Computer Code

Below follows the presentation of the general SPH computer code, developed for fluid dynamics and transport phenomena simulations, and its routines.

1. The initial positions, velocities, densities, temperatures, support radius and physical properties of the fluid particles are set to the beginning of the simulation. Besides, the boundary condition technique is chosen and the problem geometry is defined.
2. The smoothing function used in SPH interpolations is chosen.
3. Search for neighbouring particles: particles within the domain of influence of a reference particle may vary over time, and the search must be performed at each numerical iteration.
4. Pressure field prediction: in this routine, the prediction of the pressure field acting on the particles is performed. The absolute pressure has two parcels: the hydrostatic pressure (due to the fluid column over the point of the flow considered) and the dynamic pressure (predicted by the equation of state).
5. Mass conservation equation solution: the calculus of the density of each particle is provided by the SPH method.
6. Obtaining the surface forces: approximations to pressure and viscosity forces acting on the particles are obtained.
7. External forces application: the last term on the right side of Eq. (3.38) is the sum of all external forces per unity of mass. In the problems presented in this book, the only external force is gravity. Other forces are considered external forces in the literature such as the repulsive exerted by the virtual particles of type I on the fluid particles (Lennard-Jones molecular force) and free surface forces, if they are employed in the studied problem. However, there is the aforementioned conceptual contradiction of applying molecular concepts in the continuum domain (Sect. 2.1).
8. Momentum balance equation solution: the field of the accelerations of the particles is obtained.

C. A. Dutra Fraga Filho, *Smoothed Particle Hydrodynamics*,
https://doi.org/10.1007/978-3-030-00773-7

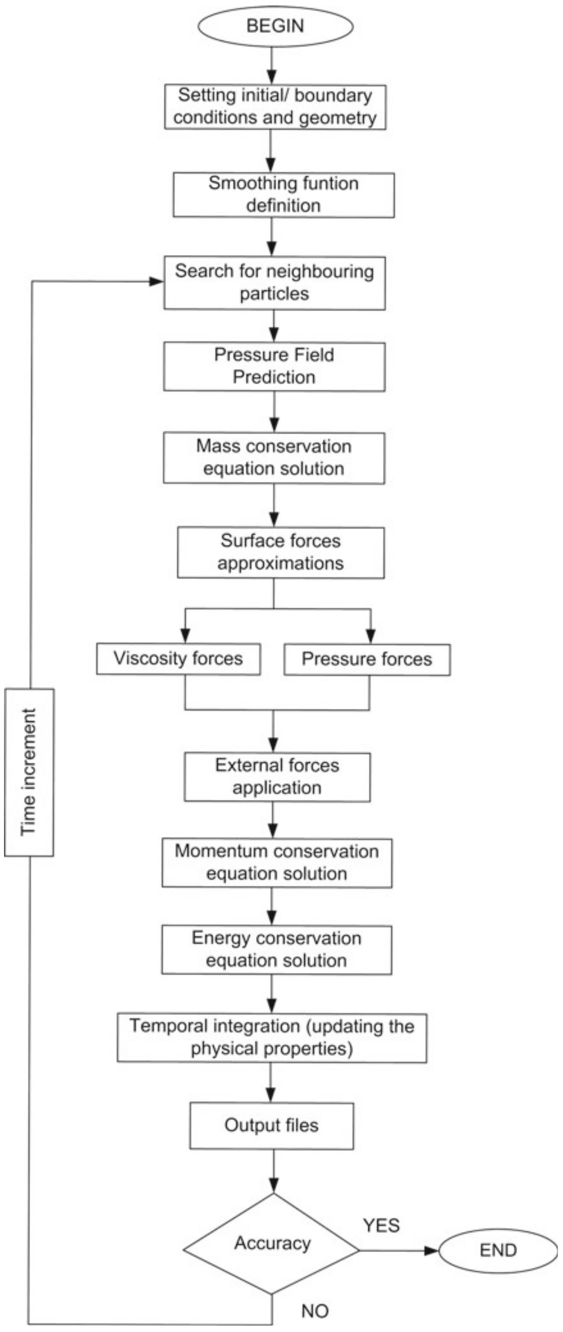

BEGIN
Setting initial/ boundary conditions and geometry
Smoothing funtion definition
Search for neighbouring particles
Pressure Field Prediction
Mass conservation equation solution
Surface forces approximations
Viscosity forces
Pressure forces
External forces application
Momentum conservation equation solution
Energy conservation equation solution
Temporal integration (updating the physical properties)
Output files
Accuracy
YES
END
NO
Time increment

Fig. 1 Flowchart of a general SPH code

9. Energy conservation equation solution: the rate of change of specific energy in time is provided by the SPH method for all particles at domain.
10. Temporal integration: the updating of the properties of the particles (position, velocity, energy and others) is performed.
11. Output files: these files are obtained at the end of each numerical iteration, and from their data, graphical representations of the fluid's physical properties are generated.
12. Accuracy: this routine verifies whether the desirable accuracy has been achieved or if a new iteration will have to be executed. In the latter case, we must return to step 3 (search for the neighbouring particles) and follow the remaining steps until a new verification of the accuracy of numerical results is performed.

In steps 5 and 6, the correction of the physical properties of the particles (density and pressure gradient) located near the boundaries can be made using the Corrective Smoothed Particle Method (CSPM), as explained in Sect. 3.4.1.

The flowchart of the general SPH code is presented in the figure. A more detailed presentation of this code is found in Fraga Filho [1] (Fig. 1).

Conclusion

At the end of this work and in the form of a conclusion, the author would like to offer a few words to the reader.

This book aimed to present the SPH particle method fundamentals and basic applications in continuum mechanics. A presentation of the physical-mathematical modelling was carried out, as well as the SPH method, its consistency, numerical aspects and techniques of treatment of the boundaries and interfaces.

Throughout the presentation of the contents, critical observations on the use of molecular concepts in the continuum scale were performed. The comments, intended for mathematicians, physicists, chemists, engineers, students and researchers in general, are questionings and some conclusions, on the techniques currently employed in the continuum scale.

The limit between the molecular and continuous scales and, consequently, the validity of microscopic concepts and models employed in the treatment of macroscopic problems were discussed. Without the desire of simply criticizing solutions already presented in the literature, alternative solutions, faithful to the concepts of the continuum mechanics, have been proposed and discussed.

In recent years, a great scientific effort is being made aiming to bridge the gap between molecular and continuum approaches, in order to solve problems in minor scales. The understanding and modelling of many phenomena require the development of studies in a challenging interface area, where occurs the intersection of multiple disciplines, such as physics, chemistry, material sciences and biology [2].

Hybrid molecular continuum, multiscale [3,4] and dissipative particle dynamics (DPD) models [5,6] are recent computational methods that couple the events in the continuum and molecular/atomistic domains.

In the nanoscopic scale, the fluctuations in the averaged properties (due to the motion and behaviour of individual particles) begin to have a significant effect on the behaviour of a system and must be taken into account in the context of any particular problem. Scientific challenges appear when moving away from the macroscopic domain and approaching the molecular scale. The behaviour of nanoparticles is not only different from macroscopic particles, but they do not obey the same physical laws. A mesoscopic region separates the laws of continuum mechanics and quantum physics. A gap has been left in the full range of scaling from macro to nano. Reference [7] makes a basic presentation on the mesoscale and its characteristics.

The problems discussed in this study do not involve the two scales (continuum and molecular); on the contrary, they are defined in the macroscopic scale. In physical and engineering problems defined in the continuum scale, these models cannot be employed. It is always necessary to obey the classical continuum mechanics laws and not to mix them with the laws governing the molecular scale (quantum physics and molecular dynamics). The computational solutions obtained in the continuum domain using fictitious particles/artificial repulsive forces are not related to hybrid or multiscale methods. They are only results obtained from purely computational techniques in which the empirical calibration of parameters, such as those in Lennard-Jones molecular force (Eq. (3.113)), is performed.

In the process of implementing the numerical code used for the simulation of oil spreading in the calm sea (section X), we adopted the methodology to verify the results provided by the SPH approximation for each term of the physical conservation equations from the comparison with analytical results, experimental or existing in the literature, on the three problems studied in previous stages of the study of oil spreading. That is, the conclusions and discussions presented in this book are based on the theoretical foundations of classical physics and continuous mechanics, but also the computational implementation and verification of the attainable results in each stage of scientific research. The author feels obliged to present this clarification, out of respect for those who develop research on methods of particles applied in the domain of the continuum.

I sincerely hope that scientific research continues in its progress of proposing solutions to the problems that arise in the most diverse knowledge areas. But I also hope that answers to the questions and doubts that wander in the minds of scientists, students and researchers are sought, taking into account that the problems must be correctly modelled and the results achieved are consistent with the physical or chemical phenomena studied.

May this work be an incentive to the incessant search for answers, which is the essence of the scientific advancement.

Appendix A Smoothing Functions, Derivatives and Normalization Constants

In this additional material, the interpolation functions used in numerical simulations, their derivatives and normalization constants, which satisfy the condition of unity (Sect. 3.1), will be presented.

- **Lucy's Quartic Kernel** (Fig. A.1)

In the polar coordinate system:

$$W(q,h) = \alpha_D \begin{cases} (1+3q)\left(1-q^3\right), & 0 \le q \le 1 \\ 0, & \text{in the other case.} \end{cases}$$

where

r is the position of the fixed point
r′ is the position of the variable point

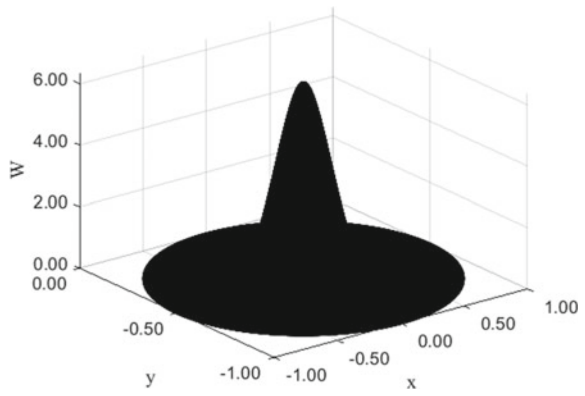


Fig. A.1 Lucy's quartic kernel in a two-dimensional domain

$q = \frac{\left|(\mathbf{r}-\mathbf{r}', h)\right|}{h}$ and α_D is the kernel's normalization constant.

