EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN — A&B DEPARTMENT

ARMANDO, a SPH code for CERN
Some theory, a short tutorial, the code description and some examples

Luca Massidda (CRS4)

Abstract

The Smoothed Particle Hydrodynamics methodologies may be a useful numerical tool for the simulation of particle beam interaction with liquid targets and obstacles. ARMANDO code is a state of the art SPH code interfaced with FLUKA and capable to solve these problems. This report presents the basic theoretical elements behind the method, describes the most important aspects of the implementation and shows some simple examples.

Geneva, Switzerland
July 2008

Contents

Overview

The Smoothed Particle Hydrodynamics is a numerical method that allows to solve highly nonlinear mechanical dynamics problems. The method has been originally developed for astrophysics and has been modified and used to simulate the dynamics of gases, liquids and also solids.

The method is particularly useful when strongly nonlinear phenomena has to be simulated, putting it in a different application field with respect to the traditional and well known Finite Element, Finite Difference and Finite Volume methods. It is in fact a mesh-less method, meaning that the computational domain do not need to be defined by some kind of structured data, and the points in which the field quantities are discretely defined do not need a pre-definite and fixed interconnection, the mesh. Here the space is defined by a finite set of discrete particles, of given mass, position, velocity etc. that interact with each other, and the interactions with particles may vary at each time step of the simulation. Moreover being an explicit method it is very flexible for the definition of the material behavior, and it appears to be well suited for the introduction of new physics in the modeling.

The method therefore allows simulating highly non-linear phenomena, with phase changes, jets, wave breaking and explosions to name a few, all the phenomena for which the common numerical methods are not well suited. The method appears to be adapt for the study of beam interaction problems, especially in the case of high power densities, that may cause a phase change or the rupture of the material, or for all the cases in which liquids are involved.

The ARMANDO code is an implementation of the SPH method, devoted in particular to the study of beam interaction with liquids. The best state of the art methodologies have been implemented. It is an open software written in FORTRAN90 that may be easily modified and extended. Moreover a simple graphical interface has been prepared that eases the normal tasks for the user of such a tool. If also we consider that public domain visualizers for mesh-less methods are already available, the designer can complete a numerical simulation of such problems using only in-house or free software.

This report describes the very first version of the software. First the theory behind the method is shortly described; then some details on the most tricky or less than standard characteristics of the code are given; then follows the description of the simulation procedure, with also an overview on the interface. The description of the code details follows, with the most important procedures and variables. Finally some example problems are described.

The code is based on the book "SPH, a meshfree particle method" by G. R. Liu and M. B. Liu, that has been tested for several standard problems,

Future work is anyway envisaged, starting from an important and interesting experimental validation of the results given by the code thanks to past experimental data available at CERN. Several improvements are possible, such as the development of a parallel version for PC clusters, the introduction of elastic materials, the MHD interaction and a bidirectional coupling with FLUKA to name a few.

Theory

The SPH method

Any continuous function $f(x)$ defined over a domain Ω can be expressed as the convolution integral of the function itself and a delta function:

$$f(x) = \int_\Omega f(x') \cdot \delta(x - x', h) dx'$$

the fundamental principle in Smoothed Particle Hydrodynamics (SPH) is to approximate the delta function with a kernel function any function $W(x-x')$ of limited support of size h:

$$f(x) \approx \int_\Omega f(x') W(x - x', h) dx'$$

h is the smoothing length of the kernel. After some minor manipulation the following expression is obtained:

$$f(x) \approx \int_\Omega \frac{f(x')}{\rho(x')} W(x - x', h) \rho(x') dx'$$

The computational domain is divided in a finite set of particle N of finite mass m_j and density ρ_j, for that the total mass of the particles is equal to the mass of the fluid in the domain, and in discrete notation, the previous approximation leads to the following expression of the value of the function at the particle position $f_i = f(x_i)$:

$$f_i = \sum_j m_j \frac{f_j}{\rho_j} W_{ij}$$

where the sum is carried out on all the particles that may interact with the particle i and that are within the region of the compact support defined by the kernel function. $W_{ij} = W(x_i - x_j, h)$ is the kernel function for the particle pair.

Similarly the derivative of any given function $\partial f(x)/\partial x$, is obtained by applying the integral interpolant on the gradient, and then using the integration by parts and neglecting the residual boundary terms, and obtaining the following form:

$$\frac{\partial f(x)}{\partial x} \approx \int_\Omega f(x) \frac{\partial W(x - x', h)}{\partial x} dx'$$

The discrete expression for the gradient of a function on a particle position is:

$$\left. \frac{\partial f}{\partial x} \right|_i = \sum_j m_j \frac{f_j}{\rho_j} \left. \frac{\partial W_{ij}}{\partial x} \right|_i$$

where the gradient of $W(x_i - x_j, h)$ is calculated with respect to the position of particle i.

In common practice two alternative symmetric forms of the expression above are used:

$$\frac{\partial f}{\partial x}\bigg|_i = \sum_j m_j \left(\frac{f_i}{\rho_i^2} + \frac{f_j}{\rho_j^2}\right) \frac{\partial W_{ij}}{\partial x}\bigg|_i$$

$$\frac{\partial f}{\partial x}\bigg|_i = \sum_j m_j \left(\frac{f_i + f_j}{\rho_i \rho_j}\right) \frac{\partial W_{ij}}{\partial x}\bigg|_i$$

The smoothing kernel

The kernel function is defined in such a way that it mimics the Dirac function as h approaches zero, so that its integral over the domain Ω is equal to unity and so that $W(x\text{-}x', h) = W(x'\text{-}x, h)$.

$$\int_\Omega W(x - x', h)dx' = 1$$

$$\lim_{h \to 0} W(x - x', h) = \delta(x - x')$$

The kernels are function of the smoothing length h and of the distance between the particles $r = \dfrac{|x - x'|}{h}$. The size of the domain of influence around a particle is proportional to the smoothing length, its size may be expressed as kh where k is lower than 4 and depends on the specific kernel adopted.
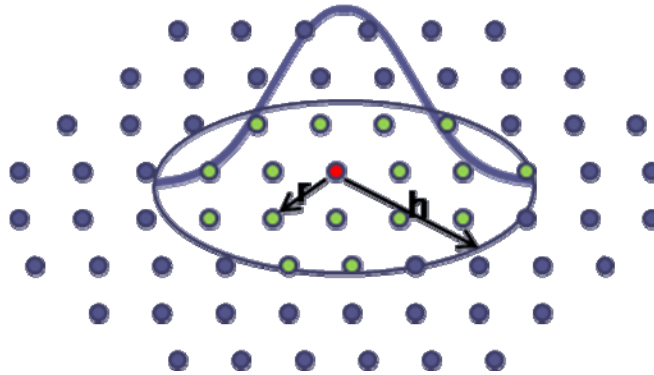

Figure 1: smoothing kernel function on a 2D domain

Several kernel functions have been proposed in the open literature; among the others the following three kernels are coded:

- Cubic spline (Monaghan and Lattanzio 1985)
- Gaussian function (Gingold and Managhan 1977)
- Quintic function (Morris 1996)

Cubic spline

$$W(r, h) = \alpha_D \begin{cases} \dfrac{2}{3} - r^2 + \dfrac{1}{2}r^3, & 0 \le r < 1 \\[2mm] \dfrac{1}{6}(2 - r)^3, & 1 \le r < 2 \\[2mm] 0, & r \ge 2 \end{cases}$$

With $r = \dfrac{|x - x'|}{h}$ and $a_D = 1/h$, $15/7h^2$ and $3/2\pi h^3$ for one, two and three-dimensional space.

Gaussian function

$$W(r, h) = \alpha_D e^{-r^2}$$

With and $a_D = 1/\pi^{1/2}h$, $1/\pi h^2$ and $1/\pi^{3/2}h^3$ for one, two and three-dimensional space.

Quintic function

$$W(r, h) = \alpha_D \begin{cases} (3 - r)^5 - 6(2 - r)^5 + 15(1 - r)^5, & 0 \leq r < 1 \\ (3 - r)^5 - 6(2 - r)^5, & 1 \leq r < 2 \\ (3 - r)^5, & 2 \leq r < 3 \\ 0, & r \geq 3 \end{cases}$$

With $r = \dfrac{|x - x'|}{h}$ and $a_D = 120/h$, $7/478\pi h^2$ and $3/359\pi h^3$ for one, two and three-dimensional space.

The Navier-Stokes equations

The Navier-Stokes are the governing equations describing the fluid flow in an eulerian or lagrangian approach. It is a set of partial differential equations that state the conservation of mass, momentum, and energy. The Lagrangian reference system is adopted, conforming to the SPH approach, and the total time derivatives are taken in the moving Lagrangian frame. The coordinate directions are denoted with the greek superscripts α and β, and the repeated index summation convention is used.

The equations are the following:

- continuity equation

$$\frac{D\rho}{Dt} = -\rho \frac{\partial v^\alpha}{\partial x^\beta}$$

- momentum equation

$$\frac{Dv^\alpha}{Dt} = \frac{1}{\rho} \frac{\partial \sigma^{\alpha\beta}}{\partial x^\beta} + g^\alpha$$

- energy equation

$$\frac{De}{Dt} = \frac{\sigma^{\alpha\beta}}{\rho} \frac{\partial v^\alpha}{\partial x^\beta} + q$$

in the equations above ρ is the density, $\boldsymbol{v}$ is the velocity vector, $\boldsymbol{g}$ is the external acceleration vector, e is the internal specific energy, and q the external power supplied and σ is the stress tensor. The latter can be expressed as follows by dividing the isotropic pressure p and the viscous tangential stresses τ

$$\sigma^{\alpha\beta} = -p\delta^{\alpha\beta} + \tau^{\alpha\beta}$$

For the Newtonian fluids we have:

$$\tau^{\alpha\beta} = \mu\varepsilon^{\alpha\beta}$$

Where ε is the deformation tensor:

$$\varepsilon^{\alpha\beta} = \frac{\partial v^\beta}{\partial x^\alpha} + \frac{\partial v^\beta}{\partial x^\alpha} - \frac{2}{3}(\nabla \cdot v)\delta^{\alpha\beta}$$

The following expression of the energy equation is obtained:

$$\frac{De}{Dt} = -\frac{p}{\rho}\frac{\partial v^\beta}{\partial x^\beta} + \frac{\mu}{2\rho}\varepsilon^{\alpha\beta}\varepsilon^{\alpha\beta} + q$$

The NS equations in SPH formulation

Continuity equation

There are two approaches to calculate the density approximation in the SPH method, the *summation density* and the *continuity density*.

