The Fokker-Planck-Kolmogorov Equation Solved Through Smoothed Particle Hydrodynamics: A Computational and Applied Study

Thesis presented to obtain the degree of
Master of Science in Physics Engineering

Université de Liège   Faculté des Sciences Appliquées

Tom BERTRAND

Supervisors:

Romain BOMAN
Vincent DENOËL

Jury:

Pierre ARCHAMBEAU
Maarten ARNST

— Academic year: 2024–2025 —

Remerciements

Ah qu'il est bon de poser sa plume, aussi numérique soit-elle, et d'abandonner, juste pour un instant, les règles de l'écriture scientifique pour retrouver sa langue maternelle. On se retrouve alors face à cette petite centaine de pages qui représente bien plus que la simple matérialisation d'un dur labeur…

C'est l'esprit léger et le regard fier que j'aimerais exprimer ma gratitude envers tous ceux et celles qui ont participé à cette grande pyramide humaine qui me permet aujourd'hui de vous présenter ce mémoire de fin d'études.

C'est à vous, chers membres du jury, que je souhaite adresser mes premiers remerciements. J'espère que ce travail vous plaira autant que son sujet me passionne et que vous comprendrez l'émerveillement, sans doute un peu candide, qu'il m'inspire.

En particulier, je tiens à exprimer ma gratitude envers mes promoteurs, Vincent DENOËL et Romain BOMAN, pour m'avoir éclairé de leur expertise durant cette longue traversée et m'avoir aidé à y forger ma propre lanterne. Soyez certains que vos conseils restent compilés dans un coin de ma tête et qu'ils m'accompagneront tout au long du chemin que vous avez ouvert devant moi. Ce chemin, je le parcourrai encore quelque temps sous votre égide, cher Vincent, alors je vous dis à l'année prochaine pour de nouvelles aventures !

On se souvient tous de l'un ou l'autre de ses enseignants d'école secondaire. S'il est un nom qui mérite d'être marqué dans mon mémoire tant il a marqué ma mémoire, c'est bien celui d'Arnaud STIEPEN. Lui qui a délaissé les méthodes de l'enseignement traditionnel pour donner du sens et de la passion à l'apprentissage de la physique a bien réussi son pari. Arnaud, tu as tout mon respect et mon admiration.

Si l'université m'a offert un enseignement de qualité, elle m'a également mené à celui qui deviendrait mon frère d'armes et mon ami, un brave gars nommé Florent. Elles nous en ont bien fait baver ces années, hein ? Et pourtant, elle ne me semble pas si folle l'idée de tout recommencer et de retrouver la douceur de nos trajets matinaux. En souvenir de ta pensée critique, toujours offerte avec modestie, du cursus que nous avons partagé, et des nombreuses anecdotes que je ne me risquerai pas à raconter ici, je te remercie sincèrement et te souhaite le meilleur pour la suite.

Avant même d'entreprendre un parcours académique, il m'a été attribué deux promoteurs d'un genre particulier. Avec eux, pas d'équations, de matrices ou d'autres barbaries : simplement l'ambition de répondre à des montagnes de « quoi ça ? » et de « et pourquoi ? » d'un enfant qui apprend le monde. Pour avoir pris soin de cet enfant aux questions incessantes, et pour lui avoir permis de réaliser son rêve de devenir « inventeur », Papa, Maman, je vous adresse un merci à la hauteur du temps passé à vos côtés, là où il fait bon de vivre.

Il serait de mauvais compte de ne pas souligner l'effort tout particulier d'un père toujours présent pour insuffler l'admiration des matières scientifiques et le sens du travail rigoureux.

All work and no play makes Jack a dull boy,
All play and no work makes Jack a mere toy.

Pour cela, en plus du reste, je te dédie cet ouvrage dont tu connais la valeur symbolique.

Quelques chips, des olives et des grissinis… et bien sûr les amis qui vont avec ! Dans le rythme effréné de mes rédactions solitaires, ils ont été nombreux à m'apporter le soutien et le recul nécessaires à la bonne réalisation de ce projet, que ce soit par un subtil sourire ou un plus direct « eh on s'en fout de ça, tu bois quoi ? ». Pour toutes ces pauses étrangement productives, deux mots : merci et encore.

Lilou. Que te dire que tu ne saches déjà ? si ce n'est la fierté que j'éprouve en nous regardant, jeunes adultes à la porte d'un nouveau chapitre de notre vie à deux. Même si mon charabia mathématique ne risque pas de te parler davantage avec les années, je sais que tu continueras d'être la bienveillance de l'oreille qui écoute, de la voix qui apaise et du bras qui soutient. Et puisque toi aussi tu t'es essayée à la rédaction toute récente de ton mémoire, je tiens à te féliciter et te le dire encore une bonne fois : je suis très fier de toi.

