GIZMO Implementation Details

Bert Vandenbroucke (bert.vandenbroucke@ugent.be), Mladen Ivkovic (mladen.ivkovic@epfl.ch)

1 GIZMO : variables and basic equations

This section is left unchanged and as the first in this document such that the comments in the code still correspond to the correct equations.

1.1 Variables

6 standard variables : $\vec{x}$, $\vec{v}$

5 conserved variables (C) : m, $\vec{p}$, E

5 primitive variables (X) : ρ, $\vec{w}$, P
It could be that $\vec{w} = \vec{v}$, but this is not generally true for all methods...

6 matrix elements + the volume (can be computed from m and ρ if this is accurate enough)

15 gradients (3 for every primitive variable)

In total: 37 quantities per particle (without the volume), 34 if the velocity is equal to the fluid velocity

1.2 Equations by Neighbour Loops

1.2.1 Loop 1: volumes (=densities) + matrix elements

Volumes are given by (combination of eqns. (7) and (27) from Hopkins) :

$$V_i = \frac{1}{\sum_j W(|\vec{x_i} - \vec{x_j}|, h_i)} \tag{1}$$

The 6 elements of the (symmetric) matrix E_i by (eqn. (14) in Hopkins without the normalization, because E_i is only used in combination with ψ_i (eqn. (12)), which contains the same normalization and cancels it out again) :

$$E_i^{\alpha\beta} = \sum_j (\vec{x_j} - \vec{x_i})^\alpha (\vec{x_j} - \vec{x_i})^\beta W(|\vec{x_i} - \vec{x_j}|, h_i) \tag{2}$$

The kernel functions in SWIFT do not include the normalization factor $\frac{1}{h_i^3}$, so this is added after the loop in the ghost.

1.2.2 Ghost 1: primitive variables + matrix inversion

Use the volume to convert conserved variables to primitive variables (eqn. (5), (7) and (9) in the AREPO paper) :

$$\rho = \frac{m}{V} \tag{3}$$

$$\vec{v} = \frac{\vec{p}}{m} \tag{4}$$

$$P = \frac{\gamma - 1}{V}\left(E - \frac{1}{2}\frac{|\vec{p}|^2}{m}\right) \tag{5}$$

Invert the matrix and call the inverse matrix B_i (eqn. (13) in Hopkins).

1.2.3 Loop 2: gradients

Calculate gradients for the primitive variables (eqn. (12) in Hopkins with eqn. (6) inserted and the normalization constant eliminated) :

$$\left(\vec{\nabla}X_i\right)^\alpha = \sum_j (X_j - X_i) \sum_\beta B_i^{\alpha\beta} (\vec{x_j} - \vec{x_i})^\beta W(|\vec{x_i} - \vec{x_j}|, h_i) \tag{6}$$

If you want to use a slope limiter of some sorts, then this would also be the time to collect the necessary data.

1.2.4 Ghost 2: nothing?

Perform the slope limiting if wanted.

1.2.5 Loop 3: hydro

For every neighbour j, calculate an interface area (combination of eqn. (12) in Hopkins and the definition of $\vec{A_{ij}}$, given in between eqns. (18) and (19))

$$\left(\vec{A_{ij}}\right)^\alpha = V_i \sum_\beta B_i^{\alpha\beta} (\vec{x_j} - \vec{x_i})^\beta W(|\vec{x_i} - \vec{x_j}|, h_i) + V_j \sum_\beta B_j^{\alpha\beta} (\vec{x_j} - \vec{x_i})^\beta W(|\vec{x_i} - \vec{x_j}|, h_j) \tag{7}$$

Calculate a position for the interface (eqn. (20) in Hopkins) :

$$\vec{x_{ij}} = \vec{x_i} + \frac{h_i}{h_i + h_j} (\vec{x_j} - \vec{x_i}) \tag{8}$$

and a velocity (eqn. (21) in Hopkins) :

$$\vec{v_{ij}} = \vec{v_i} + (\vec{v_j} - \vec{v_i}) \left[\frac{(\vec{x_{ij}} - \vec{x_i}).(\vec{x_j} - \vec{x_i})}{|\vec{x_i} - \vec{x_j}|^2} \right] \tag{9}$$

Use the velocity to boost the primitive variables X_k $(k = i, j)$ to the rest frame of the interface (X'_k). This comes down to applying a correction to the fluid velocities.