Derivative:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D\left(-12q + 24q^2 - 12q^3\right)$$

Normalization Constants:

(1) In a 2-D Domain:

$0.0 \leq q \leq 1.0$:

$$\int_{\Omega} W d\Omega = 1 \text{ (Unity property)}$$

$$\alpha_D \int_0^{2\pi}\int_0^{h} (1+3q)(1-q)^3 r dr d\theta = 1\cdot$$

$$\therefore \boxed{\alpha_D = \frac{5}{\pi h^2}}$$

(2) In a 3-D Domain:

Assuming a spherical coordinate system and integrating in the first octant, we have:

$0.0 \leq q \leq 1.0$:

$$\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_0^{h} (1+3q)\,(1-q)^3 r^2 \sin\theta dr d\theta d\phi = \frac{2\pi h^3}{105}$$

In the whole domain of influence (all octants):

$$\int_{\Omega} W d\Omega = 1 \text{ (Unity property)}$$

$$\int_{\Omega} W d\Omega = \alpha_D\left[8\left(\frac{2\pi h^3}{105}\right)\right] = 1$$

$$\therefore \boxed{\alpha_D = \frac{105}{16\pi h^3}}$$

- **Cubic Spline Kernel** (Fig. A.2)

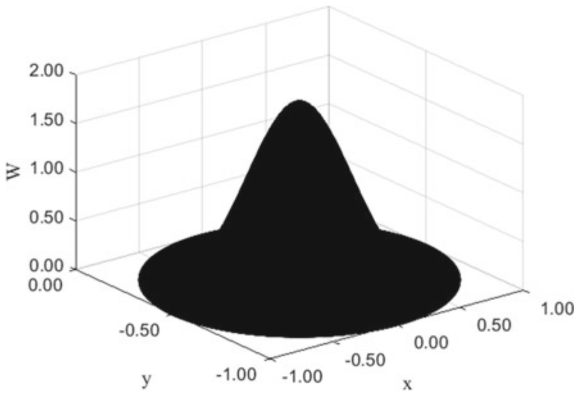


Fig. A.2 Cubic spline kernel in a two-dimensional domain

$$W(q,h) = \alpha_D \begin{cases} \left(\frac{2}{3} - q^2 + \frac{1}{2}q^3\right), & 0 \le q \le 1 \\ \left[\frac{1}{6}(2-q)^3\right], & 1 < q \le 2 \\ 0, & \text{in the other case.} \end{cases}$$

Derivatives:

$\mathbf{0.0 \le q \le 1.0}$:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D\left(-2q + \frac{3q^2}{2}\right)$$

$\mathbf{1.0 < q \le 2.0}$:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D\left(\frac{-(2-q)^2}{2}\right)$$

Normalization Constants:

(1) In a 2-D Domain:

$\mathbf{0.0 \le q \le 1.0}$:

$$\int_0^{2\pi}\int_0^{h}\left(\frac{2}{3} - q^2 + \frac{1}{2}q^3\right) r dr d\theta = \frac{11\pi h^2}{30}$$

$\mathbf{1.0 < q \le 2.0}$:

$$\int_0^{2\pi}\int_h^{2h}\frac{1}{6}(2-q)^3\, r dr d\theta = \frac{3\pi h^2}{30}$$

In the whole domain of influence:

$$\int_{\Omega} W d\Omega = 1 \text{ (Unity property)}$$

$$\int_{\Omega} W d\Omega = \alpha_D \left(\frac{11\pi h^2}{30} + \frac{3\pi h^2}{30} \right) = 1$$

$$\therefore \boxed{\alpha_D = \frac{15}{7\pi h^2}}$$

(2) In a 3-D Domain:

Integrating in the first octant:

$0.0 \leq q \leq 1.0$:

$$\int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} \int_0^{h} \left(\frac{2}{3} - q^2 + \frac{1}{2} q^3 \right) r^2 \sin\theta dr d\theta d\phi = \frac{19\pi h^3}{360}$$

$1.0 < q \leq 2.0$:

$$\int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} \int_h^{2h} \frac{1}{6} (2-q)^3 r^2 \sin\theta dr d\theta d\phi = \frac{11\pi h^3}{360}$$

In the whole domain of influence (all octants):

$$\int_{\Omega} W d\Omega = 1 \text{ (Unity property)}$$

$$\int_{\Omega} W d\Omega = \alpha_D \left[8 \left(\frac{19\pi h^3}{360} + \frac{11\pi h^3}{360} \right) \right] = 1$$

$$\therefore \boxed{\alpha_D = \frac{3}{2\pi h^3}}$$

- **New Quartic Kernel** (Fig. A.3)

$$W(\mathbf{r} - \mathbf{r}', h) = \alpha_D \begin{cases} \left(\frac{2}{3} - \frac{9}{8} q^2 + \frac{19}{24} q^3 - \frac{5}{32} q^4 \right), & 0 \leq q \leq 2 \\ 0, & \text{in the other case.} \end{cases}$$

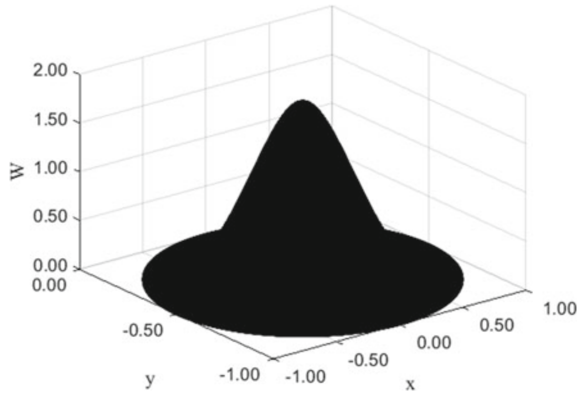


Fig. A.3 New quartic kernel in a two-dimensional domain

Derivative:

0.0 ≤ q ≤ 2.0:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D \left(-\frac{18q}{8} + \frac{57q^2}{24} - \frac{20q^3}{32} \right)$$

Normalization Constants:

(1) In a 2-D Domain:

0.0 ≤ q ≤ 2.0:

$$\int_0^{2\pi} \int_h^{2h} \left(\frac{2}{3} - \frac{9}{8}q^2 + \frac{19}{24}q^3 - \frac{5}{32}q^4 \right) r\,dr\,d\theta = \frac{7\pi h^2}{15}$$

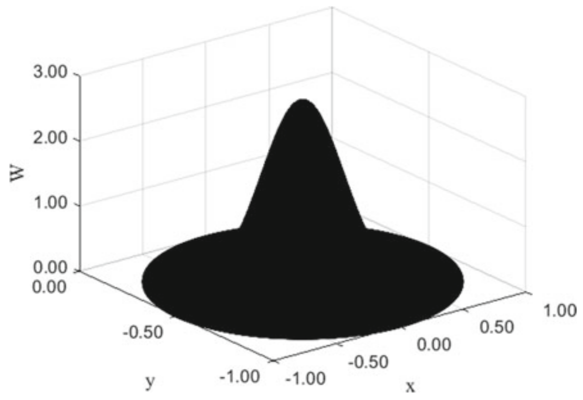
In the whole domain of influence:

$$\int_\Omega W d\Omega = 1 \text{ (Unity property)}$$

$$\int_\Omega W d\Omega = \alpha_D \left(\frac{11\pi h^2}{30} + \frac{3\pi h^2}{30} \right) = 1$$

$$\therefore \boxed{\alpha_D = \frac{15}{7\pi h^2}}$$

Fig. A.4 Quintic spline kernel in a two-dimensional domain



(2) In a 3-D Domain:

Integrating in the first octant:

$0.0 \le q \le 2.0$:

$$\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_0^{2h}\left(\frac{2}{3}-\frac{9}{8}q^2+\frac{19}{24}q^3-\frac{5}{32}q^4\right)r^2\sin\theta\, dr\, d\theta\, d\phi = \frac{52}{630\pi h^3}$$

In the whole domain of influence (all octants):

$$\int_\Omega W d\Omega = 1 \text{ (Unity property)}$$

$$\int_\Omega W d\Omega = \alpha_D\left[8\left(\frac{416\pi h^2}{630}\right)\right] = 1$$

$$\therefore \boxed{\alpha_D = \frac{315}{208\pi h^3}}$$

- **Quintic Spline Kernel** (Fig. A.4)

$$W(\mathbf{r}-\mathbf{r}', h) = \alpha_D \begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5, & 0 \le q \le 1 \\ (3-q)^5 - 6(2-q)^5, & 1 < q \le 2 \\ (3-q)^5, & 2 < q \le 3 \\ 0, & \text{in the other case.} \end{cases} \tag{A.1}$$

Derivatives:

0.0 ≤ q ≤ 1.0:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D \left(-120q + 120q^3 - 50q^4\right)$$

1.0 < q ≤ 2.0:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D \left(75 - 420q + 450q^2 - 180q^3 + 25q^4\right)$$

2.0 < q ≤ 3.0:

$$\frac{\partial W(q,h)}{\partial q} = \alpha_D \left(405 + 540q - 270q^2 + 60q^3 - 5q^4\right)$$

Normalization Constants:

(1) In a 2-D Domain:

0.0 ≤ q ≤ 1.0:

$$\int_0^{2\pi}\int_0^{h} \left[(3-q)^5 - 6(2-q)^5 + 15(1-q)^5\right] r\,dr\,d\theta = \frac{302\pi h^2}{7}$$

1.0 ≤ q ≤ 2.0:

$$\int_0^{2\pi}\int_h^{2h} \left[(3-q)^5 - 6(2-q)^5\right] r\,dr\,d\theta = \frac{171\pi h^2}{7}$$

2.0 ≤ q ≤ 3.0:

$$\int_0^{2\pi}\int_{2h}^{3h} (3-q)^5 r\,dr\,d\theta = \frac{5\pi h^2}{7}$$

In the whole domain of influence:

$$\int_\Omega W d\Omega = 1 \text{ (Unity property)}$$

$$\int_\Omega W d\Omega = \alpha_D \left(\frac{302\pi h^2}{7} + \frac{171\pi h^2}{7} + \frac{5\pi h^2}{7}\right) = 1$$

$$\therefore \boxed{\alpha_D = \frac{7}{478\pi h^2}}$$

(2) In a 3-D Domain:

Integrating in the first octant:

$\mathbf{0.0 \leq q \leq 1.0}$:

$$\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_0^{h}\left[(3-q)^5-6(2-q)^5+15(1-q)^5\right]r^2\sin\theta\, dr\, d\phi=\frac{730\pi h^3}{14}$$

$\mathbf{1.0 \leq q \leq 2.0}$:

$$\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_h^{2h}\left[(3-q)^5-6(2-q)^5\right]r^2\sin\theta\, dr\, d\phi=\frac{907\pi h^3}{14}$$

$\mathbf{2.0 \leq q \leq 3.0}$:

$$\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_{2h}^{3h}(3-q)^5 r^2\sin\theta\, dr\, d\phi=\frac{43\pi h^3}{14}$$

In the whole domain of influence (all octants):

$$\int_\Omega W d\Omega = 1 \text{ (Unity property)}$$

$$\int_\Omega W d\Omega = \alpha_D\left[8\left(\frac{730\pi h^3}{14}+\frac{907\pi h^3}{14}+\frac{43\pi h^3}{14}\right)\right]=1$$

$$\therefore \boxed{\alpha_D=\frac{1}{120\pi h^3}}$$

Comments

The integration has been performed over the domain of influence with a unity radius (kh = 1.0, where k is a scaling factor that depends on the smoothing function).