C'est non sans une certaine émotion que je clique une ultime fois sur l'icône de sauvegarde, clôturant ainsi une année de travail, avec ses doutes, ses découvertes et ses petits Eurêka. Il ne me reste plus qu'à vous abandonner au Tom du passé, qui a pas mal de choses à vous dire…

Contents

1 Introduction

Random processes are present in countless scientific disciplines. The evolution of stock markets, the fluctuation of particle positions in molecular dynamics, and the random vibrations of mechanical structures are just a few examples where stochasticity plays a central role. Understanding and predicting the behavior of such processes presents significant computational and theoretical challenges.

To study these processes, traditional approaches typically rely on the Monte-Carlo Simulation (MCS) method. This method is versatile and straightforward to implement, but often requires a large number of iterates to achieve statistical accuracy, making it computationally intensive for complex systems with high-dimensional state spaces. Additionally, this method only provides limited insight into the underlying probability structure of the solution, as it only samples discrete trajectories rather than directly computing the Probability Density Function (PDF).

In response, this master's thesis presents an alternative approach that aims to determine the evolution of the PDF itself, without sampling. To do so, stochastic processes are expressed by an equivalent Fokker-Planck-Kolmogorov (FPK) equation which is then solved by applying the Smoothed Particle Hydrodynamics (SPH) method [Luc77, ML85]. This innovative approach introduced by [Can14] remains underexplored to this date. The present study proposes to revisit and expand upon this methodology, leveraging current advancements in computational techniques. Moreover, this work adopts a pedagogical approach in its presentation of theoretical concepts with the aim of providing a foundation for future research in the field. To achieve this goal, the present work is divided in several sections, structured as follows.

First, in Section 2, stochastic processes are presented. More precisely, the transition from a Stochastic Differential Equation (SDE) to its equivalent FPK equation is explained. Some questions that we aim to answer in this section are "*What is a deterministic process?*", "*What is a stochastic process?*", "*How is it expressed as an SDE?*", and finally "*How is it expressed as an FPK equation? What does that imply?*".

Second, the SPH approach is presented in Section 3. The mathematical and practical matters of the approach are introduced for traditional SPH fluid dynamics applications [BBD24, Gof13]. The method is then generalized to more abstract problems such as solving the FPK equation in an arbitrary state space. Answers are sought for the questions "*What is the SPH method?*", "*Why use the SPH method?*" and "*Which mechanisms are introduced to solve the problem at hand?*".

Next, in Section 4, the SPH-FPK method is applied to several benchmark test cases. These test cases are chosen to exhibit some interesting mathematical properties in order to assess the solver's robustness. For this purpose, the results are confronted to MCS, and analytical solutions when applicable. As part of a specific test case, the Moment Equation Method (MEM) and the Cumulant Equation Method (CEM) are briefly introduced for comparison purposes. The main questions are "*How well does the FPK-SPH method perform?*" and "*How does it compare to other methods?*".

Then, some practical implementation matters are discussed in Section 5, especially regarding aspects of the solver that one could find useful while *running* the open-source solver[1]. In this section, the algorithmic architecture of the solver is detailed, as well as how the input and

[1] Available at https://gitlab.uliege.be/structural-and-stochastic-dynamics/fpk-sph-solver.

output files are structured. Multi-core acceleration is also discussed in this section. Overall, the questions which are raised are "*How is the solver's code structured?*", "*How to use the solver in practice?*" and "*Is the solver numerically efficient, and thanks to what?*".

Finally, the master's thesis closes upon a general overview of the work that has been achieved, as well as some interesting perspectives and open questions for future work.

The bibliography and three appendices are present at the very end of this document. For the sake of readability, please note that the first appendix defines a set of mathematical conventions and acronyms that are used throughout this work.

To cite this work: please refer to the guidelines available in this project's GitLab repository(1).

2 The Fokker-Planck-Kolmogorov equation

2.1 Introduction

Simply put, the present work could be summarized as "Solving a specific *equation* with a specific *method*". In this part, the *equation*, namely the FPK equation, is introduced step by step.

The discussion begins with deterministic processes, before extending to stochastic processes. As it is explained in the following, deterministic processes are described by Ordinary Differential Equations (ODEs) whereas the stochastic processes are governed by SDEs.