Use the gradients to interpolate the primitive variables from positions $\vec{x_i}$ and $\vec{x_j}$ to $\vec{x_{ij}}$ and predict them forward in time by half a time step to obtain second order accuracy in time (eqn. (A3) in Hopkins) :

$$X_k'' = X_k' + \left(\vec{\nabla} X\right)_k . (\vec{x_{ij}} - \vec{x_k}) + \frac{\Delta t}{2} \frac{\partial X_k'}{\partial t}. \tag{10}$$

The time derivatives are given by (eqn. (A4) in Hopkins)

$$\frac{\partial X_k'}{\partial t} = - \begin{pmatrix} \vec{w'} & \rho' & 0 \\ 0 & \vec{w'} & 1/\rho' \\ 0 & \gamma P' & \vec{w'} \end{pmatrix} \vec{\nabla} X_k, \tag{11}$$

or, for example :

$$\frac{\rho_k^{n+1/2} - \rho_k^n}{\Delta t/2} = - \left(\vec{w_k} - \vec{v_{ij}}^n\right) . \vec{\nabla} \rho_k^n - \rho_k^n \vec{\nabla} . \vec{w_k}^n \tag{12}$$

We then feed the X_k'' to a 1D Riemann solver, that gives us a vector of fluxes $\vec{F_{ij}''}$, which we deboost to a static frame of reference ($\vec{F_{ij}}$). To account for the arbitrary orientation of the interface, we also feed the normal vector to the interface to the Riemann solver. The Riemann solver internally uses the fluid velocity along the interface normal to solve an effective 1D Riemann problem. The velocity solution along the interface normal is also internally added to the velocities (see GIZMO source code). Mathematically, this is the same as first rotating the velocities to a frame aligned with the interface and then rotating the solution back to the original frame (e.g. AREPO paper), but it is computationally cheaper and causes less round off error in the solution.

Given the solution vector $X_{\rm half}$ returned by the Riemann problem, the fluxes are given by (eqns. (1)-(3) in Hopkins)

$$\vec{F_\rho} = \rho_{\rm half} \left(\vec{w}_{\rm half} - \vec{v}_{ij}\right) \tag{13}$$

$$\vec{F}_{w_k} = \rho_{\rm half} w_{k,\rm half} \left(\vec{w}_{\rm half} - \vec{v}_{ij}\right) + P_{\rm half} \vec{n}_k \tag{14}$$

$$\vec{F}_P = \rho_{\rm half} e_{\rm half} \left(\vec{w}_{\rm half} - \vec{v}_{ij}\right) + P_{\rm half} \vec{w}_{\rm half}, \tag{15}$$

with $\vec{n}_k$ the unit vector along the coordinate axes ($k = x, y, z$) and $e = \frac{P_{\rm half}}{(\gamma - 1)\rho_{\rm half}} + \frac{1}{2} \vec{w}_{\rm half}^2$.

Finally, we use the fluxes to update the conserved variables (eqn. (23) in Hopkins) :

$$\Delta C_k = -\Delta t \sum_j \vec{F_{ij}} . \vec{A_{ij}}. \tag{16}$$

2 Equations

2.1 Partition of Unity and Related Quantities

The partition of unity is defined as:

$$\psi_i(\mathbf{x}) = \frac{1}{\omega(\mathbf{x})} W(\mathbf{x} - \mathbf{x}_i, h(\mathbf{x})) \tag{17}$$

$$\omega(\mathbf{x}) = \sum_j W(\mathbf{x} - \mathbf{x}_j, h(\mathbf{x})) \tag{18}$$

where $h(\mathbf{x})$ is some "kernel size" and $\omega(\mathbf{x})$ is used to normalise the volume partition at any point $\mathbf{x}$. We use the compact support radius H of the kernels as h.

