Consider the case of an elastic body or surface immersed in a fluid to create a fluid-structure interface.

\[ F(s,t) \text{ - elastic or other force exerted on the body by external sources or interactions within the body} \]

The body exerts this force back to the fluid.
Outside of the body, there is a fluid, described by, for example, incompressible Navier-Stokes:

\[ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mu \nabla^2 \mathbf{u} + \mathbf{f}(\mathbf{r}, t) \]

where

\[ \mathbf{f}(\mathbf{r}, t) = \int \mathbf{F}(s, t) \delta (\mathbf{r} - \mathbf{X}(s, t)) \, ds \]

force density on fluid

\[ \mathbf{X}(s, t) \rightarrow \text{Eulerian position of Lagrangian point} \]

force density on body
The no-slip condition at the surface (or in the interior) of the body is:

\[
\frac{\partial \mathbf{X}(s, t)}{\partial t} = \mathbf{v}(\mathbf{X}(s, t), t)
\]

\[
= \int \mathbf{v}(r, t) \delta(r - \mathbf{X}(s, t)) \, dr
\]

These equations are difficult to discretize, and in the IBM we take two steps to make this much easier:
We extend the fluid domain to cover the whole space, i.e. we have fluid even where the body is. This is OK because:

a) If there is a rigid body or a boundary impermeable to fluid \( \frac{\partial x}{\partial t} = u(x,t) \), it does not matter what the fictitious fluid behind the boundary is doing.

b) Often there is a thin boundary like a membrane surrounded by fluid on both sides.

c) Often the "elastic" body has fluid in it, say in biological tissues.
We discretize the Dirac delta functions with "discrete" delta functions $\delta_h$ and replace the integrals with simple quadratures:

$$f(\Gamma,t) = \sum_{\text{markers} \rightarrow S_i} F(S_i,t) \delta_h(\Gamma - X(S_i,t)) \Delta S_i$$

Let us denote

$$F(S_i,t) \Delta S_i = F_i \in \text{force on "marker" } i$$

$$X(S_i,t) = X_i \in \text{position of "marker" } i$$
Similarly,

\[ \frac{dx_i}{dt} = \int W(r,t) \delta(r-x_i) \, dr \approx \sum \frac{\partial u(r,t)}{\partial t} \delta_h(r-x_i) \Delta V \]

"INTERPOLATION" → regrid (MAC) cell volume
The final equations are:

\[
S \frac{du}{dt} + u GP_i = MLE_u + \sum F_i \delta h (r - x_i)
\]

Laplacian

\[
Du = 0
\]

everywhere (on the whole Eulerian grid \( r \))

\[
\frac{d x_i}{d t} = \sum_{r} \delta h (r - x_i) \Delta V
\]

where \( F_i = F_i (x) \) is given by the (elastic or other) model for the body
the physical picture behind the FBM now becomes clear:

→ Forces applied to the markers are spread back to the fluid by using the δh kernel (discrete Dirac delta) to convert force into force density.

→ The velocity of the fluid is interpolated (averaged!) at the location of the marker to find its velocity.

This works not just for elastic bodies but also rigid bodies, polymer chains, flagella, etc.
Temporal discretization
(second order, by Boyce Griffith)

1. Predictor step for markers:
\[ \tilde{X}^{n+\frac{1}{2}} = \tilde{X}^n + \frac{\Delta t}{2} \sum_{r} u_r^n \delta h (\tilde{r} - X^n) \Delta v \]

2. Evaluate forces
\[ F^{n+\frac{1}{2}} = F(\tilde{X}^{n+\frac{1}{2}}) \]

3. Solve NS equations on a staggered grid:
\[ \begin{cases} \frac{S}{\Delta t} \left( \frac{u^{n+1} - u^n}{\Delta t} + \mathbf{G} P^{n+\frac{1}{2}} \right) = \nabla \cdot (\mathbf{v} u^{n+1}) \mathbf{v} - \mathbf{v} \cdot \nabla \mathbf{v} \\ 0 u^{n+1} = 0 \end{cases} \]

Adams-Bashforth
4. Correct the positions of the markers:

\[ x^{n+1} = x^n + \Delta t \sum_{r} \left( \frac{v^n_r + v^{n+1}_r}{2} \right) \delta_h (r - x^{n+1/2}) \Delta V \]

This is second-order in time but the IB handling of the fluid-structure interaction is only first-order accurate in space because the velocity near the boundary is not smooth but has a jump in derivatives across a boundary (interface) → leads to "leaking" of fluid through the boundary.
What is $\delta h$ and how to choose it?

From equivalence of Eulerian and Lagrangian descriptions (assume "body" was actually a fluid parcel), we need:

(see review in Acta Numerica by Pestel)

$$\begin{align*}
\sum \delta h(r-x) &= \delta V^{-1} + \hat{X} \\
\sum (r-x) \delta h(r-x) &= 0 + \hat{X}
\end{align*}$$

(Equivalents of continuum)

$$\int \delta (r-x) \, dr = 1$$

$$\int (r-x) \delta (r-x) \, dr = 0$$
These give conservation of (mass), momentum, angular momentum, and energy.

A key property of the method used to show these is that the same $\delta h$ is used for both spreading and interpolation.

Consider energy conservation:

$$\sum \mathbf{f}(r,t) \cdot \mathbf{v}(r,t) \Delta V = \text{work done on fluid}$$

$$= \sum \sum F_i \cdot v_i \Delta h \left( \mathbf{x} - \mathbf{x}_i \right) \Delta V =$$

$$= \sum \sum F_i \cdot \frac{d\mathbf{x}_i}{dt} = \text{work done on markers}$$
Mathematically, the key feature is that interpolation and spreading are adjoint linear operators (mapping between Eulerian and Lagrangian spaces). Charlie Peskin proposed to use tensor-product discrete $\delta$ functions:

$$\delta_{h}(\mathbf{r}) = \frac{1}{h^3} \varphi \left( \frac{r_1}{h} \right) \varphi \left( \frac{r_2}{h} \right) \varphi \left( \frac{r_3}{h} \right)$$

where $h$ is grid spacing and $\varphi$ is a one-dimensional kernel function with the properties:
\[ \psi(r) \text{ is continuous} \]
\[ \psi(r) = 0 \text{ for } |r| \geq 2 \rightarrow \text{compact or local kernel} \]

\[ \sum \psi(r-j) = 0 \text{ for all } j \]
(translational invariance of first moment)

\[ \sum \psi(r-j) = 1 \text{ for all } j \]
(zeroth moment)

And a normalization condition:
\[ \sum \psi^2(r-j) = C = \text{const} \text{ for all } r \]
It turns out that from these types of conditions one can derive a unique function $\Psi(r)$. The smallest support one can get is a $3^+\text{pt}$ kernel, but better translational invariance may be obtained from the $4^+\text{pt}$-kernel.

A very simple but good approximation is the cosine kernel

$$\Psi(r) = \frac{1}{4} \left( 1 + \cos \left( \frac{\pi r}{2} \right) \right) \text{ for } |r| \leq 2$$

zero otherwise
There are very optimized ways to evaluate $y$ and do the spreading and interpolation efficiently.

But in 3D a 3pt kernel requires reading from memory $3^3 = 27$ points and a 4pt kernel 64 points.

The IBAMR (Immersed-Boundary Adaptive Mesh Refinement) code developed at Courant by Boyce is a state-of-the-art parallel implementation of the IBM Griffiths.