Instead of analyzing individual realizations of a stochastic process, the focus is placed on expressing the underlying PDF directly. To this end, two methods are introduced: the MCS method and the FPK method.

For the latter, it is shown that a Partial Differential Equation (PDE), namely the FPK equation, can be derived from the SDE. This new equation is deterministic and describes the evolution of the PDF associated with the stochastic process of interest. In the next chapters of this work, a method for solving the FPK equation is studied and leads to practical applications.

2.2 Determinism and ODEs

2.2.1 Definition

Let $\underline{X}(t) : \mathbb{R} \to \mathbb{R}^n$ be a process describing the evolution of some abstract state defined in $\mathbb{R}^n$. Stating that $\underline{X}(t)$ is deterministic comes to say that the entire evolution of $\underline{X}(t)$ can be deduced from the knowledge of

- the laws which dictate the evolution of $\underline{X}(t)$,
- and a state $\underline{X}_0$ explored by $\underline{X}(t)$ at some time t_0.

This definition of determinism has interesting properties [BAB02] and is referred to as Laplacian determinism [LaP14].

> We ought to regard the present state of the universe as the effect of its antecedent state and as the cause of the state that is to follow. An intelligence knowing all the forces acting in nature at a given instant, as well as the momentary positions of all things in the universe, would be able to comprehend in one single formula the motions of the largest bodies as well as the lightest atoms in the world, provided that its intellect were sufficiently powerful to subject all data to analysis; to it nothing would be uncertain, the future as well as the past would be present to its eyes.
>
> — Pierre-Simon de Laplace

Yet another definition of determinism, coined by [Ear86], can be given. Consider two distinct realities which are subject to the same physical laws. If the two realities are identical at some time t_0, then they were and will remain identical at all times.

Core Idea

Consequently, deterministic processes evolve as if the state they describe were following some predefined trajectory.

This trajectory results from the process $\underline{X}(t)$ evolving under an abstract velocity field in state space referred to as the *drift* $\underline{f}(\underline{X}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. For the sake of generality, the drift $\underline{f}(\underline{X}, t)$ is taken as a nonlinear function of both state and time. If the drift function and the state are known at some time t, the state at a subsequent instant $t + \mathrm{d}t$, can be inferred. Mathematically, this implies that the evolution of $\underline{X}(t)$ is governed by an ODE:

$$\frac{\mathrm{d}\underline{X}}{\mathrm{d}t} = \underline{f}(\underline{X}, t) \quad \text{or} \quad \mathrm{d}\underline{X}(t) = \underline{f}(\underline{X}, t)\,\mathrm{d}t. \tag{1}$$

Formally, for a given initial condition $\underline{X}_0 = \underline{X}(t_0)$, the deterministic trajectory followed by the state vector $\underline{X}(t)$ can be computed by direct time integration.

$$\underline{X}(t) = \underline{X}_0 + \int_{t_0}^{t} \frac{\mathrm{d}\underline{X}}{\mathrm{d}t'}\,\mathrm{d}t' = \underline{X}_0 + \int_{t_0}^{t} \underline{f}(\underline{X}(t'), t')\,\mathrm{d}t' \tag{2}$$

2.2.2 Practical example

Deterministic processes are plentiful in physics, as they are the most common type of process encountered in classical mechanics. To illustrate the concepts introduced in the previous section, let us consider the simple example of the oscillating pendulum sketched in Figure 1.

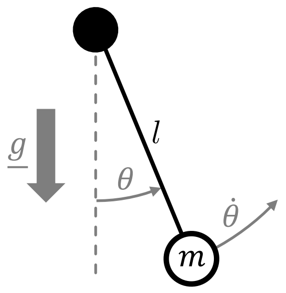


Figure 1: Illustration of the pendulum system. The pendulum consists of a mass m attached to a rod of length l and negligible mass, linked to a fixed pivot (black disk). Under the influence of a constant gravitational field $\underline{g}$, the pendulum oscillates. This movement is characterized by an angle θ and an angular velocity $\dot{\theta}$.