It can be shown that (provided $W(\mathbf{x})$ is normalized, i.e. $\int_V W(\mathbf{x})\mathrm{d}V = 1$):

$$V_i = \int_V \psi_i(\mathbf{x})\mathrm{d}V = \frac{1}{\omega(\mathbf{x}_i)} \tag{19}$$

$$V = \sum_i V_i \tag{20}$$

$$\int_V f(\mathbf{x})\mathrm{d}V = \sum_i f(\mathbf{x}_i) V_i + \mathcal{O}(h^2) \tag{21}$$

Eq. (21) can be derived by Taylor-expanding $f(\mathbf{x})$ around $\mathbf{x}_i$ and integrating the first order term by parts to show that it is zero.

2.2 Meshless Hydrodynamics à la Hopkins

Following Hopkins 2015, we arrive at the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_{k,i}) + \sum_j \mathbf{F}_{k,ij} \cdot \mathbf{A}_{ij} = 0 \tag{22}$$

with

$$\mathbf{A}_{ij}^{\alpha} = V_i \tilde{\boldsymbol{\psi}}_j^{\alpha}(\mathbf{x}_i) - V_j \tilde{\boldsymbol{\psi}}_i^{\alpha}(\mathbf{x}_j) \tag{23}$$

for every component k of the Euler equations and every gradient component α

The $\tilde{\boldsymbol{\psi}}(\mathbf{x})$ come from the $\mathcal{O}(h^2)$ accurate discrete gradient expression from Lanson and Vila 2008:

$$\frac{\partial}{\partial x_\alpha} f(\mathbf{x})\Big|_{\mathbf{x}_i} = \sum_j \left(f(\mathbf{x}_j) - f(\mathbf{x}_i)\right) \tilde{\boldsymbol{\psi}}_j^{\alpha}(\mathbf{x}_i) \tag{24}$$

$$\tilde{\boldsymbol{\psi}}_j^{\alpha}(\mathbf{x}_i) = \sum_{\beta=1}^{\beta=\nu} \mathbf{B}_i^{\alpha\beta}(\mathbf{x}_j - \mathbf{x}_i)^{\beta} \psi_j(\mathbf{x}_i) \tag{25}$$

$$\mathbf{B}_i = \mathbf{E_i}^{-1} \tag{26}$$

$$\mathbf{E}_i^{\alpha\beta} = \sum_j (\mathbf{x}_j - \mathbf{x}_i)^{\alpha}(\mathbf{x}_j - \mathbf{x}_i)^{\beta} \psi_j(\mathbf{x}_i) \tag{27}$$

where α and β again represent the coordinate components for ν dimensions.

2.3 Meshless Hydrodynamics à la Ivanova

Following Ivanova et al. 2013, we arrive at the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_i \mathbf{U}_{k,i}) + \sum_j \mathbf{F}_{k,ij} \cdot \mathbf{A}_{ij} = 0 \tag{28}$$

In the paper, no discrete expression for $\mathbf{A}_{ij}$is given, instead:

$$\mathbf{A}_{ij} = \int_V \left[\psi_j(\mathbf{x}) \nabla \psi_i(\mathbf{x}) - \psi_i(\mathbf{x}) \nabla \psi_j(\mathbf{x})\right] \mathrm{d}V \tag{29}$$

By Taylor-expanding the gradients of ψ the volume integral can be discretised as:

$$\mathbf{A}_{ij}^{\alpha} = V_i \nabla^{\alpha} \psi_j(\mathbf{x}_i) - V_j \nabla^{\alpha} \psi_i(\mathbf{x}_j) + \mathcal{O}(h^2) \tag{30}$$

This time we get analytical gradients of ψ instead of $\tilde{\boldsymbol{\psi}}$. Note that the indices i and j in these equations are switched compared to the one given in Ivanova et al. 2013 so that the evolution equations for Hopkins and Ivanova versions are identical.

3 Explicit Computations

Main purpose of this section is to have the explicit computations written down clearly somewhere. Implicitly swallowing indices in the formulae can make programming life a nightmare, so here we go.