The scaling factor divides the domain of influence in regions, in which the smoothing function presents different mathematical expressions, and defines the number of regions on which the integration is performed in order to obtain the normalization constant.

The unity property does not depend on the length of the support radius (kh).

The kernel's unit, not shown in the deductions, is the inverse unit of volume.

Appendix B
Deduction of the SPH Laplacian Operator

In this appendix, the mathematical deduction of the Laplacian of a function (in a 2-D domain) using the SPH method will be presented.

The expression obtained has been used to approximate the terms of viscous forces in the momentum conservation equation and the Laplacian of the temperature (at the study of the heat diffusion in a flat plate).

For the approximation of the Laplacian of a function, the expansion of the Taylor series was used. In a two-dimensional domain, it is possible to determine the value of a function at a point $X^{'} = (\mathrm{x}^{'}, \mathrm{y}^{'})$, around a fixed point $X = (\mathrm{x}, \mathrm{y})$:

$$f\left(\mathrm{x}^{'}, \mathrm{y}^{'}\right) = f\,(\mathrm{x}, \mathrm{y}) + \left(\mathrm{x}^{'} - \mathrm{x}\right) \frac{\partial f}{\partial \mathrm{x}}\bigg|_{(\mathrm{x},\mathrm{y})} +$$

$$\left(\mathrm{y}^{'} - \mathrm{y}\right) \frac{\partial f}{\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})} + \frac{1}{2}\left[\left(\mathrm{x}^{'} - \mathrm{x}\right)^2 \frac{\partial^2 f}{\partial \mathrm{x}^2}\bigg|_{(\mathrm{x},\mathrm{y})} +\right.$$

$$\left.\left(\mathrm{y}^{'} - \mathrm{y}\right)^2 \frac{\partial^2 f}{\partial \mathrm{y}^2}\bigg|_{(\mathrm{x},\mathrm{y})}\right] + \left(\mathrm{x}^{'} - \mathrm{x}\right)\left(\mathrm{y}^{'} - \mathrm{y}\right) \frac{\partial^2 f}{\partial \mathrm{x}\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})} +$$

$$R_n\left(X^{'} - X\right) \tag{B.1}$$

where $R_n\left(X^{'} - X\right)$ is the remainder of the Taylor series.

Assuming an error of order 3 in the approximation, multiplying Eq. (B.1), truncated, by $\frac{\left(X - X^{'}\right)}{\left|X - X'\right|^2} \nabla W(X - X^{'}, h)$, and integrating it:

$$\int_{\Omega} \overbrace{f\left(\mathrm{x}^{'}, \mathrm{y}^{'}\right) \Delta X \cdot \nabla W(X - X', h) dX'}^{A} =$$

$$\int_{\Omega}\overbrace{f\,(\mathrm{x},\mathrm{y})\,\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{B}+\int_{\Omega}\overbrace{\left(\mathrm{x}^{'}-\mathrm{x}\right)\frac{\partial f}{\partial \mathrm{x}}\bigg|_{(\mathrm{x},\mathrm{y})}\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{C}+$$

$$\int_{\Omega}\overbrace{\left(\mathrm{y}^{'}-\mathrm{y}\right)\frac{\partial f}{\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})}\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{D}+$$

$$\int_{\Omega}\overbrace{\frac{1}{2}\left(\left(\mathrm{x}^{'}-\mathrm{x}\right)^{2}\frac{\partial^{2} f}{\partial \mathrm{x}^{2}}\bigg|_{(\mathrm{x},\mathrm{y})}+\left(\mathrm{y}^{'}-\mathrm{y}\right)^{2}\frac{\partial^{2} f}{\partial \mathrm{y}^{2}}\bigg|_{(\mathrm{x},\mathrm{y})}\right)\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{E}+$$

$$\int_{\Omega}\overbrace{\frac{1}{2}\left(\mathrm{x}^{'}-\mathrm{x}\right)\left(\mathrm{y}^{'}-\mathrm{y}\right)\frac{\partial^{2} f}{\partial \mathrm{x}\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})}\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{F} \tag{B.2}$$

where $\Delta X=\dfrac{\left(X-X^{'}\right)}{\left|X-X'\right|^{2}}$.

The gradient of the kernel has the properties of anti-symmetry and non-normalization in the Euclidean coordinate system:

$$\boxed{\int_{\Omega}\left(X^{'}-X\right)_{p}\Delta X\cdot\nabla W\left(X^{'}-X,h\right)dX^{'}=0} \tag{B.3}$$

$$\boxed{\int_{\Omega}\left(X^{'}-X\right)_{l}\left(X^{'}-X\right)_{t}\Delta X\cdot\nabla W\left(X^{'}-X,h\right)dX^{'}=-\delta_{l,t}} \tag{B.4}$$

where $l=(1,2)$, $t=(1,2)$, $p=(1,2)$, $\left(X^{'}-X\right)_{1}=\left(\mathrm{x}^{'}-\mathrm{x},0\right)$, $\left(X^{'}-X\right)_{2}=\left(0,\mathrm{y}^{'}-\mathrm{y}\right)$ and $\delta_{l,t}$ is the Kronecker delta function.

Applying the properties of the kernel, we conclude:

$$\int_{\Omega}\overbrace{\left(\mathrm{x}^{'}-\mathrm{x}\right)\frac{\partial f}{\partial \mathrm{x}}\bigg|_{(\mathrm{x},\mathrm{y})}\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{C}=0, \tag{B.5}$$

$$\int_{\Omega}\overbrace{\left(\mathrm{y}^{'}-\mathrm{y}\right)\frac{\partial f}{\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})}\Delta X\cdot\nabla \mathrm{W}(\mathrm{X}-\mathrm{X}^{'},h)\mathrm{dX}'}^{D}=0, \tag{B.6}$$

$$\int_{\Omega} \overbrace{\frac{1}{2}\left(\left(\mathrm{x}' - \mathrm{x}\right)^2 \frac{\partial^2 f}{\partial \mathrm{x}^2}\bigg|_{(\mathrm{x},\mathrm{y})} + \left(\mathrm{y}' - \mathrm{y}\right)^2 \frac{\partial^2 f}{\partial \mathrm{y}^2}\bigg|_{(\mathrm{x},\mathrm{y})}\right) \Delta X \cdot \nabla \mathrm{W}(\mathrm{X} - \mathrm{X}', h)\mathrm{dX}'}^{E} =$$

$$-\frac{1}{2}\left(\frac{\partial^2 f}{\partial \mathrm{x}^2} + \frac{\partial^2 f}{\partial \mathrm{y}^2}\right)\bigg|_{(\mathrm{x},\mathrm{y})}, \tag{B.7}$$

$$\int_{\Omega} \overbrace{\left(\mathrm{x}' - \mathrm{x}\right)\left(\mathrm{y}' - \mathrm{y}\right) \frac{\partial^2 f}{\partial \mathrm{x}\partial \mathrm{y}}\bigg|_{(\mathrm{x},\mathrm{y})} \Delta X \cdot \nabla \mathrm{W}(\mathrm{X} - \mathrm{X}', h)\mathrm{dX}'}^{F} = 0, \tag{B.8}$$

$$\boxed{\nabla^2 f\Big|_{(\mathrm{x},\mathrm{y})} = \left(\frac{\partial^2 f}{\partial \mathrm{x}^2} + \frac{\partial^2 f}{\partial \mathrm{y}^2}\right)\bigg|_{(\mathrm{x},\mathrm{y})} = -2\int_{\Omega} f[\left(\mathrm{x}', \mathrm{y}'\right) - f(\mathrm{x}, \mathrm{y})]\Delta X \cdot \nabla W\left(X - X', h\right) dX'} \tag{B.9}$$

where $\nabla^2 f\big|_{(\mathrm{x},\mathrm{y})}$ is the Laplacian of the function f, evaluated at the position X.

After the domain discretization by particles, we arrive at the following expression for the SPH approximation of the Laplacian of the function f in the position occupied by the fixed particle i:

$$\boxed{\nabla^2 f_i = \left(\frac{\partial^2 f_i}{\partial \mathrm{x}^2} + \frac{\partial^2 f_i}{\partial \mathrm{y}^2}\right) = 2\sum_{j=1}^{n} \frac{m_j}{\rho_j}\left(f_i - f_j\right)\Delta X_{ij} \cdot \nabla W\left(X_i - X_j, h\right)} \tag{B.10}$$

where

i is the fixed particle, where the Laplacian of the function f is being evaluated
j is each of the neighbouring particles of the fixed particle
$\nabla^2 f_i$ is the SPH approximation of the Laplacian of the function f evaluated at the position occupied by the fixed particle $\Delta X_{ij} = \dfrac{\mathrm{X}_i - \mathrm{X}_j}{\left|\mathrm{X}_i - \mathrm{X}_j\right|^2}$.