The first consist in directly applying the SPH approximation to the density:

$$\rho_i = \sum_j m_j W_{ij}$$

the expression above is simple and direct, maybe more adherent to the basic SPH formulation and well suited for gases.

The second formulation is better suited for liquids and solids and its expression may be obtained by the continuity equation of the Navier-Stokes set, applying the SPH approximation to both sides of the equation and taking advantage of the following identity:

$$\rho\frac{\partial v^\beta}{\partial x^\beta} = \frac{\partial \rho v^\beta}{\partial x^\beta} - v^\beta\frac{\partial \rho^\beta}{\partial x^\beta}$$

The density expression then reads:

$$\frac{D\rho_i}{Dt} = \sum_j m_j v_{ij}^\beta \frac{\partial W_{ij}}{\partial x_i^\beta}$$

The density change rate around the particle appears to be related to the relative velocity of the particles in the support domain $v_{ij}^\beta = v_i^\beta - v_j^\beta$.

A third expression is available, it is called *normalized summation density* and is a modified version of the summation density expression:

$$\rho_i = \frac{\sum_j m_j W_{ij}}{\sum_j \frac{m_j}{\rho_j} W_{ij}}$$

this expression improves the accuracy of the *summation density* near the free boundaries or is material discontinuities are present and for general fluid flow problems, but it is not accurate with shock waves.

Momentum equation

Two different expressions of the momentum equation may be derived depending on the form used to approximate the gradient operator. The two expressions are apparently equivalent and are both symmetrised to reduce the particle inconsistency errors.

$$\frac{Dv_i^\alpha}{Dt} = \sum_j m_j \left(\frac{\sigma_i^{\alpha\beta} + \sigma_j^{\alpha\beta}}{\rho_i \rho_j} + \Pi_{ij} \right) \frac{\partial W_{ij}}{\partial x_i^\beta} + g_i^\alpha$$

$$\frac{Dv_i^\alpha}{Dt} = \sum_j m_j \left(\frac{\sigma_i^{\alpha\beta}}{\rho_i^2} + \frac{\sigma_j^{\alpha\beta}}{\rho_j^2} + \Pi_{ij} \right) \frac{\partial W_{ij}}{\partial x_i^\beta} + g_i^\alpha$$

The term Π_{ij} is a corrective term for the numerics called artificial viscosity and will be discussed in the following; $\mathbf{g}_i$ is an external acceleration vector for the particle .

The viscous tangential stresses and the isotropic pressure can be separated and the following two expressions of the momentum equation can be obtained

$$\frac{Dv_i^\alpha}{Dt} = \sum_j m_j \left(\frac{p_i + p_j}{\rho_i \rho_j} + \frac{\mu \varepsilon_i^{\alpha\beta} + \mu \varepsilon_j^{\alpha\beta}}{\rho_i \rho_j} + \Pi_{ij} \right) \frac{\partial W_{ij}}{\partial x_i^\beta} + g_i^\alpha$$

$$\frac{Dv_i^\alpha}{Dt} = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \frac{\mu \varepsilon_i^{\alpha\beta}}{\rho_i^2} + \frac{\mu \varepsilon_j^{\alpha\beta}}{\rho_j^2} + \Pi_{ij} \right) \frac{\partial W_{ij}}{\partial x_i^\beta} + g_i^\alpha$$

Energy equation

Here again, depending on the chosen approximation of the gradient two expressions of the energy conservation equation can be given:

$$\frac{De_i}{Dt} = \frac{1}{2} \sum_j m_j \left(\frac{p_i + p_j}{\rho_i \rho_j} + \Pi_{ij} \right) v_{ij}^\beta \frac{\partial W_{ij}}{\partial x_i^\beta} + \frac{\mu_i}{2\rho_i} \varepsilon_i^{\alpha\beta} \varepsilon_i^{\alpha\beta} + H_i + \frac{Q_i}{\rho_i}$$

$$\frac{De_i}{Dt} = \frac{1}{2} \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) v_{ij}^\beta \frac{\partial W_{ij}}{\partial x_i^\beta} + \frac{\mu_i}{2\rho_i} \varepsilon_i^{\alpha\beta} \varepsilon_i^{\alpha\beta} + H_i + \frac{Q_i}{\rho_i}$$

the artificial viscosity term Π_{ij} appears here as well, and the term H_i is related to the artificial heat that will be discussed briefly in the following. The external power for the particle is due the beam power deposition as calculated by FLUKA. The FLUKA result is energy per unit volume and per particle; it is converted to a power per unit volume on the particle position Q_i by applying the time deposition law.

Equation Of State

The NS set of equations have to be closed by an additional equation related to the material, the Equation Of State (EOS), that links the stress tensor, to the deformation tensor and to the internal energy. For a liquid the pressure is calculated as a function of the density and the specific energy.

$$p = f(\rho, e)$$

The equation of state is given in an analytical form, necessary for the speed of the computation, and is obtained by fitting the results of experimental data.

These experiments are difficult; however a great number of analytical model and corresponding parameters are available for the most common materials, even if these data is not easily accessible in the open literature nowadays.

Several analytical models have been proposed, giving good fitting for different areas of the phase diagram and different load conditions. The following models are implemented:

- Ideal Gas law
- Mie-Gruneisen Polynomial EOS
- Mie-Gruneisen Hugoniot Shock EOS
- Puff EOS

The first is the simple ideal law for gases, the others are based on the Mie-Gruneisen form of the equation of state, that can be written as:

$$p(\rho, e) = p_r(\rho) + \rho \Gamma(\rho)[e - e_r(\rho)]$$

The pressure is expressed as the sum of two separate functions of density and energy. The functions p_r and e_r are calculated on some reference curve, like and isobar, the standard adiabat or the shock Hugoniot curve

Γ is the Gruneisen Gamma, defined as:

$$\Gamma = \frac{1}{\rho}\left(\frac{\partial p}{\partial e}\right)_\rho$$

Ideal Gas law

For an ideal gas the pressure is defined as:

$$p = \frac{c_p - c_v}{c_v}\rho e$$

Where c_p and c_v are the specific heat at constant pressure and volume respectively.

Polynomial EOS

The parameter μ is defined as follows, where ρ_o is the initial density

$$\mu = \frac{\rho}{\rho_0} - 1$$

The compression ($\mu > o$) and tension ($\mu < o$) behaviour are modelled separately

$$p = \begin{cases} A_1\mu + A_2\mu^2 + A_3\mu^3 + (B_0 + B_1\mu)\rho_0 e, & \mu \geq 0 \\ T_1\mu + T_2\mu^2 + B_0\rho_0 e, & \mu < 0 \end{cases}$$

Shock EOS

The Mie-Gruneisen model is based on the Hugoniot Shock equation, pressure and energy on the Hugoniot curve are defined as follows:

$$p_H = \frac{\rho_0 c_0^2 \mu(1 + \mu)}{[1 - (s - 1)\mu]^2}$$

$$e_H = \frac{1}{2}\frac{p_H}{\rho_0}\left(\frac{\mu}{1 + \mu}\right)$$

These expression are based on the following relation between particle velocity u_p, the shock velocity U_s and the sound velocity c_o, through the fitting constant s.

$$U_s = c_o + s u_p$$

The parameter μ is defined as follows, where ρ_o is the initial density

$$\mu = \frac{\rho}{\rho_0} - 1$$

And the pressure function is defined:

$$p = p_H + \Gamma \rho (e - e_H)$$

With $\Gamma\rho = \Gamma_o \rho_o$ and Γ_o being another parameter.

PUFF EOS

This equation of state is more complicated and covers the behaviour of the material from cold shocked regions to highly expanded hot regions, the material sublimation is considered. The phase diagram is divided in three regions that are separately modelled.

The parameter μ and η are defined:

$$\eta = \frac{\rho}{\rho_0}$$

$$\mu = \frac{\rho}{\rho_0} - 1$$

With respect to the previous models, much more material parameters are required by this equation of state

$$p = \begin{cases} (A_1\mu + A_2\mu^2 + A_3\mu^3)\left(1 - \Gamma\frac{\mu}{2}\right) + \Gamma\rho e, & \mu \geq 0 \\ (T_1\mu + T_2\mu^2)\left(1 - \Gamma\frac{\mu}{2}\right) + \Gamma\rho e, & \mu < 0, \quad e < e_s \\ \rho\left[H + (\Gamma_0 - H)\eta^{\frac{1}{2}}\right]\left[e - e_s\left(1 - e^{\frac{A_1(\eta-1)}{\rho_0\Gamma_0 e_s\eta^2}}\right)\right], & \mu < 0, \quad e \geq e_s \end{cases}$$

Numerical aspects

Artificial viscosity

The calculation of fluid flows that involve the formation and propagation of shock waves require an additional term to be added to the momentum equation to regularize and dissipate the non-physical post shock oscillations, and to avoid particle interpenetration at high Mach number flows.