3.1 Normalization

To compute the normalisations 18, we need to sum over all neighbouring particles and sum the kernels correctly. To evaluate the kernels, we use the `kernel_deval(xij, wij, wij_dx)` function in SWIFT.

If a kernel is defined as

$$W_i(\mathbf{x}) = W(\mathbf{x} - \mathbf{x}_i, h(\mathbf{x})) = \frac{1}{h(\mathbf{x})^\nu} w\left(\frac{|\mathbf{x} - \mathbf{x}_i|}{h(\mathbf{x})}\right)$$

then `kernel_deval` computes

$$\begin{aligned} \texttt{wij} &= w(\texttt{xij}) \\ \text{and} \quad \texttt{wij\_dx} &= \frac{\partial w(r)}{\partial r}\Big|_{r=\texttt{xij}} \\ \text{with} \quad \texttt{xij} &= \frac{|\mathbf{x}_i - \mathbf{x}_j|}{h(\mathbf{x}_i)} \end{aligned}$$

So for a specific particle position i, we need to compute

$$\begin{aligned} \omega(\mathbf{x}_i) &= \sum_j W(\mathbf{x}_i - \mathbf{x}_j, h(\mathbf{x}_i)) \\ &= \sum_j \frac{1}{h(\mathbf{x}_i)^\nu} w\left(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{h(\mathbf{x}_i)}\right) \\ &= \sum_j \frac{1}{h_i^\nu} \texttt{wij} \end{aligned}$$

with $h_i = h(\mathbf{x}_i)$.

3.2 Analytical gradients of $\psi(\mathbf{x})$

For the Ivanova et al. 2013 expression of the effective surfaces $\mathbf{A}_{ij}$, we need analytical gradients in Cartesian coordinates of $\psi_i(\mathbf{x}_j)$.

From eq. 17 we have that

$$\psi_j(\mathbf{x}_i) = \frac{W(\mathbf{x}_i - \mathbf{x}_j, h(\mathbf{x}_i))}{\omega(\mathbf{x}_i)}$$

Let $r_{ij} \equiv |\mathbf{x}_i - \mathbf{x}_j|$ and $q_{ij} \equiv \frac{r_{ij}}{h_i}$. Then

$$\begin{aligned}
\frac{\partial}{\partial x}\psi_j(\mathbf{x}_i) &= \frac{\partial}{\partial x}\frac{W(\mathbf{x}_i - \mathbf{x}_j, h(\mathbf{x}_i))}{\omega(\mathbf{x}_i)} = \frac{\partial}{\partial x}\frac{W(r_{ij}, h_i)}{\omega(\mathbf{x}_i)} \\
&= \frac{\frac{\partial W}{\partial x}(r_{ij}, h_i)\ \omega(\mathbf{x}_i) - W(r_{ij}, h_i)\frac{\partial \omega}{\partial x}(\mathbf{x}_i)}{\omega(\mathbf{x}_i)^2} \\
&= \frac{1}{\omega(\mathbf{x}_i)}\frac{\partial W}{\partial x}(r_{ij}, h_i) - \frac{1}{\omega(\mathbf{x}_i)^2}W(r_{ij}, h_i)\frac{\partial}{\partial x}\sum_k W(r_{ik}, h_i) \\
&= \frac{1}{\omega(\mathbf{x}_i)}\frac{\partial W}{\partial x}(r_{ij}, h_i) - \frac{1}{\omega(\mathbf{x}_i)^2}W(r_{ij}, h_i)\sum_k \frac{\partial W}{\partial x}(r_{ik}, h_i)
\end{aligned} \tag{31}$$

If a kernel[1] is defined as

$$W_j(\mathbf{x}_i) = W(\mathbf{x}_i - \mathbf{x}_j, h_i)) = \frac{1}{h_i^\nu}w\left(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{h_i}\right) = \frac{1}{h_i^\nu}w(q_{ij})$$

and we assume that the smoothing length h_i is treated as constant at this point, then the gradient of the kernel is given by

