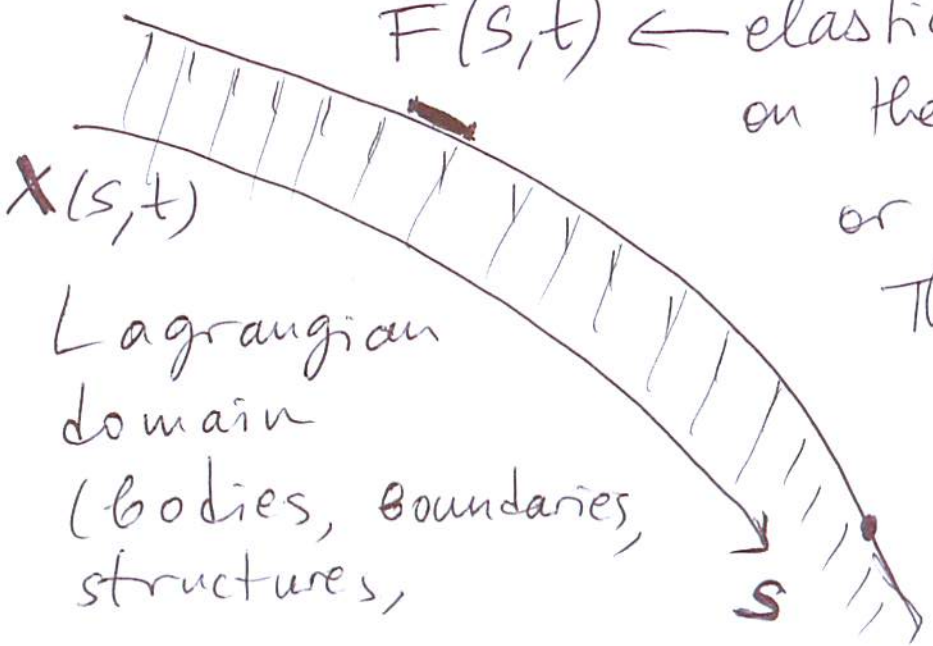
IMMERSED BOUNDARY METHODS

CFD SPRING 2013, A. DONEV

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



$F(s,t)$ ← 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:

$$\rho(\partial_t v + v \cdot \nabla v) + \nabla p = \mu \nabla^2 v + f(r,t)$$

where

$$f(r,t) = \int_S F(s,t)\, \delta(r - X(s,t))\, ds$$

$f(r,t)$: force density on fluid

$F(s,t)$: force density on body

$X(s,t)$ → Eulerian position of Lagrangian point s

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

$$\frac{\partial X(s,t)}{\partial t} = u(X(s,t),t)$$

$$= \int_\Gamma u(r,t)\,\delta(r - X(s,t))\,dr$$

These equations are difficult to discretize, and in the IBM we take two steps to make this much easier:

① 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 impenetrable to fluid $\left[\frac{\partial x}{\partial t} = v(x,t)\right]$, 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.

(2) We discretize the Dirac delta functions with "discrete" Delta functions δ_h and replace the integrals with simple quadratures:

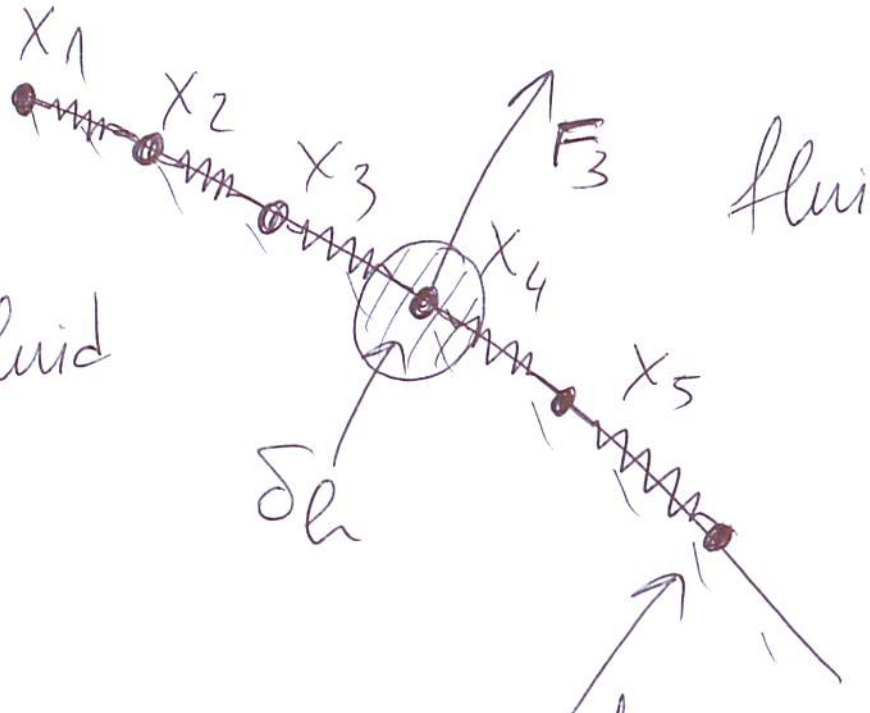
"SPREADING"

$$f(r,t) = \sum_{\text{MARKERS} \to s_i} F(s_i,t)\, \delta_h(r - X(s_i,t))\, \Delta s_i$$