Several expression have been proposed for the artificial viscosity Π_{ij}. Monoghan proposed an expression that gives good results for flow near the shock:

$$\Pi_{ij} = \begin{cases} \dfrac{\alpha \bar{c}_{ij}\mu_{ij} + \beta\mu_{ij}^2}{\bar{\rho}_{ij}}, & (v_i - v_j)\cdot(x_i - x_j) < 0 \\ 0, & (v_i - v_j)\cdot(x_i - x_j) \geq 0 \end{cases}$$

the artificial viscosity is active only when the particles are approaching each other

$$\mu_{ij} = \frac{(v_i - v_j)\cdot(x_i - x_j)}{\bar{h}_{ij}\left(\dfrac{|x_i - x_j|^2}{\bar{h}_{ij}} + \eta^2\right)}$$

here $\bar{c}_{ij}$, $\bar{\rho}_{ij}$ and $\bar{h}_{ij}$ are the values of the speed of sound of the density and of the smoothing length

averaged between the two interacting particles, i.e. $\bar{c}_{ij} = (c_i + c_j)/2$. α and β are dimensionless constants and the parameter η is necessary to avoid numerical singularities, their values are usually chosen as: $\alpha=1$, $\beta=2$ and $\eta^2=0.01$.

Artificial heat

An excessive heating may be introduced in the SPH simulation when the artificial viscosity term is included in the formulation, these errors are commonly referred to as wall heating and may appear for instance when the flow impacts a rigid wall. An artificial heat conduction term may be added to the energy equation in order to reduce these numerical errors.

The artificial heat conduction term is expressed as:

$$H_{ij} = \frac{2\gamma_{ij}(e_i - e_j)}{\rho_{ij}\bar{h}_{ij}^2\left(\dfrac{|x_i - x_j|^2}{\bar{h}_{ij}^2} + \eta^2\right)}$$

where $\gamma_{ij} = (\gamma_i + \gamma_j)/2$ and

$$\gamma_i = g_1 h_i c_i + g_2 h_i^2(|\nabla \cdot v_i| - \nabla \cdot v_i)$$

$g_1=g_2=0.5$ are two constants. This term is again active only for particles approaching each other and is zero otherwise.

The artificial heating is also useful in smoothing out the errors in the velocity field that are not corrected by the artificial viscosity and hence may improve the shock profile.

Viscosity

The SPH method was originally developed to solve the Euler problem, hence simulating inviscid flows.

The tangential stress in the momentum and energy equations is calculated from the deviatoric part of the deformation tensor ε. In the SPH formulation the following expression may be adopted.

$$\varepsilon_i^{\alpha\beta} = \sum_j \frac{m_j}{\rho_j}v_{ji}^{\beta}\frac{\partial W_{ij}}{\partial x_i^{\alpha}} + \sum_j \frac{m_j}{\rho_j}v_{ji}^{\beta}\frac{\partial W_{ij}}{\partial x_i^{\beta}} - \delta^{\alpha\beta}\left(\frac{2}{3}\sum_j \frac{m_j}{\rho_j}v_{ji}\cdot\nabla_i W_{ij}\right)$$

Implementation

Time integration

Any of the standard time advancing schemes adopted for explicit dynamics numerical methods may be adopted in SPH. The most common choices are the Leap-Frog and Runge-Kutta methods, both accurate to the second order. The Leap-Frog is the scheme adopted in the code, it requires less memory than the Runge-Kutta method and only one force term evaluation per time step. The Runge-Kutta has some advantages for adaptive time step schemes.

Considering the NS equation we define:

$$\dot{\rho}_i = D_i$$
$$\dot{v}_i = F_i$$
$$\dot{e}_i = E_i$$

According to the time evolution scheme we first calculate:

$$\rho_i^{n+\frac{1}{2}} = \rho_i^n + \frac{\Delta t}{2} D_i^{n-\frac{1}{2}}$$
$$v_i^{n+\frac{1}{2}} = v_i^n + \frac{\Delta t}{2} F_i^{n-\frac{1}{2}}$$
$$e_i^{n+\frac{1}{2}} = e_i^n + \frac{\Delta t}{2} E_i^{n-\frac{1}{2}}$$

the pressure is calculated from the equation of state $p^{n+1/2} = f(\rho^{n+1/2}, u^{n+1/2})$ and the values of $D_i^{n+1/2}$, $F_i^{n+1/2}$ and $E_i^{n+1/2}$ are calculated on the basis of the SPH approximation of the continuity, momentum and energy equations. Finally the new value of the variables is calculated:

$$\rho_i^{n+1} = \rho_i^n + \Delta t\, D_i^{n+\frac{1}{2}}$$
$$v_i^{n+1} = v_i^n + \Delta t\, F_i^{n+\frac{1}{2}}$$
$$e_i^{n+1} = e_i^n + \Delta t\, E_i^{n+\frac{1}{2}}$$

and the particle position is updated as follows.

$$x_i^{n+1} = x_i^{n+1} + \Delta t\, v_i^{n+1}$$

Any explicit time advancing scheme is subject to the Courant-Friedrichs-Levy condition for stability, that simply imposes that the maximum speed of numerical propagation cannot exceed the maximum speed of physical propagation. It requires the time step to be proportional to the particle resolution and hence to the smoothing length.

$$\Delta t < \Delta t_{CFL} \min_i \left[\frac{h_i}{c_i} \right]$$

Particle interaction

As a difference from grid methods, where the position of the nodes and cells is known in advance, the "connections" and possible interactions between particles in SPH can vary with time, and theoretically the neighbours of each particle are continuously changing.

The support of the kernel function is compact and therefore only a limited number of particles may have an influence on or be influenced by a given particle, these particles are denoted as Nearest Neighbouring Particles (NNP) and the algorithm that searches for these and establishes the possible connections is denoted as Nearest Neighbouring Particle Searching (NNPS).

Two algorithms are implemented the simple all-pair search and a linked-list algorithm.

All-pair search

This is the simplest possible algorithm. It consists in a double cycle on all the particles. All the possible particle pairs are examined, if the distance between the particles is lower than the kernel's support size the possible interacting pair is considered. The number of operation is proportional to square of the particle number.

This method is perfectly safe and accurate but can be really slow in its execution, and is practically applicable only for one dimensional problems.

Linked-list algorithm

In this method a cartesian grid is superposed to the computational domain, and the cells of the auxiliary grid are used to store the information on the particle position.

A cycle on all the particles is run to find the corresponding cell, at the same time the particles assigned to each cell are stored in a dedicated data structure.

The search for possible interactions is limited to the particles belonging to the same cell of the particle of interest or to the cells that are adjacent to it.
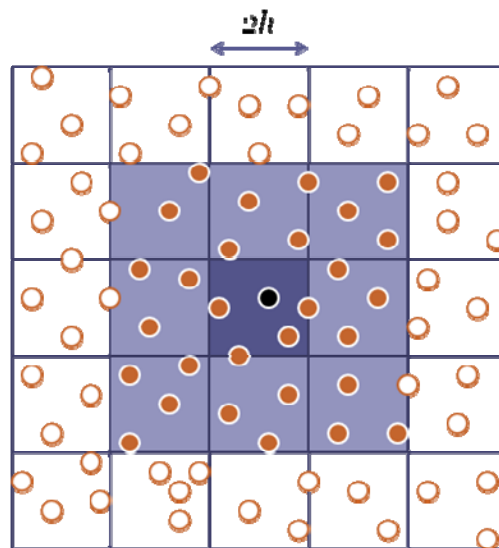
Figure 2: linked list particle searching algorithm, only the particles in the neighboring cells are possible candidates for the interaction

The algorithm is more demanding for the memory occupancy due to the auxiliary grid and related data structure, but its complexity is reduced, the number of operation is in fact directly proportional to the number of particles.

Boundary treatment

Imposing free slip boundary conditions is certainly not straightforward for this method.

A standard approach to enforce the no-penetration of a solid boundary consist in using repulsive particles, placed on the surface of the boundary and acting on the fluid particle with a force inversely proportional to their distance from the boundary. This method is computationally efficient, but requires the manual setting of parameters that are dependent on the particle spacing and properties; moreover it may cause spurious pressure waves when rigid materials are used.

The use of "ghost particles" is preferred, it is less efficient but is far more accurate and do not suffer from the limitations of the other approach.

When a particle approaches a rigid boundary a "ghost particle" is generated automatically on the opposite side, having the same properties of density and pressure but a "mirrored velocity", meaning that it has the same tangential velocity of the fluid particle but opposite normal velocity component.
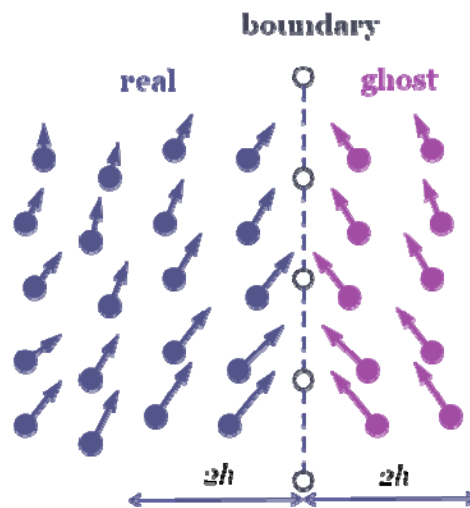
Figure 3: fee slip boundary conditions, the ghost particles are created reflecting the real particles close to the boundary

The ghost particles are included among the possible particle interaction of the fluid particles, and this is sufficient to impose free-slip boundary conditions at the rigid boundary. The "ghost particle" position is calculated at each time step mirroring the fluid particles. Only the particles that are close to the boundary are reflected, at a distance proportional to the kernel size.