[1] Helpful way to think of the indices: $W_j(\mathbf{x}_i) = W(|\mathbf{x}_j - \mathbf{x}_i|, h_i)$ is the kernel value of particle i at position $\mathbf{x}_i$ evaluated at the position $\mathbf{x}_j$. Personally I would've used the indices the other way around for simplified thinking, but I'm going to stick to this notation because Hopkins also uses it.

$$\begin{aligned}\frac{\partial}{\partial x} W_j(\mathbf{x}_i) &= \frac{\partial}{\partial x}\left(\frac{1}{h_i^\nu} w(q_{ij})\right)\\ &= \frac{1}{h_i^\nu}\frac{\partial w(q_{ij})}{\partial q_{ij}}\frac{\partial q_{ij}(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial x} \qquad (32)\end{aligned}$$

We now use

$$\begin{aligned}\frac{\partial q_{ij}(r_{ij})}{\partial r_{ij}} &= \frac{\partial}{\partial r_{ij}}\frac{r_{ij}}{h_i} = \frac{1}{h_i} \qquad (33)\\ \frac{\partial r_{ij}}{\partial x} &= \frac{\partial}{\partial x}\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^2} = \frac{1}{2}\frac{1}{\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^2}} \cdot 2(\mathbf{x}_i - \mathbf{x}_j)\\ &= \frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}} \qquad (34)\end{aligned}$$

To be perfectly clear, we should in fact write

$$r_j(\mathbf{x}) \equiv |\mathbf{x} - \mathbf{x}_j|$$

which again leads to

$$\begin{aligned}\frac{\partial r_{ij}}{\partial x} &= \frac{\partial r_j(\mathbf{x}_i)}{\partial x} = \left.\frac{\partial r_j(\mathbf{x})}{\partial x}\right|_{\mathbf{x}=\mathbf{x}_i}\\ &= \frac{\partial}{\partial x}\left.\sqrt{(\mathbf{x} - \mathbf{x}_j)^2}\right|_{\mathbf{x}=\mathbf{x}_i} = \frac{1}{2}\frac{1}{\sqrt{(\mathbf{x} - \mathbf{x}_j)^2}} \cdot \left. 2(\mathbf{x} - \mathbf{x}_j)\right|_{\mathbf{x}=\mathbf{x}_i}\\ &= \left.\frac{\mathbf{x} - \mathbf{x}_j}{r_j(\mathbf{x})}\right|_{\mathbf{x}=\mathbf{x}_i} = \frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}}\end{aligned}$$

Inserting expressions 33 and 34 in 32, we obtain

$$\frac{\partial}{\partial x} W_j(\mathbf{x}_i) = \frac{1}{h_i^{\nu+1}}\frac{\partial w(q_{ij})}{\partial q_{ij}}\frac{\mathbf{x}_i - \mathbf{x}_j}{r_{ij}} \qquad (35)$$

$\frac{\partial w(q_{ij})}{\partial q_{ij}}$ is given by `wij_dx` of `kernel_deval`.

Finally, inserting 35 in 31 we get

$$\begin{aligned}\frac{\partial}{\partial x}\psi_j(\mathbf{x}_i) &= \frac{1}{\omega(\mathbf{x}_i)}\frac{1}{h_i^{\nu+1}}\frac{\partial}{\partial r}w(r_{ij},h_i)\frac{\mathbf{x}_i-\mathbf{x}_j}{r_{ij}} - \frac{1}{\omega(\mathbf{x}_i)^2}W(r_{ij},h_i)\sum_k \frac{1}{h_i^{\nu+1}}\frac{\partial}{\partial r}w(r_{ik},h_i)\frac{\mathbf{x}_i-\mathbf{x}_k}{r_{ik}} \\ &= \frac{1}{\omega(\mathbf{x}_i)}\frac{1}{h_i^{\nu+1}}\texttt{wij\_dx}\frac{\mathbf{x}_i-\mathbf{x}_j}{r_{ij}} - \frac{1}{\omega(\mathbf{x}_i)^2}W(r_{ij},h_i)\frac{1}{h_i^{\nu+1}}\sum_k \texttt{wik\_dx}\frac{\mathbf{x}_i-\mathbf{x}_k}{r_{ik}}\end{aligned} \tag{36}$$

