \( U(t) \) is obtained using the 3-point regularized delta function according to Eq.(2).

\[ U(t) = \phi(-r)u_1(t) + \phi(-r)u_2(t) + \phi(-r)u_3(t) \, , \tag{15} \]

where \( u_1(t),u_2(t) \) and \( u_3(t) \) are the velocities at the Eulerian points \( x_1, x_2 \) and \( x_3 \) respectively. The value of \( r \) varies between 0.5 and 1.5, depending on the location of the Lagrangian point. The momentum
forcing is calculated by the feedback law (Eq. (8)) and spreads back to the three nearby Eulerian points using Eq. (3) as follows

\[ f_2(t) = \phi(2-r)F(t) + \phi(1-r)F(t) + \phi(1)F(t) \]

(16)

Since only one Lagrangian point is considered, no summation is necessary. Stability analysis for this method should be carried out using the largest Eulerian forcing as this represents the worst case. When the Lagrangian point coincides with the Eulerian point at \( r = 1 \), the Eulerian forcing reaches its maximum, i.e.,

\[ f_2(t) = \phi(0)F(t) = \frac{2}{3} F(t) \]

(17)

where \( \phi(0) = \frac{2}{3} \) for the 3-point regularized delta function.

Fig. 3. Virtual boundary velocity and forcing in one dimensional case: (a) Lagrangian domain: point (b) Lagrangian domain: line, \( \cdot \), Lagrangian grid point; \( \bullet \), Eulerian grid point

Next, we consider the case in which the Lagrangian domain is a line immersed in a one-dimensional Eulerian domain (Fig. 3(b)). For simplicity, we assume that the adjacent Eulerian velocities are uniform. Hence \( U(t) = u(t) \) according to Eq. (5), regardless of the position of the Lagrangian point. The momentum forcing at the Eulerian point \( x_i \) is then obtained

\[ f_2(t) = \phi(2-r)F(t) + \phi(1-r)F(t) + \phi(r)F(t) = F(t) \]

(18)

Similarly, we can obtain the maximum forcing for other cases; the results are listed in Table 1. In its general form, the maximum forcing can be expressed as

\[ f_{max}(t) = C_{max}F(t) = C_{max}\left( \alpha \int_0^t u(t)\,dt^+ + \beta u(t) \right) \]

(19)

where \( C_{max} \) is the coefficient of the maximum Eulerian forcing for each case. Here we consider a stationary problem for simplicity, i.e., \( U(t) = 0 \) in Eq. (8). Hence the N-S equation can be written as

\[ \frac{u^{n+1} - u^n}{\Delta t} + Nu^{n+1} = -G\alpha u^n + \frac{1}{2Re} \left( Lu^{n+1} + Lu^n \right) + C_{max}(\alpha \sum_{i=0}^n u^{n+1} + \beta u^n) \]

(20)

where \( \sum_{i=0}^n u^{n+1} \) is an approximation to \( \int_0^t u(t)\,dt^+ \). In the present method, since \( \alpha \approx \frac{1}{\Delta t} \) and \( \beta \approx \frac{1}{\Delta t} \) are much larger than the other terms, we can simplify Eq. (21) to

\[ u^{n+1} - u^n \approx C_{max} \alpha \int_0^t u(t)\,dt^+ + \beta u^n \]

To obtain the recurrence formula for stability analysis, the equation at the previous time step is subtracted from the equation at present time step, resulting in

\[ u^{n+1} - 2u^n + u^{n-1} = \alpha' u^n + \beta'(u^n - u^{n-1}) \]

(22)

Where \( \alpha' = C_{max} \alpha \Delta t^2 \) and \( \beta' = C_{max} \beta \Delta t \). Substitution of \( u^n = u^n r^n \) \((r=\frac{u^{n+1}}{u^n})\) into Eq. (22) leads to the equation for \( r \).

\[ r^{2} - (2 + \alpha' + \beta') r + 1 + \beta' = 0 \]

(23)