In the polar coordinate system, the expression of the SPH Laplacian becomes:

$$\boxed{\nabla^2 f_i = 2\sum_{j=1}^{n} \frac{m_j}{\rho_j}\left(f_i - f_j\right)\frac{\partial W(\mathbf{r}_i - \mathbf{r}_j, h)}{\partial r}\frac{1}{\left|\mathbf{r}_{ij}\right|}} \tag{B.11}$$

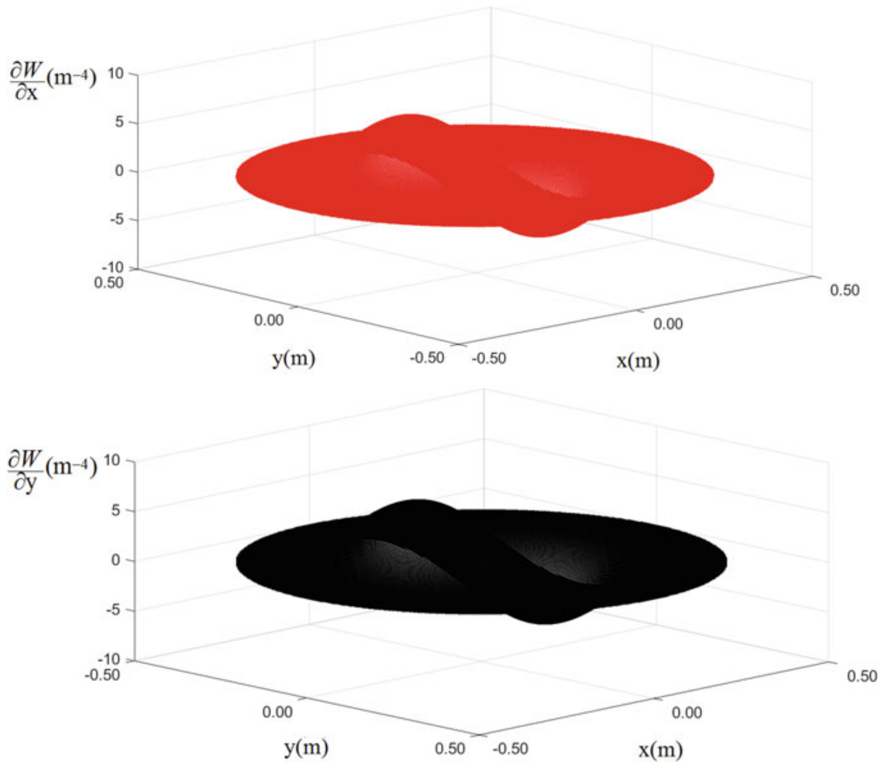


Fig. B.1 First derivatives of the kernel

where

r is the radial direction
$W(\mathbf{r}_i - \mathbf{r}_j, h)$ is the smoothing function written in polar coordinates evaluated at the position $(\mathbf{r}_i - \mathbf{r}_j)$
$|\mathbf{r}_{ij}| = |(\mathbf{r}_i - \mathbf{r}_j)|$ is the radial distance between the fixed and a neighbouring particle.

Figures B.1 and B.2 present the graphs of the kernel's derivatives in the following order: first derivatives (allowing the visualization of the anti-symmetry property) and second radial derivative. The kernel used was the cubic spline, but the behaviour of the graphs is the same for most of the smoothing functions used in the SPH method.