Particle regularization

The SPH method may "suffer" numerically when the material considered becomes stiffer. A high value of the speed of sound, means in fact that a minimal variation in the density may cause high pressure gradients, that propagate in the computational domain.

The initial density must be as uniform as possible to avoid these effects, and this implies that the particle distribution shall be perfectly regular.

It is not easy anyway to guarantee this near the boundaries of the domain, either free, or free-slip.

Near the free boundaries there is the "particle deficiency" problem: the support of a particle close to such a border may be partly outside of the computational domain, hence there are not enough particles to have the nominal density value, when the summation density approach is used. This can be avoided is the "continuity equation" is adopted for the density calculation.

The problem still holds close to rigid free-slip boundaries, since the distance of the particles from the border is not constant and the presence of the ghost particles do not solve the problem completely.

A density regularization is therefore performed before the real calculation. It consists in an optimization of the mass of the particles, that is changed in order to have density as uniform as possible.
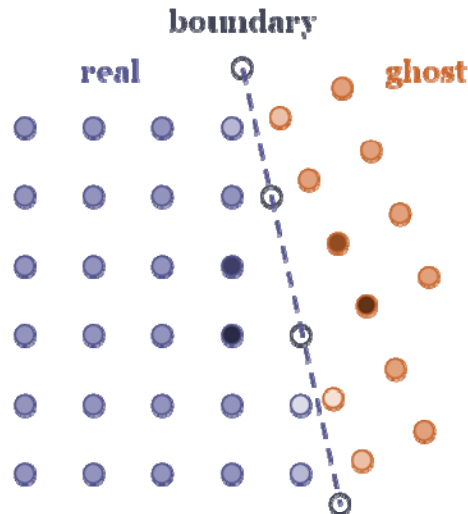
Figure 4: particle regularization: the mass of the particles close to a boundary is increased if the particle is far from the boundary and decreased if it is close to the boundary; the initial density is thus kept uniform

The particle mass is updated based on the local value of the density, either increased if the density is lower than the nominal value of decreased is the local density is higher. This calculation is iterated several times until the density is not changing anymore and is limited to the particles close to the rigid boundaries, and not to those on the free surfaces.

FLUKA interface

The energy deposition due to the particle beam interaction with the material is calculated by FLUKA, with a Montecarlo algorithm.

The grids adopted by FLUKA are structured and regular, either Cartesian or cylindrical, and the output is given as a matrix. The FLUKA binning file is directly read by the code. The energy deposition, multiplied by the appropriate scaling factors, is converted in a power deposition considering the number of particles involved and the time deposition law.

The position of each particle is calculated with respect to the FLUKA grid, and the value of the specific power for the particle is calculated interpolating the values of the matrix. This additional particle energy is included in the balance of the energy equation and causes the expansion of the fluid as imposed by the equation of state.
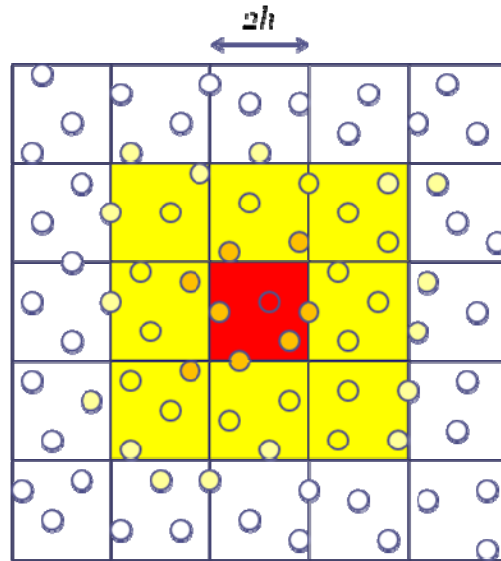
Figure 5: FLUKA interface: the power deposited on each particle is calulatied interpolating the results on the FLUKA results matrix

Simulation procedure

The SPH code presented is suitable for the transient simulation of fluid-dynamics under the effect of particle beam interaction and related power deposition.

Computer Aided Engineering methodologies share some procedural steps, that are common to the method and code proposed as well. These steps may be summarized as follows:

- Geometry definition
- Simulation setup
- Simulation run
- Results review

Each step is briefly discussed hereafter. The solution proposed allows to complete the simulation procedure with tools that have been implemented or that are already available at CERN or adopt the Free Software or Open Source model.

Geometry definition

This is probably the most time consuming part for the user. It is difficult to find a good CAD or modelling software that may be really easy and efficient to use, even if there are several solutions available, either commercial or free.

The geometry normally has to be defined specifically for a simulation, the models used for drawing, are not normally the same used for calculus. This is due to the fact the details and the information required in drawing and in simulation are different. The meshing procedure required by the traditional FEM or CFD simulation demand for a specific modelling and meshing tool.

Fortunately, for the SPH methodology proposed, being a mesh-less method, only the geometry has to be input, and any possible tool that defines volumes and surface may in principle be used, such as CAD tools, or FEM and CFD modellers or also specific geometry definition languages, as the one used for FLUKA.

Up to now a FEM interface has been developed, that reads a finite element model, meshed with tetrahedrons for the volume and with triangles for the surfaces, for instance using the ANSYS workbench package.
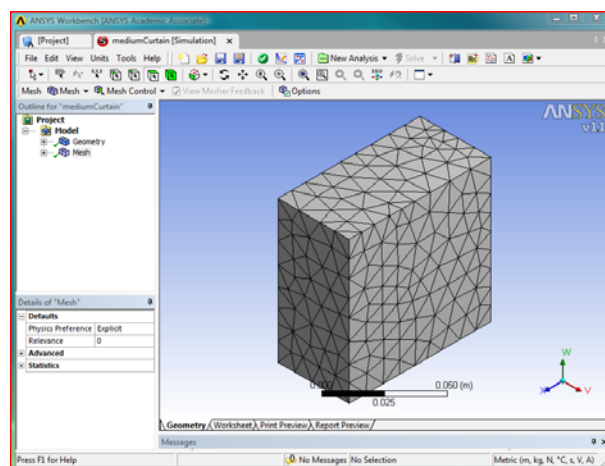
Figure 6: model meshing with tetrahedron in ANSYS Workbench

The grid is used only for geometry definition, its accuracy is required only to exactly define the borders of the model. The grid has to be exported, in a standard format, namely the Nastran bulk data text file.
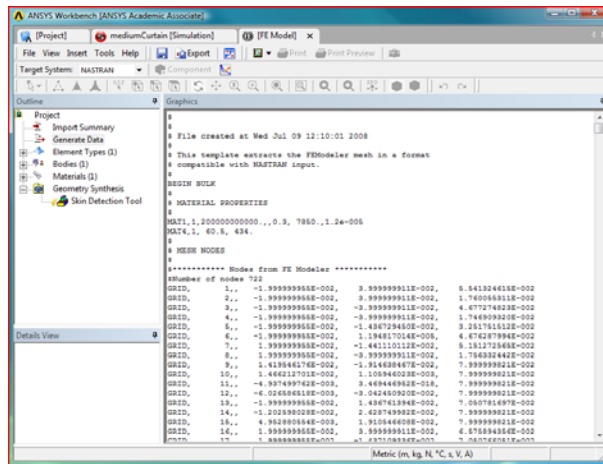

Figure 7: FEM model export in NASTRAN format

This file defines the geometry and is used in the setup of the simulation to define the position of the particles. The methodology is mesh-less and the particles are "sprayed" uniformly in the volume and on the surfaces, the original model has little influence on the particle position.

Simulation setup

The parameters adopted in the simulation, like the time step and the number of steps for instance have to be defined, together with the initial particle position and properties. The simulation code has to be provided with some file detailing all this data, and it may be really annoying for the user to manually prepare this input, that's why a graphical interface has been developed to ease this otherwise tedious part of the procedure.

The interface is a Python script that uses the common and free wxPython and NumPy libraries, it can be very easily extended and modified. All the simulation data is stored in a directory, separate directories are used for different analyses, and the file names are always the same. The interface is composed by a menu system, and a notebook like panel, with several pages covering the different data to be input.

The menus
File
The file menu (as usual) allows starting a new input from scratch, and to read and save the data in a directory.

Particles
This menus allows to define some properties of the particle such as the material assigned to them, their mass and smoothing length or the initial conditions of pressure and velocity.

The panels
Options
The options panel allow to set the fundamental parameters for the simulation and for the behaviour of the simulation code.

The user can choose among the different kernel functions implemented, if and which algorithm to use for a variable smoothing length, the algorithm for Nearest Neighbouring Particle Searching, which form of the momentum and energy equation and which approach for the density calculation.

Moreover here the user can decide if using an XPSH approach and consider an averaged velocity for the particles, can decide if free-slip boundary conditions have to be included in the model, if the fluid has significant viscosity. The user can also activate the use of artificial viscosity and heat to reduce the numerical errors.

Finally the user can tell the simulation code if an external heating (due to particle deposition) and if gravity have to be included in the simulation.
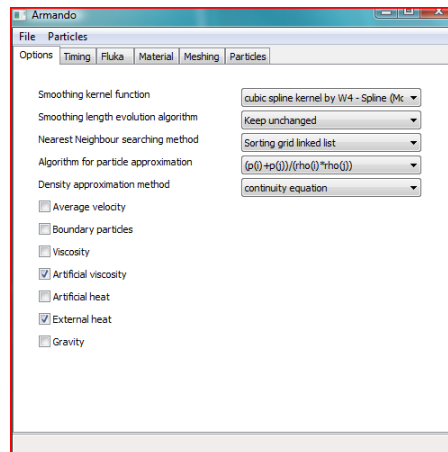
Figure 8: the options panel of the graphical interface

Timing

In this panel the time step value and the number of time steps is set. The user is responsible of the choice of an appropriate time step, keeping in mind the limitations imposed by the CFL condition. An automatic tool to warn the user for excessive short or long time step is foreseen and can be easily implemented.