Thus the stability region \( abs(r) \leq 1 \) is

\[ -\alpha' - 2\beta' \leq 4 \Rightarrow -C_{max} \alpha \Delta t^2 - 2C_{max} \beta \Delta t \leq 4 \]

(24)

From the above equation, we can see that the stability region is mainly affected by the coefficient \( C_{max} \), which depends on the type of Lagrangian domain and Eulerian domain, as shown in Table 1. For example, for the case in which the Lagrangian domain is a line in a 2-D flow, \( C_{max} = \phi(0) \) which is smaller than \( C_{max} = 1 \) for the case where the Lagrangian domain is a plane in a 2-D flow. In other words, without applying the Lagrangian forcing to the interior of the solid object (Fig. 1), the stability region relaxes for a given feedback forcing gain \( \alpha, \beta \), as shown in Eq. (24). The stability region can also be affected by the time advancing scheme. Although only the forward Euler scheme is considered here, the stability regimes for various time advancing schemes can be obtained in an imilar way [5].

Table 1. Maximum Eulerian forcing for 1-D, 2-D, 3-D cases.

<table>
<thead>
<tr>
<th></th>
<th>1D</th>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>( \phi(0)F(t) )</td>
<td>( \phi(0)^2 F(t) )</td>
<td>( \phi(0)^3 F(t) )</td>
</tr>
<tr>
<td>Line</td>
<td>( F(t) )</td>
<td>( \phi(0)F(t) )</td>
<td>( \phi(0)^2 F(t) )</td>
</tr>
<tr>
<td>Area</td>
<td>( F(t) )</td>
<td>( \phi(0)F(t) )</td>
<td>( \phi(0)^2 F(t) )</td>
</tr>
</tbody>
</table>

The stability regimes of different types of delta functions are displayed in Fig. 4 for the case in which the Lagrangian domain is a line immersed in 2-D flow. The flow is stable in the region below the line, and unstable above the line. The stability regimes are wider for regularized delta functions supported by more points. Compared with the 2-point regularized delta function, the stability region of the 4-point regularized delta function is
twice as wide in each direction ($-\alpha \Delta t^2, -\beta \Delta t$). To validate the numerical analysis, simulations of a moving cylinder in a 2-D flow using the 4-point regularized delta function were carried out for different $-\alpha \Delta t^2$ and $-\beta \Delta t$. Stable and unstable cases are denoted by circles and crosses, respectively, in Fig.4. We can see that the analytical solution is in good agreement with the numerical results obtained using the present IB method, despite the fact that the stability analysis was conducted assuming that the adjacent Eulerian velocities are uniform for simplicity. In fact, the adjacent velocities are not much different due to the small mesh width. Since the worst case is always considered in the theoretical analysis, the actual stability regions of the numerical results will be slightly wider than those of the analytical predictions, as shown in Fig.4.

**COMPARISON BETWEEN PESKIN’S IB METHOD AND THE VIRTUAL BOUNDARY METHOD**

**Feedback forcing scheme**

Flow around a rigid moving boundary can be solved by two methods in a continuous forcing approach among IB methods [1]. One is Peskin’s IB method, the other is a virtual boundary method. Peskin’s IB method was originally proposed to solve elastic boundary problems. Lai & Peskin [11] applied Peskin’s IB method to simulate the flow past a cylinder by considering the body to be elastic but extremely stiff. In Fig.5(a), to make immersed boundary location $X_{ib}(s,t)$ attach to body surface position $X_{s}(s,t)$, a feedback forcing term is defined as follow

$$F(s,t) = \kappa (X_{s}(s,t) - X_{ib}(s,t)),$$  

(25)

where $\kappa$ is a positive constant such that $\kappa \geq 1$. And the immersed boundary location $X_{ib}(s,t)$ of next time step moves with the local fluid velocity. As shown in Fig.5(b), the virtual boundary method considers the body surface $X_{vb}(s,t)$ as a virtually existent boundary where feedback forcing is applied on the fluid so that the fluid will be at rest on the surface (no-slip condition). A feedback forcing term is

$$F(s,t) = \alpha \int_{0}^{\tau} (U_{ib}(s,t') - U_{s}(s,t'))dt + \beta (U_{ib}(s,t) - U_{s}(s,t)),$$  

(26)

where $\alpha$ and $\beta$ are large negative constants. And virtual boundary location $X_{vb}(s,t)$ is considered as body surface position $X_{s}(s,t)$.