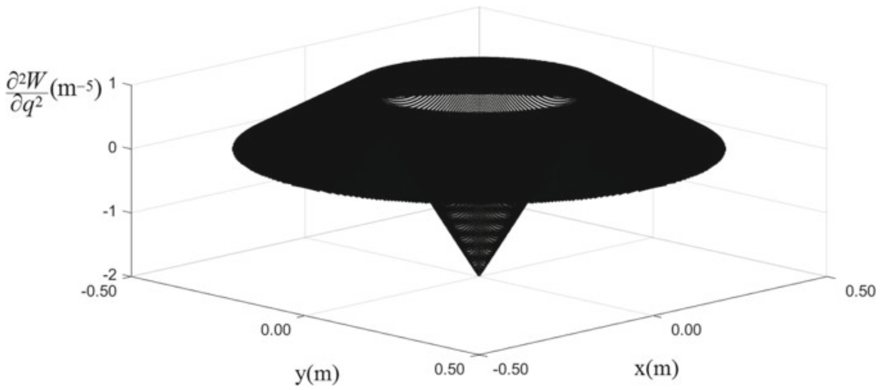


Fig. B.2 Second radial derivative of the kernel

FORTRAN Source Files

Heat Diffusion in a Homogeneous Flat Plate

The following are the open-source code (in FORTRAN Language) and a script in the MATLAB language for plotting the temperature images. Instructions for creating the executable program are provided at the beginning of the document.

```
PROGRAM flat_plate_diffusion

!-----------------------------------------------------------------
! These FORTRAN source files have been used in heat diffusion in a
! homogeneous flat plate (Sect. 4.2).
!-----------------------------------------------------------------
! The reader should follow the instructions below in order to
! create the program executable:
! 1. Within the working directory, create and open a new file in
! the FORTRAN Editor˜
! 2. Copy and paste this PROGRAM and save as ‘‘name.f90’’
! 3. Create a folder called output within the working directory
! 4. Create a subfolder called temperature within the folder output
! 5. Compile and Run the program
! Output: in the file called TEMPERATURE_DIFFERENCES.DAT are the
! points where the smallest and the largest temperature differences
! occur (comparing the SPH results and solution provided by series).
!
!-----------------------------------------------------------------
! VARIABLES:
! aux_p - variable used in memory allocation
! dist - distance between the centres of mass of a fixed and a
! neighbour particle
! dt - time step (in seconds)
! dx - horizontal distance between the centres of mass of
! particles
! dy - vertical distance between the centres of mass of
! particles
! eps - accuracy of simulation
! finish - variable used to record the simulation time
! hsml - smoothing lengths of particles
! int_function - interpolation function (kernel)
! itimestep - current iteration
! LX - length of the plate
! LY - width of the plate
! mass - mass of particles
! n_bound - number of boundary particles
! n_cells - number of cells in each Cartesian direction
! neighbour - matrix containing the neighbours of each particle
! of the domain
! n_global - sum of the number of particles at domain and
! boundary particles
! np_side - number of particles per side of the plate
! ntotal - total number of particles at domain
! response - variable used to verify the achievement of accuracy
! rho - densities of particles
! start - variable used to record the simulation time
! step_out - step for recording results
! support_radius - radius of the domain of influence
! time - simulation time
! T_0 - initial temperatures of particles
! T - temperature provided by SPH method at each numerical
! iteration
! TE - prescribed temperature at right side of the plate
! TN - prescribed temperature at top of the plate
! TS - prescribed temperature at bottom of the plate
! TW - prescribed temperature at left side of the plate
! temp_Laplacian - SPH Laplacian of temperature
! x - coordinates of particles
!-----------------------------------------------------------------
IMPLICIT NONE

INTEGER, DIMENSION (:,:), allocatable :: neighbour

DOUBLE PRECISION, DIMENSION (:), allocatable :: rho, mass, hsml, T_0, T,
temp_Laplacian
DOUBLE PRECISION, DIMENSION (:,:), allocatable :: x, dist, dx, dy
INTEGER int_function, aux_p, ntotal, itimestep, d, m, i, hours, minutes,
hours_seg, np_side, n_global, n_cells, n_bound, step_out
DOUBLE PRECISION TS, TN, TW, TE, LX, LY, xl, yl, factor
DOUBLE PRECISION start, finish, time, seconds, dt, eps, support_radius
CHARACTER(LEN=15) response

CALL cpu_time(start)

OPEN(01,file="output/domain_particles_positions.dat")
OPEN(02,file="output/boundary_particles_positions.dat")
OPEN(03,file="output/initial_temperature.dat")

!------------------------Plate geometry---------------------------
LX=1.0
LY=1.0
!-----------------------------------------------------------------

!-----------------------------------------------------------------
!---------------DEFINING THE TEMPERATURES AT BOUNDARIES-----------
```

```
!-------------------(DIRICHLET BOUNDARY CONDITIONS)----------------
!-----------------------------------------------------------------

TS=100.0 !Temperature at the lower boundary
TN=0.0 !Temperature at the upper side boundary
TW=0.0 !Temperature at the left side boundary
TE=0.0 !Temperature at the right side boundary

!--------------------Time step and Accuracy------------------------
dt = 1.e-5
eps = 1.e-6
!-----------------------------------------------------------------

print*, ' '
print*, 'Enter with the number of particles per side of the plate:'
print*, ' '
read*, np_side
print*, ' '
print*, 'Choose the interpolation Function (Kernel):'
print*, ' '
print*, '1 - Lucy''s Quartic Kernel'
print*, '2 - Cubic Spline Kernel'
print*, '3 - New Quartic Kernel'
print*, '4 - Quintic Spline Kernel'
print*, ' '
read*, int_function
print*, ' '
print*, 'Enter with the step for the output files:'
print*, ' '
read*, step_out

!Distance between the centres of mass

aux_p = np_side**2 + 1500 !total number of particles (for memory
allocation)

allocate (neighbour(aux_p,150), rho(aux_p), hsml(aux_p),
T_0(aux_p), T(aux_p), temp_Laplacian(aux_p), x(aux_p,2),
dist(aux_p,150), dx(aux_p,150), dy(aux_p,150), mass(aux_p))

response='non_convergence'
itimestep=1

print*, ' '
print*,'------------------------------------------------------'
print*,'--FLAT PLANE DIFFUSION SIMULATION USING THE SPH METHOD-'
print*,'------------------------------------------------------'
print*, ' '

dx = LX/np_side
dy = LY/np_side

CALL temperature_series(aux_p, np_side, TS, TN, TW, TE, dx, dy, dt)

CALL input(aux_p, x, mass, rho, np_side, ntotal, T_0, LX, LY,
support_radius, n_cells)
```

```
CALL boundary_particles(aux_p, itimestep, np_side, TS, TN, TW, TE, LX, LY,
dx, dy, n_bound, ntotal, n_global, mass,x,rho,T_0)

CALL neighbour_search(aux_p, x, n_global, ntotal, n_bound, neighbour, dx,
dy, support_radius, dist)

!-----------------------------------------------------------------
!--------------------UNTIL THE ACCURACY IS ACHIEVED---------------
!-----------------------------------------------------------------

DO WHILE (response.eq.'non_convergence')

  CALL SPH_Temperature_Laplacian (aux_p, int_function, factor, hsml,
ntotal, dist, mass, rho, support_radius, neighbour, T_0, temp_Laplacian,
itimestep)

  CALL temporal_integration(aux_p, itimestep, x, ntotal, dt,
temp_Laplacian, T_0, T, step_out)

  CALL convergence (aux_p, itimestep,T_0,T,eps,response, ntotal)

  CALL temperature_differences(itimestep, dt, ntotal, step_out)

  IF (response.eq.'convergence') THEN
    PRINT*, ' ACCURACY ACHIEVED IN THE ', itimestep, 'a. ITERATION'
    PRINT*, ' END OF SIMULATION '
  END IF

END DO

CALL cpu_time(finish)

time=finish-start
hours_seg=3600; !hours em seconds
hours = (time/hours_seg) !resultado da hora
minutes = (time -(hours_seg*hours))/60
seconds = (time -(hours_seg*hours)-(minutes*60))
print*,' '
print*, ' ---------------------------------------------------'
WRITE (*,*) 'TIME PROCESSING (CPU TIME) '
WRITE (*,*) hours ,' hours', minutes,' minutes', seconds, ' seconds.'
print*, ' ---------------------------------------------------'

deallocate(neighbour, rho, hsml, T_0, T, temp_Laplacian, x, dist, dx, dy, mass)

OPEN(776,file= 'output/SIMULATION_PARAMETERS.DAT')
REWIND(776)
WRITE (776,*) ' '
IF (int_function.EQ.1) THEN !--------LUCY'S QUARTIC KERNEL----
  WRITE (776,*) 'Interpolation Function: Lucy''s Quartic Kernel'
  WRITE(776,*) ' '
ELSE
  IF (int_function.EQ.2) THEN !------CUBIC SPLINE KERNEL-------
   WRITE (776,*) 'Interpolation Function: Cubic Spline Kernel'
    WRITE(776,*) ' '
  ELSE
    IF (int_function.EQ.3) THEN !----NEW QUARTIC KERNEL--------
```

```
      WRITE (776,*) 'Interpolation Function: New Quartic Kernel'
      WRITE(776,*) ' '
    ELSE
      IF (int_function.EQ.4) THEN !-----QUINTIC SPLINE KERNEL---
        WRITE (776,*) 'Interpolation Function: Quintic Spline Kernel'
        WRITE(776,*) ' '
      END IF
    END IF
  END IF
END IF

OPEN(25,FILE='NUMBER_OF_ITERATIONS.DAT')
WRITE(25,*) itimestep
 CLOSE(25)

WRITE (776,*) 'Number of particles per side of plate = ', np_side
WRITE (776,*) ' '
WRITE (776,*) 'Time step (dt, em seconds) = ', dt
WRITE (776,*) ' '
WRITE (776,*) 'Accuracy ( abs(T(m+1) - T(m)) ) = ', eps
WRITE (776,*) ' '
WRITE (776,*) 'Time for the physical diffusion =', itimestep*dt
WRITE (776,*) ' '
WRITE (776,*) 'Time processing (CPU time) = ', hours ,' hours',
minutes,' minutes', seconds, ' seconds.'

 CLOSE(01)
 CLOSE(02)
 CLOSE(03)
 CLOSE(776)

END PROGRAM

!******************* SOLUTION BY SERIES ***********************

SUBROUTINE temperature_series(aux_p, np_side, TS, TN, TW, TE, dx, dy, dt)

!---------------------------------------------------------------
! This subroutine defines the positions of the centres of mass of
! particles at domain and makes the calculation of the
! temperatures at steady-state using the solution by series
!
! x - coordinates of particles
! mass - mass of particles
! KMAX - number of terms in series
!---------------------------------------------------------------

IMPLICIT NONE

DOUBLE PRECISION, DIMENSION (:,:), ALLOCATABLE :: T , X1
DOUBLE PRECISION, DIMENSION (:), ALLOCATABLE :: POSX, POSY, TEMP
DOUBLE PRECISION LX, LY, dx, dy, aux, TS, TN, TW, TE, X, Y, SINX, Txy, PI,
dt
INTEGER aux_p, np_side, I, d, J, K, MP1, NP1, KMAX, cont, n_bound

MP1=np_side + 1
NP1=np_side + 1
```

```
ALLOCATE( T(MP1,NP1), POSX(aux_p), POSY(aux_p), TEMP(aux_p), X1(1500,3))

PI=ACOS(-1.)
KMAX=90

OPEN(UNIT=10,FILE='output/TEMPERATURES.DAT')
OPEN(UNIT=11,FILE='output/VERIFYING_TEMP_SERIES.DAT')

!-----------------------------------------------------------------