The final time of the simulation is given. The user has also to input the frequency, or the number of steps at which the particle connections are refreshed and calculated again with the NNPS algorithm, and the frequency at which the results of particle position velocities and properties are saved.
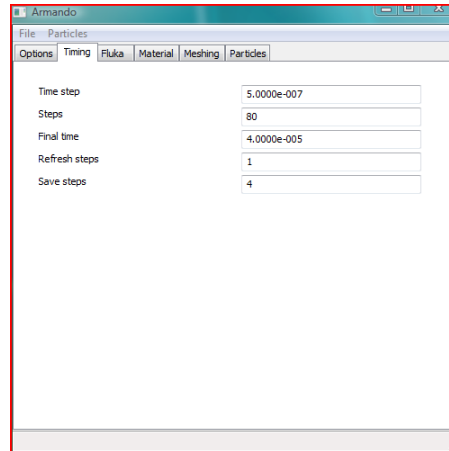
Figure 9: the timing panel of the graphical interface

Fluka

This panel allows to set the parameters to read and use the FLUKA binning file, for the energy deposition input. Two conversion factors can be set due to the units used in FLUKA, namely cm for length and GeV for energies. The energy deposition per one particle is given, therefore the number of particles and the duration of the energy deposition may be input here.

More complicate energy deposition laws (for instance when several bunches are deposited) may be easily defined modifying the script of the interface, but also running the simulation in subsequent steps and turning on and off the external heating switch.

Moreover a translation of the FLUKA grid with respect to the reference system of the model can be defined.
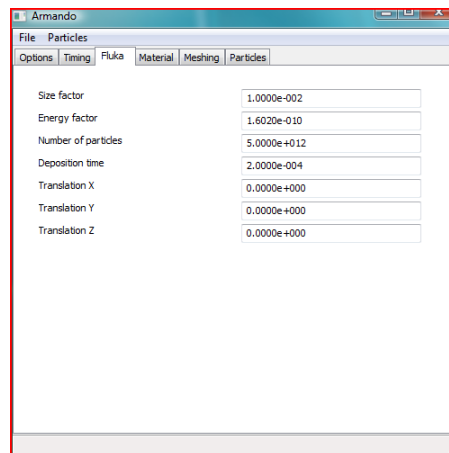
Figure 10: FLUKA energy deposition panel of the graphical interface

Material

This panel is used to set the material properties. In the present implementation only one material may be present in the model, the material number defines the type of equation of state to use: the first ten for ideal gas, then ten for polynomial EOS, then shock EOS and PUFF EOS. The fact that ten materials for each kind may be defined is in view of future developments of the code.

Depending on the material number and on the corresponding material model the material parameters have to be input in the corresponding text fields.
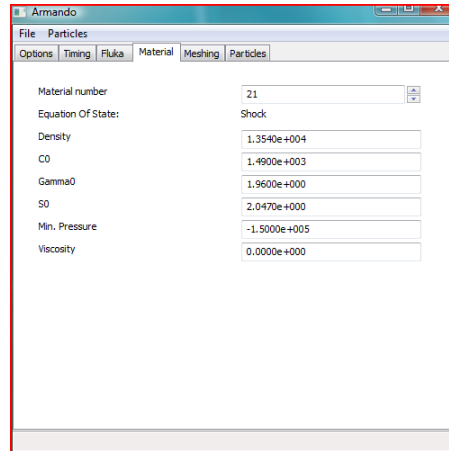
Figure 11: material definition panel of the graphical interface

Meshing

This panel is used to define the number and position of the SPH particles. It is based on the FEM geometry definition as discussed. The standard procedure consists in reading a finite element model file, in the Nastran bulk data text format, define the smoothing length and the number of particles per smoothing length (usually 2) and then simply press the mesh button.

The algorithm calculates the geometrical limits of the model, and consider the possible particles to be positioned in a regular Cartesian grid. Only the particles that are inside the model are effectively included in the model. These have the material number defined in the previous model, with its given density. The mass is calculated to be consistent with the density.
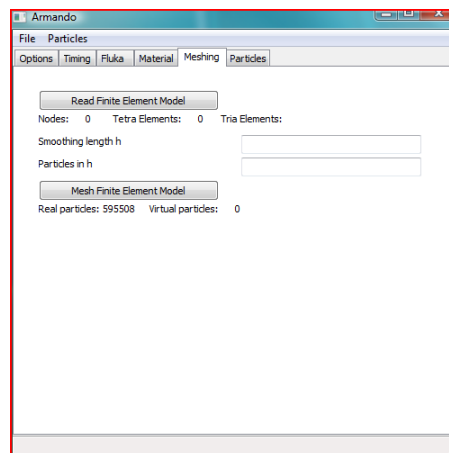
Figure 12: particle meshing panel of the graphical interface

Particles

Finally in this panel the particle position may be reviewed, for the moment only as a list of coordinates. This panel serves to verify the correctness of the previous input and of the meshing procedure, more sophisticated and efficient graphical output is foreseen as a possible future development of the graphical interface.

Figure 13: particle position panel of the graphical interface

Simulation run

The graphical interface prepares all the data for the simulation and saves the files in a directory chosen by the user. Seven files are saved:

- **f_xv.dat**   position and velocity of the particles
- **f_state.dat** properties of the particles: mass, density, pressure and specific energy
- **f_other.dat** other particle data: material number and smoothing length
- **v_xv.dat**   position and normal vector of the virtual particles
- **v_state.dat** properties of the virtual particles
- **v_other.dat** other virtual particles properties
- **input.sph**   file containing the simulation parameters

The virtual particles are positioned on the free-slip boundary surfaces if present, and have a vector associated defining the local normal to the surface. These particles are only used as a geometrical tool to define the boundaries.

The file **input.sph** is a text file defining all the simulation parameters an example is given in the figure.
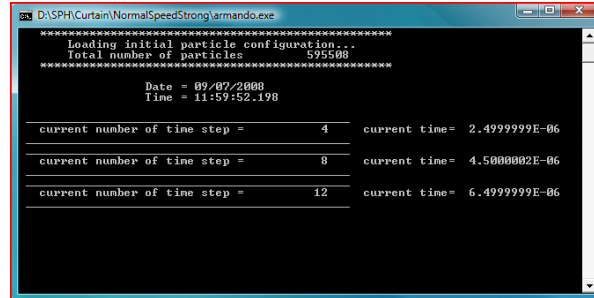

Figure 14: Example of the input.sph file

Two other files have to be manually copied in the simulation directory: the code executable and the FLUKA binning file.

The code reads the FLUKA file in standard binary form, the name of the file is fixed and is **fluka_binning**.

The simulation can be run either through the command prompt window or simply double clicking on the executable.

Figure 15: a simulation running

Results review

The snapshots of the particles are saved with a frequency that is determined by the user with the graphical interface.

The file format chosen is denominated EnSight, it is a text file format composed by several files, a master file addressing the others and defining the quantities saved and the time at which these are saved, and a set of files for the results, one for each quantity and for each saving time, listing the scalar and vector results for each particle.

The EnSight results file may be visualized by the commercial visualization software EnSight of course, but also several other commercial and also free visualizers are capable of reading and represent this data format.

It is not easy to represent efficiently meshless results, it is a rather new technique and all the visualization software is designed for grid based methods. There is an extremely interesting project of the Swiss National Supercomputing Centre (CSCS) that have developed Paraview-meshless, an extension of the visualization platform Paraview dedicated to meshless software results.

ParaView is an open-source, multi-platform application designed to visualize data sets of size varying from small to very large. ParaView runs on distributed and shared memory parallel as well as single processor systems, it is open-source and multi-platform.
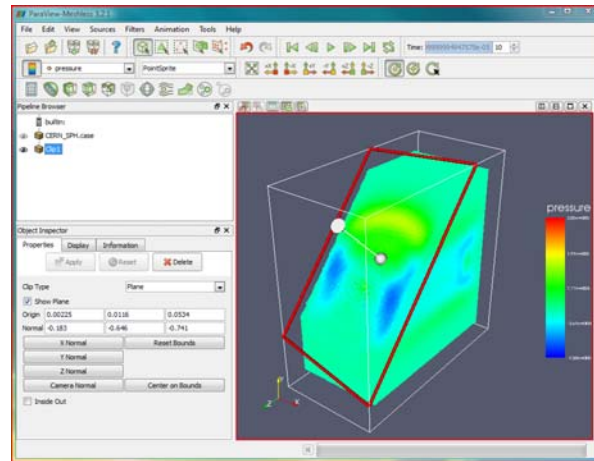
Figure 16: results review with Paraview-meshless

Code description

The simulation code is developed taking as a starting point the SPH code of the book "Smoothed Particle Hydrodynamics: a meshfree particle method" by G.R. Liu and M.B. Liu. It is written in Fortran 90, to take advantage of the dynamic memory allocation.

The following table show the list of the principal variables used, and a list of the subroutines with a brief description. The diagrams show graphically the structure of the code and how subroutines are related.