![Fig.5. Schematic diagram for comparison of two method (a) Peskin’s IB method (b) virtual boundary method](image)

Peskin’s IB method uses a boundary whose position moves according to local flow velocity. however, the virtual boundary method uses a boundary that does not move from the body surface. Therefore, in Peskin’s IB method, the position compensation has to correct the position and velocity of each point on the boundary. However, in the virtual boundary method, the velocity compensation has to correct only the velocity of each point on the boundary, because the boundary remains on the body surface in a virtual boundary method. Such differences require $|\kappa|$ larger than $|\alpha|$ in order to ensure an accurate body surface conditions. This suggests that the time-step restriction will be stricter for Peskin’s IB method than for the virtual boundary method.

**Transfer of quantities between Lagrangian and Eulerian locations**

In both methods, a transfer process between Lagrangian and Eulerian locations is necessary since the virtual boundary points do not always coincide with the computational meshes in the discrete mesh system. The transfer process of Peskin’s IB method is explained in Eqs.(2) and (3), while the transfer process is a little different in the virtual boundary method. First, the velocity at the virtual boundary point $(X_{vb})$ is obtained through a linear interpolation using the velocity at the nearby mesh points $(x_{i})$.

$$U(X_{vb}) = \sum u(x_{i})D(x_{i} - X_{vb}), \quad 1 \leq i \leq N_{l},$$  

(27)

$$D(x) = \phi \left( \frac{x}{h} \right) \phi \left( \frac{x}{h} \right) \phi \left( \frac{x}{h} \right), \quad \phi(r) = \begin{cases} 1 - |r|, & |r| \leq 1 \\ 0, & \text{otherwise} \end{cases},$$  

(28)

where $\phi(r)$ is same with that in 2-point regularized delta function. After the feedback forcing is obtained, it is extrapolated back to the nearby mesh points,

$$f(x_{i}) = \frac{1}{N_{m}} \sum_{i=1}^{N_{m}} F(X_{ib})D(X_{ib} - x_{i}), \quad 1 \leq i \leq N_{l},$$  

(29)
where $N_m$ is the number of the virtual boundary points, $X_m$, that influence the mesh point $x_i$.

In Peskin’s IB method, transfer processes are a little different with the virtual boundary method. The velocity at the immersed boundary point, $X_m$ is approximated by regularized delta function, since Dirac delta function is not directly realizble in the discrete mesh system.

$$U(X_m) = \sum u(x_i)\delta_i(x_i - X_m)h^3, \quad 1 \leq l \leq N_l,$$  \hspace{1cm} (30)

where $\delta_i(x) = \frac{1}{h^3} \phi(x_i/h)\phi(x_i/h)\phi(x_i/h)$. After the feedback forcing is calculated, it is also spread back to the nearby mesh points though regularized delta function

$$f(x_i) = \sum \frac{N_{ij}}{h^3} \delta_i(x_i - X_{ij})\Delta V_{ij}, \quad 1 \leq l \leq N_l,$$  \hspace{1cm} (31)