!Definition of positions occupied by centres of mass of particles

cont=0
DO I = 1,np_side
  X =(I-1)*dx + dx/2

  DO J = 1,np_side
    cont = cont + 1
    Y = (J-1)*dy + dy/2
    POSX(cont)= X
    POSY(cont)= Y

    !Calculation of temperatures

     Txy=0.
     DO K = 1, KMAX, 2
       SINX = SIN(K*PI*X)
       aux = ((SINH(K*PI*(Y-1.))) /(SINH(K*PI)))
       Txy = Txy + ((-4.*TS)/(K*PI))* SINX * aux
     END DO

     TEMP(cont)=Txy

   END DO

 END DO

 DO I = 1,cont
   WRITE(11,23) I, POSX(I), POSY(I), TEMP(I)
 END DO
23 FORMAT (1X,I10, 1X, D21.14, 1X, D21.14, 1X, D21.14)
 close(11)
 OPEN(23,FILE='N_PART_SERIES.DAT')
 WRITE(23,*) cont
 WRITE(23,*) dt
 CLOSE(23)

!Defining a line of boundary particles and temperatures
 n_bound = 0
 J = 1
 DO I = 1, MP1
   n_bound= n_bound+1
   X1(n_bound,1)=(I-1)*dx
   X1(n_bound,2)=0.
   X1(n_bound,3)=TS !Lower boundary
 END DO
```

```
J = NP1
DO I = 1,MP1
  n_bound = n_bound+1
  X1(n_bound,1)=(I-1)*dx
  X1(n_bound,2)=(NP1-1)*dy
  X1(n_bound,3)=TN !Upper boundary
END DO

I=1
DO J=2,np_side
n_bound = n_bound+1
   X1(n_bound,1)=0.
   X1(n_bound,2)=(J-1)*dy
   X1(n_bound,3)=TW !Left boundary
 END DO

 I=MP1
 DO J=2,np_side
  n_bound=n_bound+1
  X1(n_bound,1)=(MP1-1)*dx
  X1(n_bound,2)=(J-1)*dy
  X1(n_bound,3)=TE !Right boundary
END DO

OPEN(12,FILE='output/BOUNDARY_PARTICLES.DAT')
DO I=1,n_bound
  WRITE(12,23) I, (X1(I,d),d=1,3)
END DO
 CLOSE(12)

END SUBROUTINE temperature_series

!************************** INPUT ******************************

SUBROUTINE input(aux_p, x, mass, rho, np_side, ntotal, T_0, LX, LY,
support_radi us, n_cells)

!-----------------------------------------------------------------
! This subroutine makes the distribution of particles at domain
! and the initial properties are set
!
! x - positions of particles
! mass - mass of particles
! rho - densities of particles
! hsml - smoothing lengths of particles
! ntotal - total number of particles
! ncells - total number of cells in each direction
! T_0 - initial temperature

IMPLICIT NONE
INTEGER np_side, aux_p , n_cells, ntotal
DOUBLE PRECISION LX, LY, xl, yl, dx, dy, support_radius
INTEGER i, k, d
DOUBLE PRECISION mass(aux_p), rho(aux_p), T_0(aux_p)
DOUBLE PRECISION x(aux_p,2), M(aux_p,4)
```

```
n_cells=np_side/5
yl= LY
xl = LX
dx = xl/np_side
dy = yl/np_side
OPEN(23,FILE='N_PART_SERIES.DAT')
READ (23,*) k
 CLOSE(23)

OPEN(UNIT=11,FILE='output/VERIFYING_TEMP_SERIES.DAT') !PARTICLES AT DOMAIN
DO i=1,k
  READ(11,*) (M(i,d),d=1,4)
END DO

DO i=1,k
  x(i,1)=M(i,2)
  x(i,2)=M(i,3)
END DO

 CLOSE (11)

!----------------------DEFINING THE SUPPORT RADIUS ----------------
support_radius = (xl/real(n_cells))/2.
!------------------------------------------------------------------

!----------------------DEFINING THE PROPERTIES OF PARTICLES -------

DO i = 1, aux_p
  rho (i) = 1.
  mass(i) = dx*dy*rho(i)
  T_0(i) = 0.
  WRITE (03,654) T_0(i)
END DO

654 format(1x,D21.14)

DO i=1, k
  WRITE (01,1013) (x(i,d), d = 1, 2)
END DO

1013 format(2(1x,D21.14))
ntotal = k

WRITE (*,*)'---------------------------------------------------'
WRITE (*,*)' TOTAL NUMBER OF PARTICLES AT DOMAIN : '
WRITE (*,*) ntotal
WRITE (*,*)'---------------------------------------------------'
WRITE (*,*) ' '
WRITE (*,*) ' PRESS ANY KEY TO CONTINUE... '

pause

END SUBROUTINE input
```

```
!*********************** BOUNDARY PARTICLES*********************

SUBROUTINE boundary_particles(aux_p, itimestep, np_side, TS, TN, TW, TE,
LX, LY, dx, dy, n_bound, ntotal, n_global, mass,x,rho,T_0)
!------------------------------------------------------------------
! This subroutine defines a line of particles at the contour of
! the flat plate and the Dirichlet boundary conditions
!
! itimestep - current time step
! ntotal - total number of particles at domain
! n_bound - number of boundary particles
! n_global - sum of the number of particles at domain and
! boundary particles
! hsml - smoothing length
! mass - masses of partic¸ces
! x - positions of particles
! rho - densities of particles
! T_0 - initial temperature
!------------------------------------------------------------------

IMPLICIT NONE

INTEGER aux_p, n_global, np_side, ntotal, n_bound, itimestep
INTEGER i, j, d, k
DOUBLE PRECISION TS, TN, TW, TE, LX, LY, dx,dy,MAX_X, MAX_Y
DOUBLE PRECISION :: rho(aux_p), mass(aux_p), hsml(aux_p), T_0(aux_p)
DOUBLE PRECISION :: x(aux_p,2)

n_bound = 0

!-----------Defining a line of particles on the Upper side and setting the
physical properties

DO i = 1, 2*np_side -1
  n_bound = n_bound + 1
  x(ntotal + n_bound,1) = i*dx/2
  x(ntotal + n_bound,2) = LY
  rho (ntotal + n_bound) = 1.
  mass(ntotal + n_bound) = rho (ntotal + n_bound) * dx * dy
  T_0(ntotal + n_bound)= TN
END DO

! -----------Defining a line of particles on the Lower side and setting the
physical properties
 DO i = 1, 2*np_side -1
   n_bound = n_bound + 1
   x(ntotal + n_bound,1) = i*dx/2
   x(ntotal + n_bound,2) = 0.
   rho (ntotal + n_bound) = 1.
   mass(ntotal + n_bound) = rho (ntotal + n_bound) * dx * dy
   T_0(ntotal + n_bound)= TS
 END DO

!----------------Defining a line of particles on the Left side and setting
the physical properties
 DO i = 1, 2*np_side + 1
   n_bound = n_bound + 1
   x(ntotal + n_bound,1) = 0.
   x(ntotal + n_bound,2) = (i-1)*dx/2
   rho (ntotal + n_bound) = 1.
   mass(ntotal + n_bound) = rho (ntotal + n_bound) * dx * dy
   T_0(ntotal + n_bound)= TW
 END DO

!----------------Defining a line of particles on the Right side and setting
the physical properties
 DO i = 1, 2*np_side+1
   n_bound = n_bound + 1
   x(ntotal + n_bound,1) = LX
   x(ntotal + n_bound,2) = (i-1)*dx/2
   rho (ntotal + n_bound) = 1.
   mass(ntotal + n_bound) = rho (ntotal + n_bound) * dx * dy
   T_0(ntotal + n_bound)= TE
 END DO

MAX_Y= 0.
MAX_Y =0.
!------------------------ Output Files----------------------------
DO i=ntotal+1, ntotal+ n_bound

  IF (x(i,1).GT.MAX_X) MAX_X = x(i,1)
  IF (x(I,2).GT.MAX_Y) MAX_Y = x(i,2)

  WRITE (01,1016) (x(i,d), d = 1, 2)
  n_global = ntotal + n_bound
  WRITE (02,1016) (x(i-ntotal,d), d = 1, 2)
  WRITE (03,654) T_0(i)

END DO
1016 format(2(1x,D21.14))
654 format(1x,D21.14)

OPEN(17,FILE='output/GEOMETRY.DAT')
WRITE(17,*) 0., MAX_X
WRITE(17,*) 0., MAX_Y
 CLOSE(17)

END SUBROUTINE boundary_particles

!********************* NEIGHBOUR SEARCH ************************

SUBROUTINE neighbour_search(aux_p, x, n_global, ntotal, n_bound, neighbour,
dx, dy, support_radius, dist)

!-----------------------------------------------------------------
! This subroutine finds the direct search for neighbour particles ! of
each fixed particle
!
! ntotal - number of particles at domain
! n_bound - number of boundary particles
! n_global - sum of the number of particles at domain and
! boundary particles
! hsml - smoothing length
! x - positions of all particles
! n_neigh - number of neighbour particles
! dist - distance between a fixed particle and a neighbour particle
! neighbour - matrix of neighbour particles
!-----------------------------------------------------------------

IMPLICIT NONE

INTEGER aux_p, n_global, ntotal, n_bound, n_neigh, i, j
INTEGER neighbour(aux_p,150)
DOUBLE PRECISION r , dx_local, dy_local, support_radius
DOUBLE PRECISION x(aux_p,2), dist(aux_p,150), dx(aux_p,150), dy(aux_p,150)

DO i=1,aux_p
 DO j=1,150
   neighbour(i,j)=0
   dist(i,j) = 0.
   dx(i,j)=0.
   dy(i,j)=0.
  END DO
 END DO

 DO i=1,ntotal
    n_neigh = 1
    neighbour(i,n_neigh)=i

   DO j = 1, n_global
     dx_local = x(i,1) - x(j,1)
     dy_local = x(i,2) - x(j,2)
     r = sqrt(dx_local**2 + dy_local**2)

     IF (r.LE.support_radius) THEN

       n_neigh = n_neigh +1
       neighbour(i,n_neigh)=j
       dx(i,n_neigh) = dx_local
       dy(i,n_neigh) = dy_local
       dist(i,n_neigh) = r
     END IF

   END DO
 END DO

 OPEN(222,file='output/NEIGHBOURING_PARTICLES.dat')
 REWIND(222)

 DO i = 1, ntotal
  n_neigh = 1
   DO WHILE (n_neigh.ne.150)
     WRITE (222,150), neighbour(i,n_neigh)
     n_neigh = n_neigh +1
   END DO
 END DO

 150 format(I10)
  CLOSE(222)

 END SUBROUTINE neighbour_search
```

```
!***************** LAPLACIAN OF TEMPERATURE ********************

SUBROUTINE SPH_Temperature_Laplacian(aux_p, int_function, factor, hsml,
ntotal, dist, mass, rho, support_radius, neighbour, T_0, temp_Laplacian,
itimestep)

!-----------------------------------------------------------------
! This subroutine makes calculations of the divergent of the heat
! flux !using a kernel chosen by user
!
! dist - Distance between a fixed particle and a neighbouring
! particle
! ntotal - number of particles at domain
! k_scale - scaling factor (depends on kernel)
! factor - normalization constant of kernel
! temp_Laplacian - Laplacian of temperature provided by SPH
! method
! int_function - kernel interpolation
!
! Kernel options:
! 1. Lucy's quartic kernel
! 2. Cubic spline kernel
! 3. New quartic kernel
! 4. Quintic spline kernel
!-----------------------------------------------------------------

IMPLICIT NONE

INTEGER aux_p, i, j, k, d, ntotal, n_global, n_neigh, int_function,
itimestep
INTEGER neighbour(aux_p,150)
DOUBLE PRECISION temp_Laplacian (aux_p), aux, q, factor, r,
support_radius, k_scale
DOUBLE PRECISION rho(aux_p), mass(aux_p), hsml(aux_p), T_0(aux_p)
DOUBLE PRECISION dist(aux_p,150), dx(aux_p,150), dy(aux_p,150), PI

PI=ACOS(-1.)

IF (itimestep.EQ.1) THEN

  DO i=1,aux_p
   hsml(i) = 0.
  END DO

  DO i=1,ntotal

   IF (int_function.EQ.1) THEN !----------LUCY'S QUARTIC KERNEL
      k_scale = 1.0
      hsml(i) = support_radius/k_scale
      factor = 5.0/(PI*hsml(i)*hsml(i))
   ELSE
      IF (int_function.EQ.2) THEN !--------- CUBIC SPLINE KERNEL
        k_scale = 2.0
        hsml(i) = support_radius/k_scale