Table 1: main variables description

Variable name	Description
General variables	
np	Number of real fluid particles
nv	Number of virtual particles used to define the boundary surfaces
ng	Number og ghost particles used to impose the free-slip boundary conditions
x(:)	Particle position vector
v(:)	Particle velocity vector
mat(:)	Particle material number
h(:)	Smoothing length vector
mass(:)	Particle mass vector
rho(:)	Particle density vector
p(:)	Particle pressure vector
u(:)	Particle internal energy vector
c(:)	Particle sound speed
Particle interaction	
niac	Number of interaction pairs
pair_i(:)	Index of the first particle of the interaction pair
pair_j(:)	Index of the second particle of the interaction pair
w	Smoothing kernel value for a given interaction pair
dwdx	First derivative of the kernel function for an interaction pair
FLUKA interface	
ofluka	Origin of the binning
dfluka	Size of the binning cell
nfluka	Number of cells for each index
tfluka	Identifier for the fluka grid type

Variable name	Description
datafluka(:)	Data of the binning

Table 2: subroutine description

Function name	Description
art_heat	Subroutine to calculate the artificial heat correction (Noh 1978)
art_visc	Subroutine to calculate the artificial viscosity (Monaghan 1992)
av_vel	Subroutine to calculate the average velocity (XSPH) to correct velocity for preventing penetration (Monaghan, 1992)
default	This subroutine is used to define the default values of the common options, like the kernel function, the density approximation etc.
sum_density	Subroutine to calculate the density with SPH summation algorithm
nor_density	Subroutine to calculate the density with SPH summation algorithm with density normalization to avoid the boundary deficiency problem (Randles and Libersky 1996)
con_density	Subroutine to calculate the density variation based on the continuity equation
direct_find	Subroutine to calculate the interaction pairs between the particles the pairs are determined by directly comparing the particle distance with the corresponding smoothing length
ensightout_results	Subroutine for saving particle information on position, velocity, density, pressure and enery to external disk file in the EnSight format
ensightout_case	Subroutine for saving the master file of the results in the EnSight format to external disk file
eos_gas	Subroutine for calculating pressure with the Ideal Gas EOS
eos_poly	Subroutine for calculating pressure with the Mie-Gruneisen Polynomial EOS
eos_shock	Subroutine for calculating pressure with the Mie-Gruneisen Shock EOS
eos_puff	Subroutine for calculating pressure with the Mie-Gruneisen PUFF EOS
read_binning_bin	Subroutine to read and store the FLUKA results from a binning binary file
read_binning_txt	Subroutine to read and store the FLUKA results from a binning text file
fluka_heat	Subroutine to calculate the power deposition on the particles from the FLUKA results
ghost_particles	Subroutine to calculate the number of ghost particles and to define an array that linking real and ghost particles
h_upgrade	Subroutine to evolve smoothing length
read_input_file	Subroutine to read the input file, and set the options for the simulation
read_initial_conditions	Subroutine for loading initial status of the real particles
read_virtual_particles	Subroutine for loading the data realtive to the boundary particles

Function name	Description
int_force	Subroutine to calculate the internal forces on the right hand side of the Navier-Stokes equations, i.e. the pressure gradient and the gradient of the viscous stress tensor, used by the time integration. Moreover the entropy production due to viscous dissipation, tds/dt, and the change of internal energy per mass, du/dt, are calculated.
kernel	Subroutine to calculate the smoothing kernel wij and its derivatives dwdxij
link_list	Subroutine to calculate the interaction pairs between the particles the pairs are determined by by using a sorting grid linked list
output	Subroutine for saving particle information to external disk file
single_step	Subroutine to determine the right hand side of a differential equation in a single step for performing time integration
smooth_particles	Subroutine to optimize the mass of the particles close to the boundary to have a uniform density in the computational domain
time_integration	Subroutine for leapfrog time integration. The particle position, velocity and energy is updated on the basis of the right hand side of the momentum and energy equation as calculated by the single_step subroutine
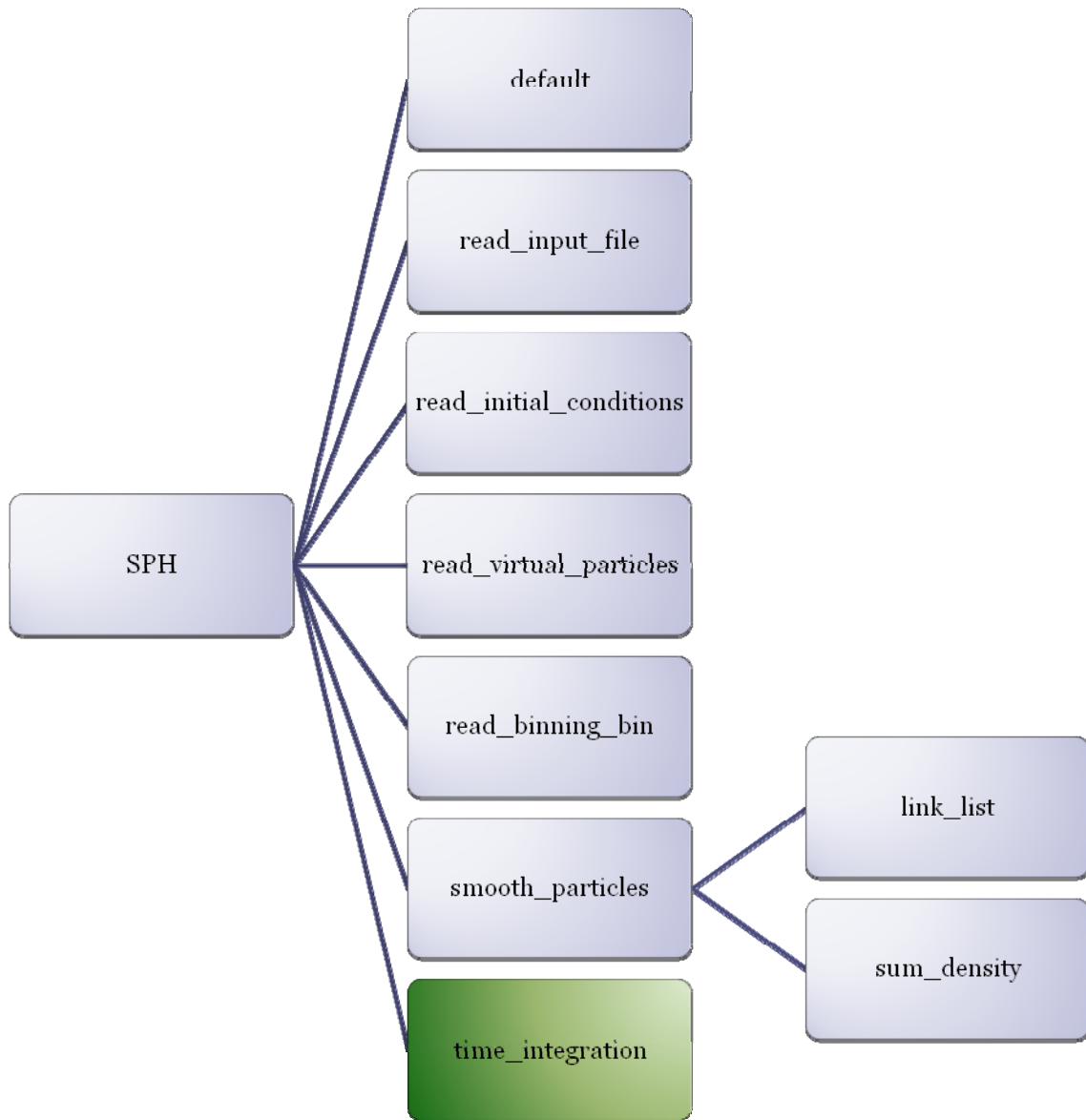
Figure 17 program structure: main program
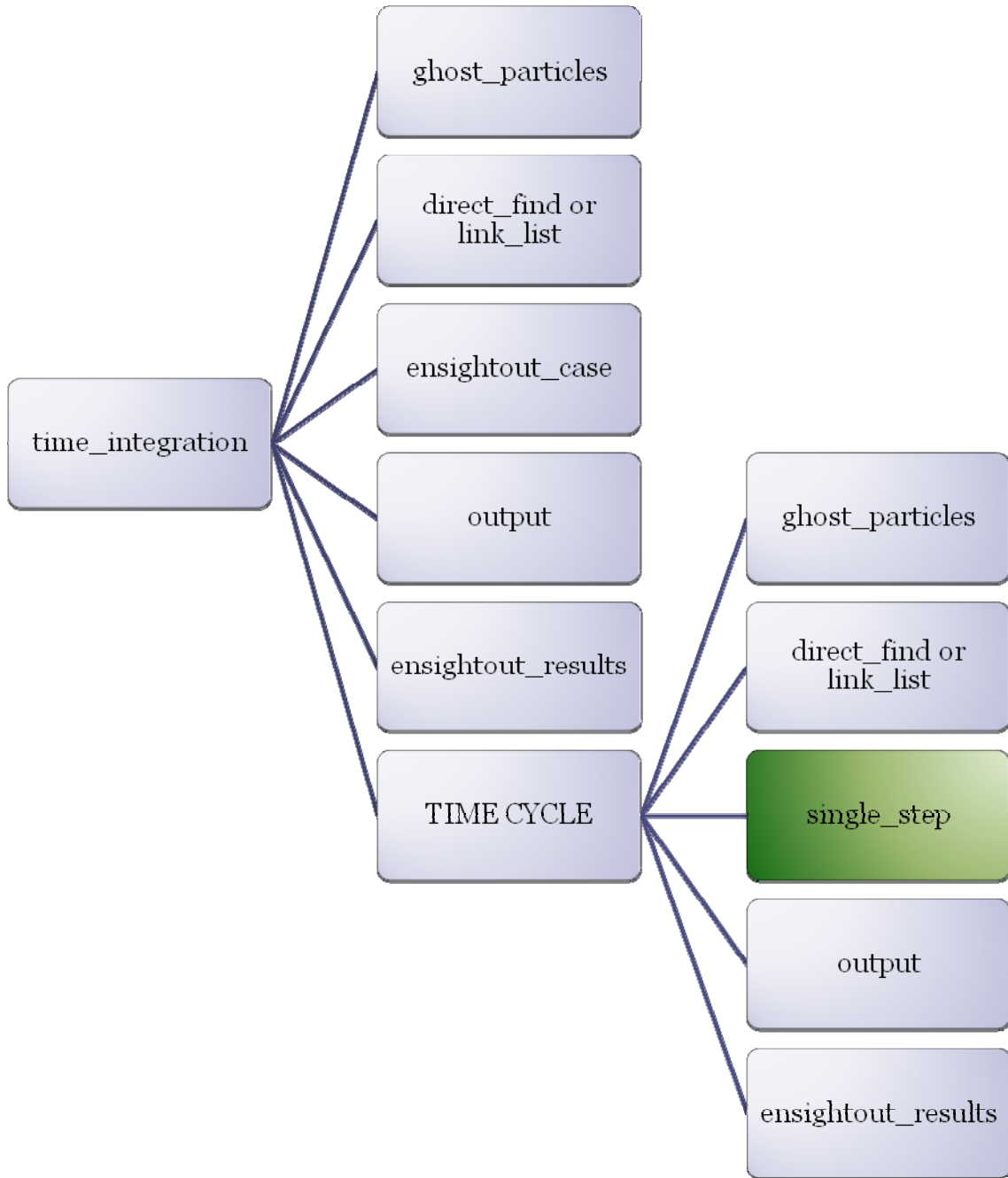
Figure 18 program structure: time integration procedure
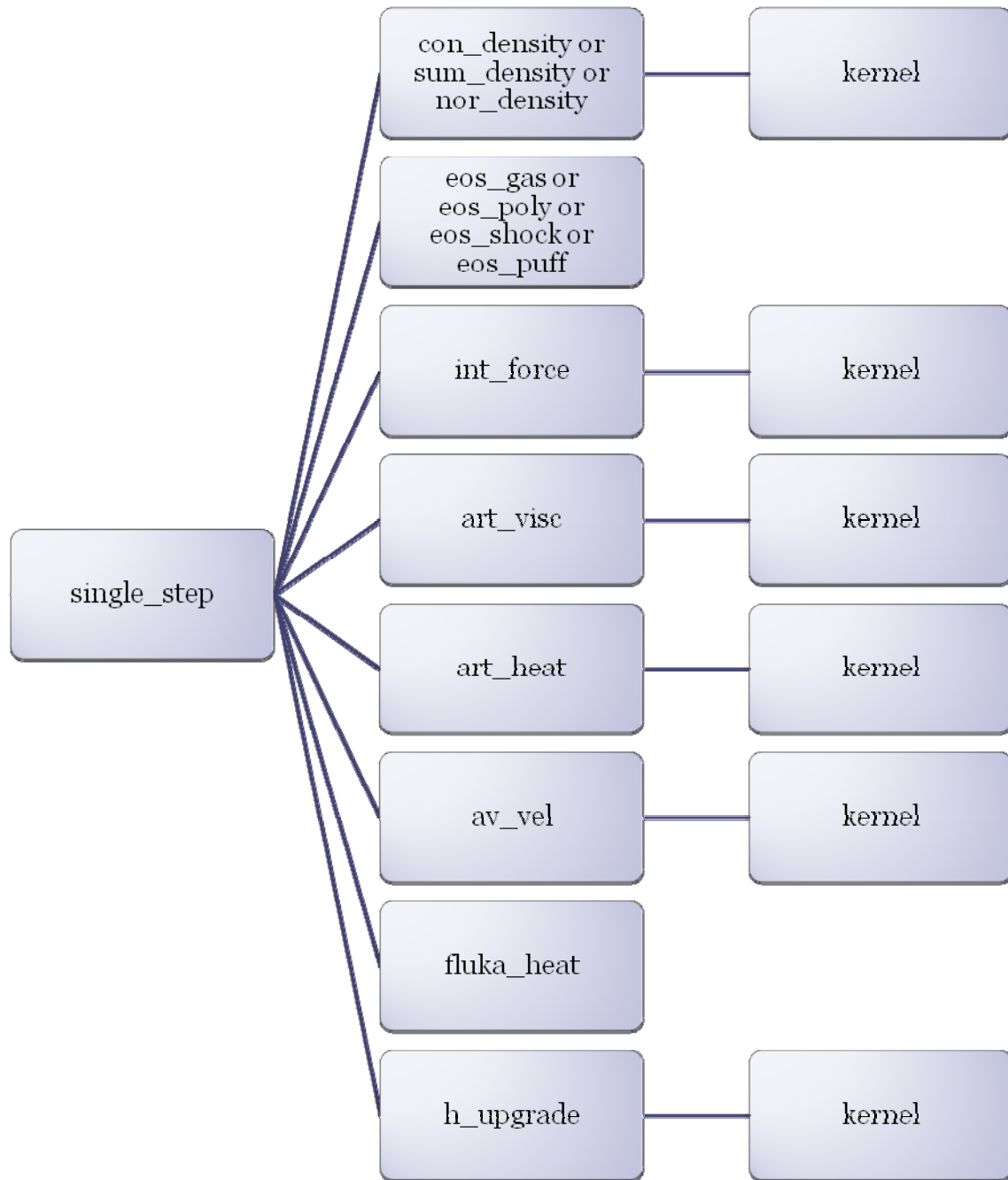
Figure 19 program structure: single_step procedure, the core of the program