The velocity approximation is almost same in two methods, but the forcing approximation is not. There are two significant differences in the forcing approximation. One difference is the number of Lagrangian points. The virtual boundary method uses many points, leading to high computational overheads, and the rule for determining the number of points is ambiguous. By comparison, Peskin’s IB method uses fewer points; in this method, it is recommended to use the number of Lagrangian points that makes the volume of each forcing point equivalent to a finite volume of the Eulerian grid ($\Delta L^3 = h^3$, $D$ is the number of space-dimensions) [6]. The second major difference is the conservation of quantities between the Lagrangian and the Eulerian domains. Peskin’s IB method uses the class of regularized delta functions as kernels in the transfer steps between Lagrangian and Eulerian locations. Accordingly, the total amount of quantities is not changed by the transfer step [8]. The virtual boundary method, by contrast, uses an average forcing and hence changes the total amount of quantities between the transfer steps. Fig.6 displays two-dimensional case for the two methods. We assume that all values of the forcing in the Lagrangian domain are $F$. Fig.6(a) illustrates Peskin’s IB method using 2-point regularized delta function. The Eulerian forcing $f$ can be obtained using,

$$f = \phi(1 - r_x)\phi(1 - r_y)F + \phi(1 - r_y)\phi(r_x)F$$

$$+ \phi(r_x)\phi(1 - r_y)F + \phi(r_x)\phi(r_y)F$$

$$= \phi(1 - r_x)\phi(1 - r_y)F + \phi(r_x)\phi(r_y)F$$

$$+ \phi(1 - r_y)\phi(1 - r_x)F + \phi(r_y)\phi(r_x)F$$

$$= \phi(1 - r)^2F + \phi(r)^2F \quad (\phi(1 - r) + \phi(r) = 1)$$  \hspace{1cm} (32)

Since Peskin’s IB method conserves forcing between Lagrangian and Eulerian domain, the drag and lift forces of a stationary cylinder can be obtained by integrating the Lagrangian forcings on the boundary $\partial S$ [11].

$$F_D = -\int f ds = -\int F ds, \quad F_L = -\int f ds = -\int F ds$$  \hspace{1cm} (33)

where $F_D$, $F_L$ are the drag and lift force, respectively.

Fig. 6(b) shows that the Eulerian forcing $f$ is not equal to $F$ by the virtual boundary method, i.e.,

$$f = \lim_{N_i \rightarrow N_l} \frac{1}{N_i} \sum_{j=1}^{N_i} F_{ij}$$

$$= \lim_{N_i \rightarrow N_l} \frac{1}{2N_l} \sum_{j=1}^{N_l} F_{ij}$$

$$= \frac{F}{2} \left[ 1 - \left( \frac{r_x}{N_x} \right)^2 + \left( \frac{r_y}{N_y} \right)^2 \right] \frac{1}{2}$$

$$= \frac{F}{4}$$

where $\Delta r_x = \frac{2}{N_x}$, $\Delta r_y = \frac{2}{N_y}$, and $N_x$ and $N_y$ are the number of Lagrangian points in the $x$-direction and $y$-direction, respectively. The virtual boundary method does not conserve the forcing in transfer step.

**RESULTS AND DISCUSSION**

**Stationary cylinder**

The flow past a stationary cylinder in a free-stream was simulated at two different Reynolds number (100 and 185) based on the free-stream velocity $u_*$ and the cylinder diameter. The first case was simulated to compare the present method with other methods such as Peskin’s IB method. The second case was examined to investigate the effects of feedback forcing gains ($\alpha$ and $\beta$) and types of delta functions.