        factor = 15./(7.*PI*hsml(i)*hsml(i))
      ELSE
        IF (int_function.EQ.3) THEN !--------NEW QUARTIC KERNEL
```

```
          k_scale = 1.0
          hsml(i) = support_radius/k_scale
          factor = 15./ (7.*PI*hsml(i)*hsml(i))
        ELSE
          IF (int_function.EQ.4) THEN !-----QUINTIC SPLINE KERNEL
            k_scale = 3.0
            hsml(i) = support_radius/k_scale
            factor = 7./(478.*PI*hsml(i)*hsml(i))
          END IF
        END IF
      END IF
    END IF

  END DO

END IF

!---------- SPH TEMPERATURE LAPLACIAN CALCULUS(in the polar coordinates
system)

DO i=1,ntotal
  n_neigh =2
  temp_Laplacian(i)=0.

  DO WHILE (neighbour(i,n_neigh).ne.0)
    j= neighbour(i,n_neigh)
    r = dist(i,n_neigh)
    q = r/hsml(i)

!-----------------------------------------------------------------
!--------------------LUCY'S QUARTIC KERNEL------------------------
!-----------------------------------------------------------------

    IF (int_function.EQ.1) THEN

      aux = 2*((T_0(i)-T_0(j))) * factor * ( (-12./hsml(i)**2.) +
(24.*r)/hsml(i)**3. - (12*r**2.)/hsml(i)**4. ) * (mass(j)/rho(j))
      temp_Laplacian(i)= temp_Laplacian(i) + aux
    END IF

!-----------------------------------------------------------------
!---------------------CUBIC SPLINE KERNEL-------------------------
!-----------------------------------------------------------------

    IF (int_function.eq.2) THEN

      IF ((q.ge.0.).and.(q.le.1.)) THEN

        aux = 2.*(( T_0(i)- T_0(j))) * factor * ( (-2.)/(hsml(i)**2.) +
(3.*r)/(2.*hsml(i)**3.) ) * (mass(j)/rho(j))
        temp_Laplacian(i)= temp_Laplacian(i) + aux

      ELSE
        IF ((q.gt.1.).and.(q.le.2.)) THEN
          aux = 2.*( (T_0(i)- T_0(j)) ) * factor * (-1./(2.*hsml(i))) *
((2. - (r/hsml(i) ) )**2 ) * (1./r) * (mass(j)/rho(j))
          temp_Laplacian(i)= temp_Laplacian(i) + aux
        END IF
      END IF
    END IF

!-----------------------------------------------------------------
!---------------------NEW QUARTIC KERNEL--------------------------
!-----------------------------------------------------------------
    IF (int_function.EQ.3) THEN

      aux = 2*((T_0(i)-T_0(j))) * factor * ( -18./(8.*hsml(i)**2.) +
(57.*r)/(24.*hsml(i)**3) - (20.*r*r)/(32.*hsml(i)**4.) ) * (mass(j)/rho(j))
      temp_Laplacian(i)= temp_Laplacian(i)+ aux

      END IF

!-----------------------------------------------------------------
!---------------------QUINTIC SPLINE KERNEL-----------------------
!-----------------------------------------------------------------

    IF (int_function.EQ.4) THEN

      IF ((q.ge.0.).and.(q.le.1.)) THEN
        aux = 2.*( (T_0(i)-T_0(j)) ) * factor * ( -5./hsml(i))* ( 10.*
(r**3)/hsml(i)**4 - 24.*(r**2)/hsml(i)**3 + 24./hsml(i) ) *
(mass(j)/rho(j))
        temp_Laplacian(i)= temp_Laplacian(i) + aux

      ELSE
        IF ((q.gt.1.).and.(q.le.2.)) THEN
          aux = 2.*( (T_0 (i)- T_0 (j)) ) * factor * ( -5./hsml(i) )* ( -
5.* (r**3/hsml(i)**4) + 36.*(r**2/hsml(i)**3) - 90.*(r/hsml(i)**2) +
84./hsml(i) - 15./r )* (mass(j)/rho(j))
          temp_Laplacian(i)= temp_Laplacian(i) + aux

        ELSE
          IF ((q.gt.2.).and.(q.le.3.)) then
            aux = 2.*( (T_0 (i)- T_0 (j)) ) * factor * ( -5./hsml(i) )* (
(r**3)/hsml(i)**4 -(12.*r**2)/hsml(i)**3 + (54.*r)/hsml(i)**2 -
108./hsml(i) + 81./r ) * (mass(j)/rho(j))
            temp_Laplacian(i)= temp_Laplacian(i) + aux
          END IF
        END IF
      END IF
    END IF

    n_neigh =n_neigh +1

  END DO

END DO

end SUBROUTINE SPH_Temperature_Laplacian

!****************** TEMPORAL INTEGRATION ***********************

SUBROUTINE temporal_integration(aux_p, itimestep, x, ntotal, dt,
temp_Laplacian, T_0, T, step_out)
```

```
!-----------------------------------------------------------------
! This subroutine performs the integration of the temperature of ! each
particle at the domain
!
! ntotal - total number of particles at domain
! dt - timestep
! T_0 - temperature at current iteration
! T - temperature in the next iteration
!-----------------------------------------------------------------

IMPLICIT NONE

INTEGER aux_p, ntotal, itimestep, d, step_out, i, j, aux, funit
DOUBLE PRECISION temp_Laplacian(aux_p), T_0(aux_p), T(aux_p), x(aux_p,2),
dt
CHARACTER(LEN=20) aux2
CHARACTER(LEN=35) nome
OPEN(unit=66, file='output/VERIFYING_TEMP_SPH.dat')
REWIND(66)

!--------CALCULATION OF TEMPERATURE OF EACH PARTICLE AT DOMAIN---

 DO i=1,ntotal
  T(i)= T_0(i) + temp_Laplacian(i)*dt
 END DO

!------------------------- OUTPUT FILES ------------------------
aux=itimestep

IF ((aux.EQ.1).OR.(mod(aux,step_out).eq.0)) THEN
  WRITE (aux2,*) aux
  funit=aux+150

  IF (aux.LT.10)
nome='output/temperature/000000'//trim(adjustl(aux2))//'.dat'
  IF ((aux.GE.10).AND.(aux.LT.100))
nome='output/temperature/00000'//trim(adjustl(aux2))//'.dat'
  IF ((aux.GE.100).AND.(aux.LT.1000))
nome='output/temperature/0000'//trim(adjustl(aux2))//'.dat'
  IF ((aux.GE.1000).AND.(aux.LT.10000))
nome='output/temperature/000'//trim(adjustl(aux2))//'.dat'
  IF ((aux.GE.10000).AND.(aux.LE.100000))
nome='output/temperature/00'//trim(adjustl(aux2))//'.dat'
  IF ((aux.GE.100000).AND.(aux.LT.1000000))
nome='output/temperature/0'//trim(adjustl(aux2))//'.dat'
  IF (aux.EQ.1000000)
nome='output/temperature'//trim(adjustl(aux2))//'.dat'

   OPEN(funit,file=nome,status='unknown')

   DO i = 1, ntotal
     WRITE (funit,89) i, x(i,1), x(i,2), T(i)
   END DO

   CLOSE (funit)

END IF
```

```
DO i=1,ntotal
  WRITE (66,89) i, (x(i,d), d = 1, 2), T(i)
END DO

89 format(1x,I10, 1x,D21.14, 1x,D21.14, 1x,D21.14)
  CLOSE(66)

OPEN(UNIT=26,FILE='STEP_OUT.DAT')
WRITE (26,*) step_out
 CLOSE (26)

END SUBROUTINE temporal_integration

!**********************CONVERGENCE ****************************

SUBROUTINE convergence(aux_p, itimestep,T_0,T,eps,response, ntotal)

!-----------------------------------------------------------------
! This subroutine verifies the convergence of the SPH solution
! Convergence criteria: abs(T_0(i) - T(i)) $<$= accuracy
! where i each particle at domain
!
! T_0 - temperature at current iteration
! T - temperature in the next iteration
! ntotal - total number of particles at domain
! dt - timestep
!-----------------------------------------------------------------

IMPLICIT NONE
INTEGER aux_p, itimestep, I, ntotal
DOUBLE PRECISION T_0(aux_p), T(aux_p), error(ntotal), eps, max_error
CHARACTER(LEN=15) response

DO I=1,ntotal
  error(I) = abs(T(I) - T_0(I))

END DO

max_error = maxval(error)

print*, 'MAXIMUM ERROR = ', max_error, ' AT ', itimestep, 'a. ITERATION'

  IF (max_error.GT.eps) THEN
    response ='non_convergence'
    itimestep = itimestep+1
  ELSE
    response = 'convergence'
  END IF

!------------- UPDATING TEMPERATURES TO THE NEXT ITERATION -------
DO I=1,ntotal
  T_0(I) = T(I)
END DO

END SUBROUTINE convergence
```

```
!*********** POINT-TO-POINT TEMPERATURE DIFFERENCE *************

SUBROUTINE temperature_differences(itimestep, dt, ntotal, step_out)

!-----------------------------------------------------------------
! This subroutine calculates the differences between the
! temperatures obtained by the SPH method and provided by series
!
!-----------------------------------------------------------------

INTEGER i, d, ntotal, itimestep, part_higher, part_smaller, step_out
DOUBLE PRECISION series_solution(ntotal,4), SPH_solution(ntotal,4),
Temp_differences(ntotal), smaller_difference, higher_difference, dt

OPEN(unit=11,file='output/VERIFYING_TEMP_SERIES.DAT')
OPEN(unit=66,file='output/VERIFYING_TEMP_SPH.dat')
OPEN(unit=73,file='output/TEMPERATURE_DIFFERENCES.DAT')
OPEN(unit=74,file='output/FINAL_DIFERENCES.DAT')
OPEN(unit=75,file='output/STEP_DIFFERENCES.DAT')
REWIND(73)
REWIND(74)

DO i=1,ntotal
  read(11,*) (series_solution(i,d), d=1,4)
  read(66,*) (SPH_solution(i,d), d=1,4)
END DO

DO i=1,ntotal
 Temp_differences(i)= abs(series_solution(i,4)- SPH_solution(i,4))
 WRITE (73,55) (series_solution(i,d), d=2,3), Temp_differences(i)
 WRITE (74,55) (series_solution(i,d), d=2,3), Temp_differences(i)

 IF (mod(itimestep,step_out).eq.0) THEN
   WRITE (75,55) (series_solution(i,d), d=2,3), Temp_differences(i)
 END IF

END DO
55 format (1x,D21.14, 1x,D21.14, 1x, D21.14)

higher_difference = 0.
smaller_difference = 1500.

DO i=1,ntotal
  IF (Temp_differences(i).GT.higher_difference) then
    part_higher =i
    higher_difference = Temp_differences(i)
  END IF

  IF (Temp_differences(i).LT.smaller_difference) then
    part_smaller = i
    smaller_difference = Temp_differences(i)
  END IF
END DO

WRITE (73,*) ''
WRITE (73,*) 'RESULTS AT THE ', itimestep , 'a. ITERATION'
WRITE (73,*) 'TIME = ' , itimestep*dt, ' seconds'
WRITE (73,*) ''
WRITE (73,*) '----------------------------------------'
WRITE (73,*) 'HIGHER TEMPERATURE DIFFERENCE'
WRITE (73,*) '----------------------------------------'
WRITE (73,*) ' Position = ', (series_solution(part_higher,d),d=2,3)
WRITE (73,*) ' Series Temperature --- SPH Temperature --- Difference'
WRITE (73,*) series_solution(part_higher,4), SPH_solution(part_higher,4),
higher_difference
WRITE (73,*) '-----------------------------------'
WRITE (73,*) '----------------------------------------'
WRITE (73,*) 'SMALLER TEMPERATURE DIFFERENCE'
WRITE (73,*) '----------------------------------------'
WRITE (73,*) ' Position = ', (series_solution(part_smaller,d),d=2,3)
WRITE (73,*) ' Series Temperature --- SPH Temperature --- Difference'
WRITE (73,*) series_solution(part_smaller,4),
SPH_solution(part_smaller,4), smaller_difference
WRITE (73,*) '---------------------------------------'

 CLOSE(11)
 CLOSE(66)
 CLOSE(73)
 CLOSE(74)
 CLOSE(75)

END SUBROUTINE temperature_differences
```