Examples

Elastic wave

The first is a 2d example of pressure wave propagation. It is based on a preliminary design of a collimator in which a graphite block is shrink fitted in a rigid pipe. A particle beam is deposited in the block, heating it and giving rise to a pressure wave that travels through the material.

A section of the block is considered, it is almost circular with a partial cut. The circular border is considered rigid whereas the cut surface is free. The diameter of the section is 6 cm. The material is graphite; the block is heated by $3.03 \cdot 10^{13}$ electrons deposited in 140ns. A FLUKA binary output file is used to define the energy deposition.

27732 particles are used, and the PUFF material model is used for graphite. The simulation is divided in two parts: the very fast energy deposition that is completed in 140ns and 14 time steps; and the elastic wave propagation simulated for 20µs.

The following output figures show the temperature increase in the section, the pressure value, and the velocity magnitude after 6µs.
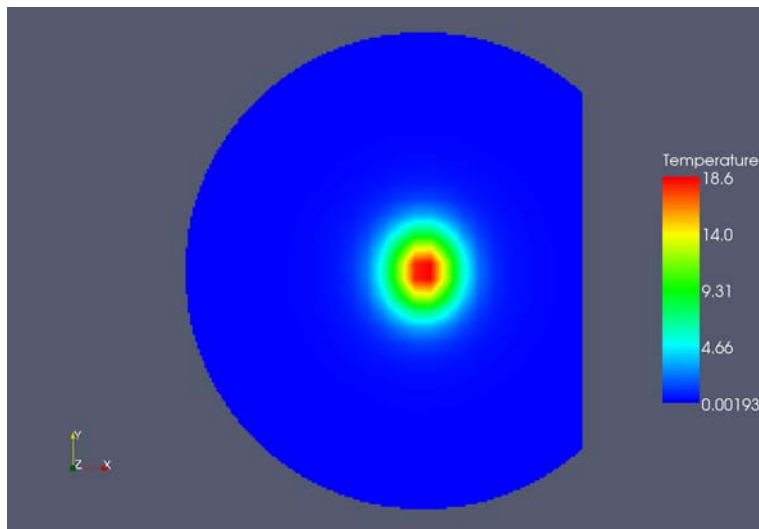
Figure 20: temperature distribution in the section at the end of the beam power deposition
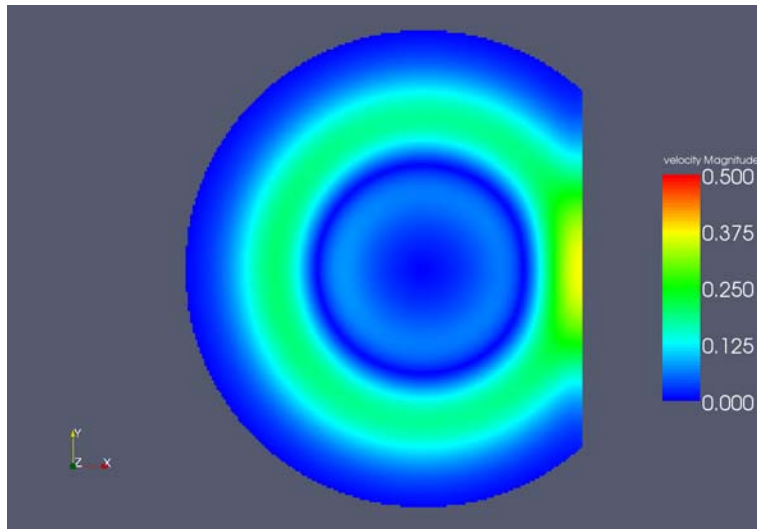
Figure 21: velocity magnitude after 6µs, the elastic wave is reflected on the free surface
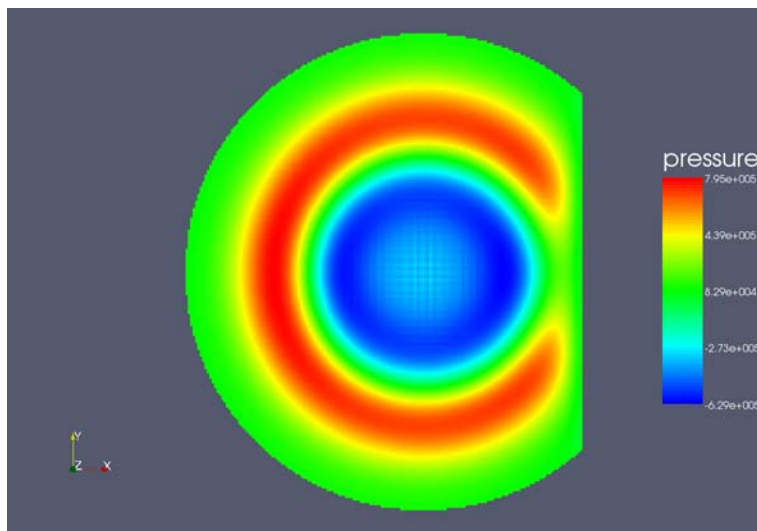

Figure 22: pressure in the section after 6µs, the pressure on the free surface is null

Mercury jet

The second example is 3d and shows how the code can simulate splashing.