**Stationary cylinder in a free-stream at Re=100**

We used a computational domain $0 \leq x, y \leq 8$ and a cylinder with diameter $d=0.30$ whose center is located at $(1.85, 4.0)$. A Dirichlet boundary condition ($u = 1$, $v = 0$) is used at the inflow and farfield boundaries, and convective boundary condition ($\partial u / \partial t + c \partial u / \partial x = 0$, where $c$ is the space-averaged stream-wise velocity) is used at the outflow boundary. Table 2 shows the present drag and lift coefficients $C_D, C_L$ as well as the Strouhal number defined from the oscillation frequency of the lift force. The drag and lift forces were obtained by integrating all the momentum forcing applied on the boundary as shown in Eq.(33).
The parameter such as the mesh width \( h \), time step \( \Delta t \), feedback forcing gain \( \alpha \) were selected to match the conditions of Lai & Peskin [11]. And a 4-point regularized delta function was employed [11]. To compare the present method with Peskin’s IB method, we used feedback forcing gains of \( \alpha = -4.8 \times 10^4 \) and \( \beta = 0 \). Table 2 indicates that the value of \( \alpha = -4.8 \times 10^4 \) used in the present method is large enough to obtain reliable results. By contrast, the results of Lai & Peskin [11] using \( \alpha = 4.8 \times 10^4 \) deviate somewhat from the other results, especially those obtained in the same study using \( \alpha = 4.8 \times 10^4 \). These findings are consistent with previous reports showing that compared to the value of \( -\Delta \alpha \) in the virtual boundary method, a larger value of the stiffness coefficient \( \kappa \) in Peskin’s IB method is required to ensure accurate results for a rigid boundary problem. Since we tested the stability region of feedback forcing gain \((\alpha, \beta)\) with the 4-point regularized delta function (see Fig.4), we used a computational time step of \( 1.2 \times 10^{-5} \) (\( -\Delta \alpha \) = 6.912) to be content with \( -\alpha \Delta t^2 < 8 \). As a consequence, the present results are in good agreement with those of Lai & Peskin [11], even though the computational time step of present method \( (\Delta t = 1.2 \times 10^{-5}) \) is about an order of magnitude larger than that of Lai & Peskin \( (\Delta t = 9.6 \times 10^{-4}) \). The maximum CFL number in the present simulations exceeded 1 due to the adoption of the feedback forcing scheme and the optimization of parameters by stability analysis. Uhlmann [6] also simulated the same problem with the same domain size. Uhlmann’s results showed some deviation even though the computational time-step and mesh size are smaller than those in the present work.

Table 2. Comparison of drag coefficient, lift coefficient and Strouhal number with those obtained in previous studies.

<table>
<thead>
<tr>
<th>Case</th>
<th>( \kappa ) or ( -\alpha )</th>
<th>( \Delta t )</th>
<th>( C_D )</th>
<th>( C_L )</th>
<th>( St )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td>( 4.8 \times 10^4 )</td>
<td>( 1.2 \times 10^{-2} )</td>
<td>1.44</td>
<td>0.35</td>
<td>0.168</td>
</tr>
<tr>
<td>Case2</td>
<td>( 4.8 \times 10^4 )</td>
<td>( 6.0 \times 10^{-3} )</td>
<td>1.44</td>
<td>0.35</td>
<td>0.168</td>
</tr>
<tr>
<td>Lai &amp; Peskin [11]</td>
<td>( 4.8 \times 10^4 )</td>
<td>( 1.8 \times 10^{-3} )</td>
<td>1.52</td>
<td>0.29</td>
<td>0.155</td>
</tr>
<tr>
<td>Lai &amp; Peskin [11]</td>
<td>( 9.6 \times 10^4 )</td>
<td>( 9.0 \times 10^{-4} )</td>
<td>1.45</td>
<td>0.33</td>
<td>0.165</td>
</tr>
<tr>
<td>Uhlmann[6]</td>
<td>( 3.0 \times 10^4 )</td>
<td>( 1.50 )</td>
<td>0.35</td>
<td>0.172</td>
<td></td>
</tr>
</tbody>
</table>

Stationary cylinder in a free-stream at Re=185

We consider the case of a large computational domain \(-50d \leq x, y \leq 50d \). The number of grid points in the streamwise \((x)\) and transverse \((y)\) directions was \( 352 \times 192 \), respectively. Thirty grid points in each direction are uniformly distributed inside the cylinder and stretched outside the cylinder. The Reynolds number \( Re_x = u_x d / \nu \) was set at 185 and the computational time step is \( \Delta t = 0.01 \), leading to a maximum CFL number of approximately 0.6. The boundary conditions at inflow, far field, outflow are same with those of the former simulation. Four different forcing gain values were simulated using 3-point regularized delta function. For quantitative comparison, a \( l_2 - \text{norm error} \) is defined as,