MATLAB SCRIPT TO PLOT GRAPHS OF TEMPERATURE

This is the MATLAB script to plot graphs of temperatures distribution

from the transient to steady-state regime.

Following the instructions to obtain the output data from the

FORTRAN program and plot figures.

```
%-----------------------------------------------------------------
% Instructions:
% 1.Within the working directory containing the FORTRAN program,
% create and open a new file in the MATLAB Editor˜
% 2.Copy and paste this script and save as ‘‘name.m’’
% 3.Create a subfolder called figures within the output folder
% 4.Update the path name of the working directory in this MATLAB script
% 5.Run
% 6.Figures will be save within the figures subfolder
%-----------------------------------------------------------------

%-----------------------------------------------------------------
%----------------- PLOTTING GRAPH (SERIES SOLUTION)---------------
%-----------------------------------------------------------------
clc
clear all
close all

a= load('working_directory/output/GEOMETRY.DAT');
Xm = a(1,1);
Xmx = a(1,2);
```

```
Ym = a(2,1);
Ymx = a(2,2);

b= load('working_directory/N_PART_SERIES.DAT');
part_side = sqrt(b(1,1));
dt = b(2,1);

c= load('working_directory/NUMBER_OF_ITERATIONS.DAT');
iterations = c(1,1);

d= load('working_directory/STEP_OUT.DAT');
step_out = d(1,1);

f=load ('working_directory/output/VERIFYING_TEMP_SERIES.DAT');
fr=sortrows(f);
x_plot=fr(:,2);
y_plot=fr(:,3);
T1=fr(:,4);

x_series=reshape(x_plot,part_side,part_side);
y_series=reshape(y_plot,part_side,part_side);
T_series=reshape(T1,part_side,part_side);

figure(1);
contourf(x_series, y_series, T_series, 20);
grid on
daspect ([ 1 1 1 ]);
colormap jet;
cH = colorbar;
set(gcf, 'Position', get (0,'Screensize'));
set(cH,'FontSize',12);
set(get(cH,'title'),'string','T(^{o}C)','FontSize',12);
caxis([0 100]);
axis ([Xm Xmx Ym Ymx]);
set(gca,'xticklabel',num2str(get(gca,'ytick')','%.2f'),'fontsize',10);
set(gca,'yticklabel',num2str(get(gca,'ytick')','%.2f'),'fontsize',10);

title('SOLUTION BY SERIES','fontsize',14);
xlabel('x (m)','fontsize',12);
ylabel('y (m)','fontsize',12);
disp('Figures will be saved in working_directory/output/figures')
disp('Press any key to continue...');
pause

dir_out = (['cd working_directory/output/figures']);
eval(dir_out);

img = getframe(gcf);
imwrite(img.cdata, ['SERIES_SOLUTION.png']);
close;

%-----------------------------------------------------------------
%-----------------PLOTTING GRAPHS (SPH SOLUTIONS)------------------
%-----------------------------------------------------------------
dir_in = (['cd working_directory/output/temperature']);
%----------------------INITIAL DISPOSITION------------------------
```

```
FileName=(['0000001']);
t = load ('working_directory/output/temperature/0000001.dat');

fr=sortrows(t);
x_plot=fr(:,2);
y_plot=fr(:,3);
T1=fr(:,4);

x_SPH =reshape(x_plot,part_side,part_side);
y_SPH =reshape(y_plot,part_side,part_side);
T_SPH=reshape(T1,part_side,part_side);

eval(dir_out);
figure(2);
contourf(x_SPH, y_SPH, T_SPH, 20);

grid on
daspect ([ 1 1 1 ]);
colormap jet;
cH = colorbar;
set(gcf, 'Position', get (0,'Screensize'));
set(cH,'FontSize',12);
set(get(cH,'title'),'string','T(^{o}C)','FontSize',12);
caxis([0 100]);
axis ([Xm Xmx Ym Ymx]);
set(gca,'xticklabel',num2str(get(gca,'ytick')','%.2f'),'fontsize',10);
set(gca,'yticklabel',num2str(get(gca,'ytick')','%.2f'),'fontsize',10);

title1 = (['SPH SOLUTION - Cubic Spline Kernel - t = 0.00 s']);
title(title1,'Units','Normalized','fontsize',12)
title(title1,'fontsize',12,'fontweight','b')
xlabel('x (m)','fontsize',12);
ylabel('y (m)','fontsize',12);

img = getframe(gcf);
imwrite(img.cdata, [FileName, '.png']);
clear FileName;

for i = step_out:step_out:iterations,
    num=int2str(i);
    char1=num2str(i);
    char2=num2str(iterations);
    eval(dir_in);

      if (i $>$= 0) & (i $<$ 10)
           FileName=([ '000000' num]);
        else
          if (i $>$= 10) & (i $<$ 100)
            FileName=([ '00000' num]);
          else
            if (i $>$= 100) & (i $<$ 1000)
              FileName=([ '0000' num]);
            else
              if (i $>$= 1000) & (i $<$ 10000)
                FileName=([ '000' num]);
             else
               if (i $>$= 10000) & (i $<$ 100000)
        FileName=([ ’00’ num]);
        else
         if (i $>$= 100000) & (i $<$ 1000000)
         FileName=([’0’ num]);
         else
            FileName=([ num]);
          end
         end
       end
      end
     end
   end

    disp([’opening’ FileName ’.dat’ ]);
    t = load ([FileName ’.dat’]);

    fr=sortrows(t);
    x_plot=fr(:,2);
    y_plot=fr(:,3);
    T1=fr(:,4);

    clear FileName;

    x_SPH =reshape(x_plot,part_side,part_side);
    y_SPH =reshape(y_plot,part_side,part_side);
    T_SPH=reshape(T1,part_side,part_side);

    figure(i);
    contourf(x_SPH, y_SPH, T_SPH, 20);

    grid on
    daspect ([ 1 1 1 ]);
    colormap jet;
    cH = colorbar;
    set(gcf, ’Position’, get (0,’Screensize’));
    set(cH,’FontSize’,12);
    set(get(cH,’title’),’string’,’T(^{o}C)’,’FontSize’,12);
    caxis([0 100]);
    axis ([Xm Xmx Ym Ymx]);
set(gca,’xticklabel’,num2str(get(gca,’ytick’)’,’%.2f’),’fontsize’,10);
set(gca,’yticklabel’,num2str(get(gca,’ytick’)’,’%.2f’),’fontsize’,10);

    aux2=i*dt;
    num1=num2str(aux2);
    title1 = ([’SPH SOLUTION - t = ’ num1 ’ s’]);
    title(title1,’Units’,’Normalized’,’fontsize’,12)
    title(title1,’fontsize’,12,’fontweight’,’b’)
    xlabel(’x (m)’,’fontsize’,12);
    ylabel(’y (m)’,’fontsize’,12);

    eval(dir_out);
    img = getframe(gcf);
    if i $<$ 10
     nome=([’000000’ num ’.png’]);
     saveas(gcf,nome);
    end
    if (i $>$= 10) & (i $<$ 100)
```

```
          nome=(['00000' num '.png' ]);
          imwrite(img.cdata, nome);
        end
        if (i $>$= 100) & (i $<$ 1000)
          nome=(['0000' num '.png' ]);
          imwrite(img.cdata, nome);
        end
        if (i $>$= 1000) & (i $<$ 10000)
          nome=(['000' num '.png' ]);
          imwrite(img.cdata, nome);
        end
        if (i $>$= 10000) & (i $<$ 100000)
          nome=(['00' num '.png' ]);
          imwrite(img.cdata, nome);
        end
        if (i $>$= 100000) & (i $<$ 1000000)
          nome=(['0' num '.png' ]);
          imwrite(img.cdata, nome);
        end
        if (i $>$= 1000000)
          nome=([ num '.png']);
          imwrite(img.cdata, nome);
        end
   close all
end
```

References

1. Fraga Filho, C.A.D.: Development of a computational instrument using a Lagrangian particle method for physics teaching in the areas of fluid dynamics and transport phenomena. Rev. Bras. Ensino Fís. (online) **39**(4), e4401 (2017). https://doi.org/10.1590/1806-9126-rbef-2016-0289
2. Coveney, P.V., Boon, J.P., Succi, S.: Bridging the gaps at the physics-chemistry-biology interface. Philos. Trans. R. Soc. A (2016). https://doi.org/10.1098/rsta.2016.0335
3. Stalter, S., Yelash, L., Emamy, N., Statt, A., Hanke, M., Lukácové-Medvid'ová, M., Virnau, P.: Molecular dynamics simulations in hybrid particle-continuum schemes: pitfalls and caveats. Comput. Phys. Commun. **224**, 198–208 (2018)
4. Liu, W.K., Park, H.S., Qian, D., Karpov, E.G., Hiroshi, K., Wagner, G.J.: Bridging scale methods for nanomechanics and materials. Comput. Methods Appl. Mech. Eng. **195**, 1407–1421 (2006)
5. Petsev, N.D., Leal, L.G., Shell, M.S.: Multiscale simulation of ideal mixtures using smoothed dissipative particle dynamics. J. Chem. Phys. **144**, 084115 (2016). https://doi.org/10.1063/1.4942499
6. Liu, M.B., Liu, G.R., Zhou, L.W., Chang, J.Z.: Dissipative particle dynamics (DPD): an overview and recent developments. Arch. Comput. Methods Eng. **22**, 529–556 (2015)
7. Hemminger, J.: From quanta to the continuum: opportunities for mesoscale science. A Report from the Basic Energy Sciences Advisory Subcommittee, U.S. Department of Energy, USA (2012). https://doi.org/10.2172/1183982
8. Delgado-Buscalioni, R., Coveney, P.V., Riley, G.D., Ford, R.W.: Hybrid molecular-continuum fluid models: implementation within a general coupling framework. Philos. Trans. R. Soc. A **363**(1833), 1975–85 (2005)
9. Borgh, M.K., Lockerby, D.A., Reese, J.M.: Fluid simulations with atomistic resolution: a hybrid multiscale method with field-wise coupling. J. Comput. Phys. **255**, 149–165 (2013)
10. Sih, G.C. (ed.): Multiscaling in Molecular and Continuum Mechanics: Interaction of Time and Size from Macro to Nano: Application to Biology, Physics, Material Science, Mechanics, Structural and Processing Engineering. Springer, The Netherlands (2007)
11. Mukhopadhyay, S., Abraham, J.: A particle-based multiscale model for submicron fluid flows. Phys. Fluids **21**, 027102 (2009). https://doi.org/10.1063/1.3073041
12. Yamaguchi, T., Ishikawa, T., Imai, Y., Matsuki, N., Xenos, M., Deng, Y., Bluestein, D.: Particle-based methods for multiscale modeling of blood flow in the circulation and in devices: challenges and future directionsD. Ann. Biomed. Eng. **38**(3), 1225–35 (2010)
13. Ren, W., Weinan, E.: Heterogeneous multiscale method for the modeling of complex fluids and micro-fluidics. J. Comput. Phys. **204**, 1–26 (2005)

14. Chen, S., Wang, W., Xia, Z.: Multiscale fluid mechanics and modeling. In: Bai, Y., Wang, J., Fang, D. (eds.) Mechanics for the World: Proceedings of the 23rd International Congress of Theoretical and Applied Mechanics, ICTAM2012, Beijing, China, Procedia IUTAM, vol. 10, pp. 110–114 (2014)
15. Moeendarbary, E., Ng, T.Y., Zangeneh, M.: Dissipative particle dynamics: introduction, methodology and complex fluid applications—a review. Int. J. Appl. Mech. **1**(4), 737–763 (2009)

Index

C. A. Dutra Fraga Filho, *Smoothed Particle Hydrodynamics*,
https://doi.org/10.1007/978-3-030-00773-7