Let us denote

$F(s_i,t)\,\Delta s_i \equiv F_i \leftarrow$ force on "marker" i

$X(s_i,t) = X_i \leftarrow$ position of marker i



Similarly,

we have to use δ_h here also to conserve energy

$$\frac{dX_i}{dt} = \int v(r,t)\,\delta_h(r-X_i)\,dr \approx \sum_{r \in \text{grid}} v(r,t)\,\delta_h(r-X_i)\,\Delta V$$

"INTERPOLATION" (MAC)

ΔV: cell volume

the final equations are:

$$\rho \frac{dv}{dt} + \nabla p = \mu L_v v + \sum_i^{\text{markers}} F_i \delta_h(r - X_i) - \rho (v \cdot \nabla) v + \text{other forces}$$

($\frac{dv}{dt}$: discrete; $L_v v$: Laplacian)

$$\nabla \cdot v = 0$$

everywhere (on the whole Eulerian grid r)

$$\frac{dX_i}{dt} = \sum_r^{\text{grid}} v_r \, \delta_h(r - X_i) \, \Delta V$$

where

$F_i \equiv F_i(X)$ is given by the (elastic or other) model for the body

the physical picture behind the IBM now becomes clear:

→ Forces applied to the markers are spread back to the fluid by using the δ_h 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:

$$\tilde{X}^{n+1/2} = X^n + \frac{\Delta t}{2} \sum_r u_r^n \delta_h(r - X^n) \Delta V$$

(2) Evaluate forces $F^{n+1/2} = F(\tilde{X}^{n+1/2})$

(3) Solve NS equations on a staggered grid:

$$\begin{cases} \rho \dfrac{u^{n+1} - u^n}{\Delta t} + G p^{n+1/2} = \mu L_u \left(\dfrac{u^n + u^{n+1}}{2} \right) - \rho (u \cdot \nabla u)^{n+1/2} \\ \qquad\qquad + \sum_i F_i \, \delta_h(r - X_i) \\ D u^{n+1} = 0 \end{cases}$$

↑ Adams-Bashforth

(4) Correct the positions of the markers:

$$X^{n+1} = X^n + \Delta t \sum_r \left(\frac{u_r^n + u_r^{n+1}}{2} \right) \delta_h (r - \tilde{X}^{n+1/2}) \Delta V$$

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 accross a boundary (interface)
→ leads to "leaking" of fluid through the boundary.

What is δ_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 Peskin)

$$\begin{cases} \sum_r \delta_h(r-x) = \Delta V^{-1} \quad \forall x \\ \sum_r (r-x)\,\delta_h(r-x) = 0 \quad \forall x \end{cases}$$

(equivalents of continuum

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

$$\int (r-x)\,\delta(r-x)\,dr = 0\Big)$$

These give conservation of (mass), momentum, angular momentum, and energy.

A key property of the method used to show these is that the same δ_h is used for <u>both</u> spreading and interpolation.

Consider energy conservation:

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

$$= \sum_r \sum_i F_i \cdot v(r,t) \delta_h(r - X_i) \Delta V =$$

$$= \sum_i F_i \cdot \frac{dX_i}{dt} = \text{work done on markers}$$

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 δ functions:

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

where h is grid spacing and φ is a one-dimensional kernel function with the properties:

$\varphi(r)$ is continuous

$\varphi(r) = 0$ for $|r| \geq 2$ or 1 → compact or local kernel

$$\sum_j (r-j)\,\varphi(r-j) = 0 \quad \text{for all } r$$

(j: integer)

(translational invariance of first moment)

$$\sum_j \varphi(r-j) = 1 \quad \text{for all } r$$

(zeroth moment)

And a normalization condition:

$$\sum_j \varphi^2(r-j) = C = \text{const} \quad \text{for } \forall r$$

It turns out that from these types of conditions one can derive a unique function $\varphi(r)$.

The smallest support one can get is a 3pt-kernel, but better translational invariance may be obtained from the 4pt-kernel.

A very simple but good approximation is the cosine kernel

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

zero otherwise

There are very optimized ways to evaluate φ 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 Griffith is a state-of-the-art parallel implementation of the IBM