\[
l_2 - \text{norm error} = \frac{1}{N_t} \sqrt{\sum_{i=1}^{N_t} (U(X_i,t))^2},
\]

where \( N_t \) is the number of Lagrangian points and \( U(X_i,t) \) is the velocity at the \( k \)th Lagrangian point. The \( l_2 - \text{norm error} \) of the four cases, displayed in Fig.7, show that the error converges to a smaller value for the larger value of \( -\alpha \Delta t^2 \) and the error decays more rapidly for the larger value of \( -\beta \Delta t \). In stationary boundary problem, \( -\beta \Delta t \) influences only initial behavior of the error. Two cases using same \( -\alpha \Delta t^2 \) value with different \( -\beta \Delta t \) values, show the same level of the error, suggesting that \( -\alpha \Delta t^2 \) is more critical parameter for enforcing the no-slip condition [5]. However, \( -\beta \Delta t \) is also important to save total computing time. In case with \( -\beta \Delta t = 0 \), it takes long computing time for error to converge the small value. Therefore, both \( -\alpha \Delta t^2, -\beta \Delta t \) have important role in the accuracy and efficiency of the computation, respectively.

The behaviors of the error of the streamwise component of the virtual velocity are shown in Fig.8 for three types of regularized delta functions with same feedback forcing gain, \( -\alpha \Delta t^2 = 1.9, -\beta \Delta t = 0.9 \). The error of 4-point delta function is smaller than 2-point, 3-point delta function.
Inline oscillation of a circular cylinder

We simulated a periodic inline oscillation of a circular cylinder in fluid at rest. The Reynolds number is defined as $Re = u_\infty d / \nu$ based on the maximum velocity $u_\infty$ and the cylinder diameter $d$. The Keulegan-Carpenter number is defined as $KC = u_\infty / f d$ based on the frequency of the oscillation. The parameter set of the present investigation was $Re=100$ and $KC=5$, according to experimental and numerical results of Dütsch et al. [15]. We set the cylinder in time-periodic motion,

$$x_c(t) = -A_c \sin(2\pi f t),$$

where $x_c(t)$ is the position of the cylinder center and $A_c$ is the amplitude of the oscillation. The computational domain was $-50d \leq x, y \leq 50d$ and the number of grid points in the oscillatory ($x$) and transverse ($y$) directions was $416 \times 282$, respectively. Sixty grid points in each direction were uniformly distributed inside the cylinder and the remaining grid points were stretched outside the cylinder. Because the cylinder oscillates in $x$-direction, the uniformly distributed region had a rectangular form with longer length in $x$-direction. Neumann boundary conditions were used at all four boundaries. The computational time-step was $\Delta t = T / 720$ based on the period of the oscillation, leading to a maximum CFL number of approximately 0.6. The 4-point regularized delta function was selected and the feedback forcing gain was chosen as $-\alpha \Delta t^2 = 3.9$, $-\beta \Delta t = 1.9$ which is one of the largest cases in stability region of 4-point regularized delta function. In Fig.9, the time history of the drag coefficient in the oscillatory direction is in an excellent agreement with that shown in Dütsch et al. [15].

Transverse oscillation of a circular cylinder

We simulated a periodic transverse oscillation of a circular cylinder in a free-stream:

$$y_c(t) = A_y \cos(2\pi f_y t),$$

where $y_c$ is the position of the cylinder center and $A_y$ is the amplitude of the oscillation, and $f_y$ is the frequency of the oscillation. The detailed calculations were $Re=185$, $A_y / d = 0.2$ and $f_y / f_c = 0.8$. $f_c$ is the natural shedding frequency from the stationary cylinder. The computational domain was $0 \leq x, y \leq 8$ and $1024 \times 1024$ grid points were uniformly distributed in the oscillatory ($x$) and transverse ($y$) directions, respectively. The diameter was set to $d=0.30$ whose center is located at $(1.85, 4.0)$. A Dirichlet boundary condition ($u / u_\infty = 1, v = 0$) was used at the inflow and farfield boundaries, and convective boundary condition ($\partial u / \partial t + c \partial u / \partial x = 0$) was used at the outflow boundary.