Here a flow of liquid mercury with a diameter of 2cm is subject to an high power deposition that rises its temperature of 200K almost instantaneously. This energy is not defined by a FLUKA output file but as an initial condition for the energy field.

50560 particles are used for the simulations and the Shock EOS is used for mercury. The simulation lasts 400µs, a wide splashing in the energy deposition area may be observed.

The following figures show the model at the start and at the end of the simulation.
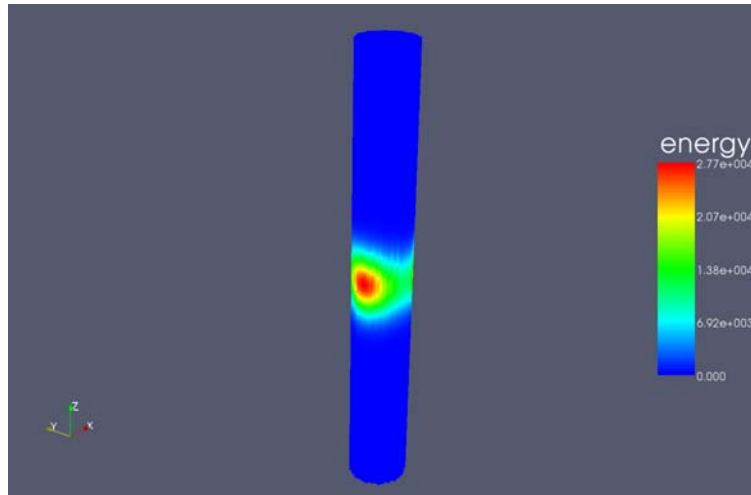
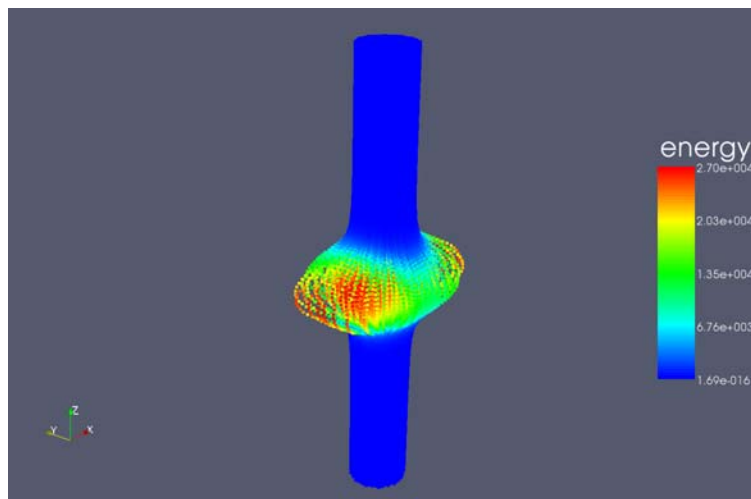**Figure 23: the model immediately after the power deposition is completed**

Figure 24: the model after 400μs

Mercury curtain

The third example is 3d again. As a possible configuration of the mercury target for the EURISOL project, a vertically flowing curtain of mercury has been proposed. This example examines a portion of this curtain, where the maximum energy density is deposited; the model is 4cm wide and 8cm long and high.

The fluid is continuously heated by a proton beam of 4mA depositing a max power of 320 kW/cm^3 given by a FLUKA binning file; 40μs of transient are simulated.

The model is composed by 595508 particles, the Shock EOS is adopted for the mercury with a tensile limit of -1.5bar. Extensive cavitations may be observed.

The following figures show the temperature distribution at the end of the simulation and the pressure wave departing from the heated area at 10μs.
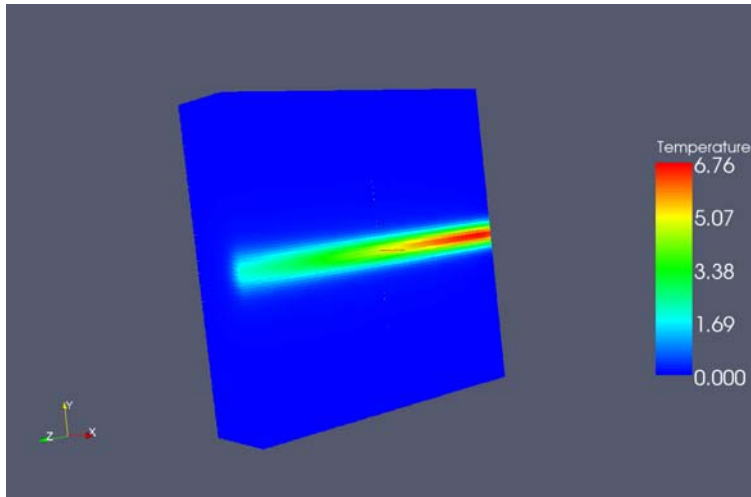
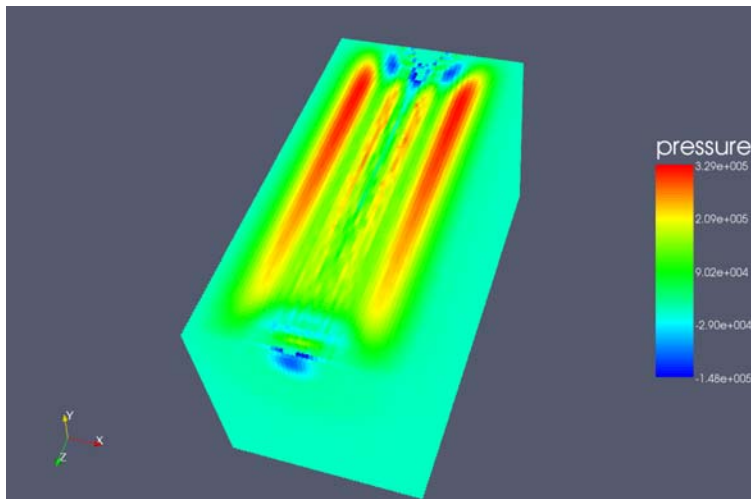**Figure 25: temperature in the model at 20µs**

Figure 26: pressure distribution inside the model at 10µs

Disclaimer

The Armando code and interface is created by the author with the support of CERN. Authorized people may use and improve the code at their own risk and as long as proper reference and acknowledgement are given, and the authors and CERN are notified.

Acknowledgements

References

G.R. Liu and M.B. Liu, Smoothed Particle Hydrodunamics, a meshfree particle method, World Scientific Publishing Co., 2003

A. Colagrossi and M. Landrini, Numerical simulation of interfacial flows by smoothed particle hydrodynamics, Journal of Computational Physics 191 (2003) 448-475

L. Sigalotti, H. Lopez, A. Donoso , E Sira and J. Klapp, A shock capturing SPH scheme based on adaptive kernel estimation, Journal of Computationa Physics 212 (2006) 124-149

G. Viccione, V. Bovolin, E.P. Caratelli, A fast neighbor-search algorithm for free surface flow simulations using SPH, ECCOMAS Thematic Conference on Computational Methods, Rethymo, Crete, Greece, 13-16 June 2007

T. Rabczuk, T. Belytschko and S.P. Xiao, Stable particle methods based on Lagrangian kernels, Computational Methods in Applied Mechanics and Engineering, 193 (2004) 1035-1063

P.W. Randles and L.D. Libersky, Smoothed Particle Hydrodynamics: Some recent improvements and applications, Computational Methods in Applied Mechanics and Engineering, 139 (1996) 375-408

S.Børve, M. Omang and J. Trulsen, Regularized smoothed particle hydrodynamics with improved multi-resolution handling, Journal of Computational Physics, 208 (2005) 345-367

S. Fernández-Méndez, J. Bonet and A. Huerta, Continuous blending of SPH with finite elements, Computers and Structures, 83 (2005) 1448-1458

J. Bonet, S. Kulasegaram, M.X. Rodriguez-Paz and M. Profit, Variational formulation for the smooth particle hydrodynamics (SPH) simulation of fluid and solid problems, Computer Methods in Applied Mechanics and Engineering, 193(2004) 1245-1256

G. Oger, M. Doring, B. Alessandrini and P. Ferrant, Two-dimensional SPH simulations of wedge water entries, Journal of Computational Physics, 213 (2006) 803-822

M.G. Gesteira, B.D. Rogers, R.A. Dalrymple, A.J.C. Crespo and M. Naryanaswamy, User guide to the SPHysics code, 02/2008, http://wiki.manchester.ac.uk/sphysics

Century Dynamics Inc., AUTODYN User Manual, 2007

Paraview-meshless wiki, https://twiki.cscs.ch/twiki/bin/view/ParaViewMeshless/WebHome

E. Noah, L.Bruno, R. Catherall, J. Lettry and T.Stora, Hydrodynamics of ISOLDE liquid metal targets

B.W. Riemer, Benchmarking dynamic strain predictions of pulsed mercury spallation target vessels, Journal of Nuclear Materials, 343 (2005) 81-91

H. Kogawa, S. Hasegawa, M. Futakawa, B. Riemer, M. Wendel and J. Haines, Numerical study on pressure wave propagation in a mercury loop, Journal of Nuclear Materials (2008)

A. Fabich, High Power Proton Beam Shocks and Magnetohydrodynamics in a Mercury Jet Target for a Neutrino Factory, CERN thesis 2002-038