The definition of r_{ij} requires a bit more discussion. Since kernels used in hydrodynamics (at least in those methods currently implemented in SWIFT) are usually taken to be spherically symmetric, we might as well have defined

$$r'_{ij} = |\mathbf{x}_j - \mathbf{x}_i|$$

which would leave the evaluation of the kernels invariant $[r'_{ij} = r_{ij}]$, but the gradients would have the opposite direction:

$$\frac{\partial r'_{ij}}{\partial x} = \frac{\mathbf{x}_j-\mathbf{x}_i}{r'_{ij}} = -\frac{\mathbf{x}_i-\mathbf{x}_j}{r_{ij}} = -\frac{\partial r_{ij}}{\partial x}$$

So which definition should we take?

Consider a one-dimensional case where we choose two particles i and j such that $x_j > x_i$ and $q_{ij} = |x_j - x_i|/h_i < H$, where H is the compact support radius of the kernel of choice. Because we're considering a one-dimensional case with $x_j > x_i$, we can now perform a simple translation such that particle i is at the origin, i.e. $x'_i = 0$ and $x'_j = x_j - x_i = |x_j - x_i| = r_{ij}$. In this scenario, the gradient in Cartesian coordinates and in spherical coordinates must be the same:

$$\begin{aligned} & \frac{\partial}{\partial x'}W(|x'_i - x'|,h_i)\Big|_{x'=x'_j} = \frac{\partial}{\partial r_{ij}}W(r_{ij},h_i) \\ \Rightarrow\quad & \frac{1}{h_i^\nu}\frac{\partial w(q'_{ij})}{\partial q'_{ij}}\frac{\partial q'_{ij}(r'_{ij})}{\partial r'_{ij}}\frac{\partial r'_i(x')}{\partial x'}\Big|_{x'=x'_j} = \frac{1}{h_i^\nu}\frac{\partial w(q_{ij})}{\partial q_{ij}}\frac{\partial q_{ij}(r_{ij})}{\partial r_{ij}}\end{aligned} \tag{37}$$

We have the trivial case where

$$\begin{aligned}
r'_{ij} &= |x'_i - x'_j| = |x_i - x_j| = r_{ij} \\
q'_{ij} &= r'_{ij}/h_i = q_{ij} \\
\Rightarrow \quad \frac{\partial w(q'_{ij})}{\partial q'_{ij}} &= \frac{\partial w(q_{ij})}{\partial q_{ij}}, \quad \frac{\partial q'_{ij}(r'_{ij})}{\partial r'_{ij}} = \frac{\partial q_{ij}(r_{ij})}{\partial r_{ij}}
\end{aligned}$$

giving us the condition from 37:

$$\frac{\partial r'_i(x')}{\partial x'}\Big|_{x'=x'_j} = 1 = \frac{\partial r_i(x)}{\partial x} \tag{38}$$

this is satisfied for

$$\begin{aligned}
&r_j(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_j| \\
&\Rightarrow r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|, \quad \text{not} \quad r_{ij} = |\mathbf{x}_j - \mathbf{x}_i|
\end{aligned}$$

References

Hopkins, Philip F. (June 2015). “GIZMO: A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods”. In: *Monthly Notices of the Royal Astronomical Society* 450.1, pp. 53–110. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stv195. arXiv: 1409.7395.

Ivanova, N. et al. (Nov. 2013). “Common Envelope Evolution: Where We Stand and How We Can Move Forward”. In: *The Astronomy and Astrophysics Review* 21.1. ISSN: 0935-4956, 1432-0754. DOI: 10.1007/s00159-013-0059-2. arXiv: 1209.4302.

Lanson, Nathalie and Jean-Paul Vila (Apr. 2008). “Renormalized Meshfree Schemes I: Consistency, Stability, and Hybrid Methods for Conservation Laws”. In: *SIAM J. Numer. Anal.* 46.4, pp. 1912–1934. ISSN: 0036-1429. DOI: 10.1137/S0036142903427718.