![Fig.9](image)

Fig.9. Time history of the drag coefficient at $Re = 100$ and $KC = 5$

Fig.11. $L_2$-norm error of the virtual surface velocity in the streamwise direction normalized by the free-stream velocity $u_\infty$ for three different forcing gains

The behavior of the $L_2$-norm error of streamwise virtual surface velocity is shown in Fig.11 for three different forcing gains with the 3-point regularized delta function. The error converges to the smaller value for larger $-\alpha \Delta t^2$, as observed above for the stationary problem, and the error decays rapidly for larger $-\beta \Delta t$. However the error also converges to a smaller value for $-\beta \Delta t$. Accordingly, $-\alpha \Delta t^2, -\beta \Delta t$ should be as large as possible to decrease the error.

![Fig.12](image)

Fig.12. Time history of the drag coefficient (a) for different forcing gain using 3-point delta function (b) for different types of delta functions with $-\alpha \Delta t^2 = 0.4, -\beta \Delta t = 1$

Fig.12(a) shows time history of the drag coefficient for three different forcing gains using 3-point delta function. Non-growing oscillations becomes smaller, as $-\beta \Delta t$ is increased, but are increased for $-\alpha \Delta t^2$. The influence of $-\beta \Delta t$ is greater than that of $-\alpha \Delta t^2$, in that when $-\beta \Delta t$ is large the non-growing oscillations are not significant regardless of the value of $-\alpha \Delta t^2$. The time history of the drag coefficient is illustrated in Fig.12(b) for three types of delta functions with same forcing gains. As the number of points in the regularized delta function increases, the non-growing oscillations decrease. This suggests that 4-point regularized delta function and large forcing gains yield better results.
To compare the present results with previous ones, we performed simulations using a large computational domains $-50d \leq x, y \leq 50d$ with a grid size of $416 \times 282$. Sixty grid points were uniformly distributed in each direction inside the cylinder and the grids were stretched outside the cylinder. Several cases were simulated for the different oscillating frequencies in the range $0.8 \leq f_0 / f_s \leq 1.2$. The computational time-step was chosen as $\Delta t = T / 720$ based on the period of the oscillation, leading to a maximum CFL number in the range of $0.7 \sim 1.1$. The 4-point regularized delta function was employed and the feedback forcing gains were $-\alpha \Delta t^2 = 3.9$ and $-\beta \Delta t = 1.9$. The variations of the mean drag, rms drag and lift fluctuation coefficients as a function of $f_0 / f_s$ are presented in Fig.13(a). and the phase angles between the lift coefficient and the vertical position of the cylinder are shown in Fig.13(b). The present results agree well with those of Kim and Choi [16].

CONCLUSIONS

In the present study, we compared the virtual boundary method and Peskin’s IB method with a focus on two aspects: the feedback forcing scheme and the transfer process of quantities between Lagrangian and Eulerian locations. And we incorporate the feedback forcing scheme of the virtual boundary method with Peskin’s regularized delta function approach to improve performance. The resulting numerical method was implemented in a finite-difference and fractional-step context. We analyzed stability regimes of the feedback forcing gains in the proposed method for several types of delta function. The stability region of the 4-point regularized delta function was much wider than that of the 2-point delta function. The Effects of regularized delta functions and feedback forcing gains ($\alpha, \beta$) were also investigated. As a consequence, we can recommend the optimum region of feedback forcing gains which makes possible to use large CFL number and decrease $L_1 - \text{norm}$ error and non-growing oscillations. The proposed method was applied to the flow past a stationary cylinder, inline oscillation of a cylinder in a quiescent fluid and transverse oscillation of a cylinder in a free-stream at large CFL numbers ($0.6 \sim 1.3$). The present findings are in excellent agreement with previous numerical and experimental results.

REFERENCES