In classical dynamics, the equation of motion for this pendulum is simply

$$ml\ddot{\theta} = -mg\sin\theta. \tag{3}$$

Introducing the pulsation $\omega = \sqrt{g/l}$, the equation of motion can be rewritten as

$$\ddot{\theta} + \omega^2 \sin\theta = 0. \tag{4}$$

Using the state vector $\underline{X} = \left(\theta, \dot{\theta}\right)$, the latter equation can be recast as a system of first-order ODEs, falling into the form of Equation 1. One gets

$$\frac{\mathrm{d}\underline{X}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\theta \\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}\dot{\theta} \\ -\omega^2 \sin\theta\end{pmatrix} = \underline{f}(\underline{X}, t) \tag{5}$$

and the drift field is identified as

$$f(\underline{X}, t) = \begin{pmatrix}\dot{\theta} \\ -\omega^2 \sin\theta\end{pmatrix} = \begin{pmatrix}[\underline{X}]_2 \\ -\omega^2 \sin\,[\underline{X}]_1\end{pmatrix} \tag{6}$$

like represented in Figure 2.

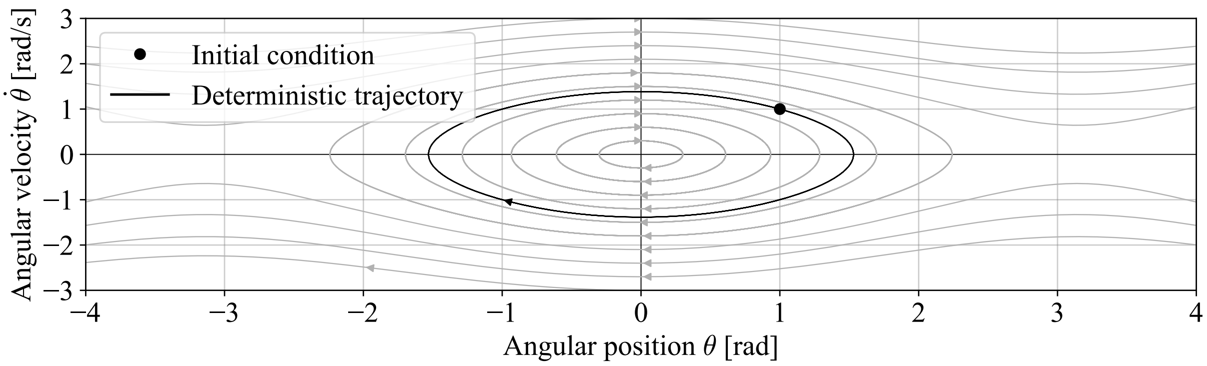


Figure 2: Drift field of a pendulum system for $\omega = 1$ [rad/s]. A realization is integrated from the initial condition $\underline{X}_0 = (1, 1)$ over an infinite time horizon.

In this example, the drift function is time-independent; however, this is not always the case. If the pendulum interacts with an environment subject to time-dependent perturbations, these can be modeled by a time-varying pulsation $\omega^2(t)$, leading to a time-dependent drift function $\underline{f}(\underline{X}, t)$. The impact of such interactions is illustrated in Figure 3 and Figure 4.

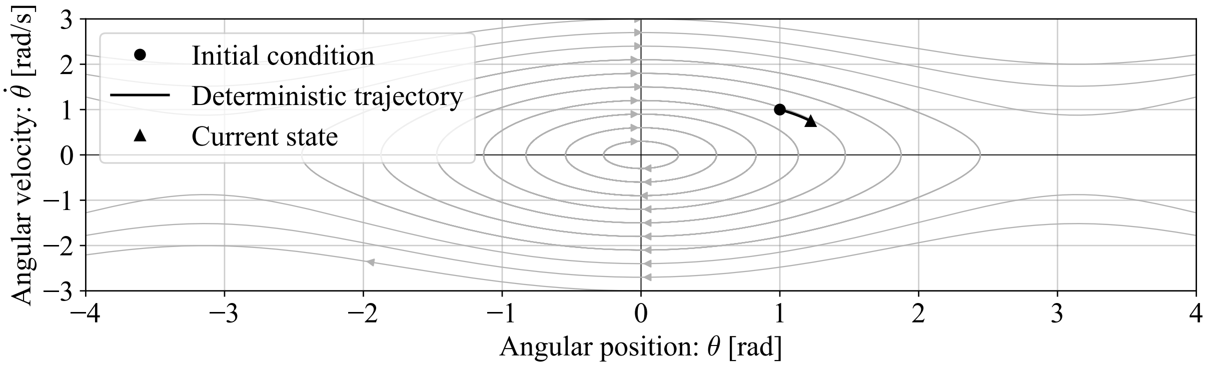


Figure 3: Drift field of a pendulum system for $\omega^2(t) = 1 + \sin(t/4) + \sin(3t/4)$. A realization is integrated from the initial condition $\underline{X}_0 = (1, 1)$ until $T = 0.25$ [s].

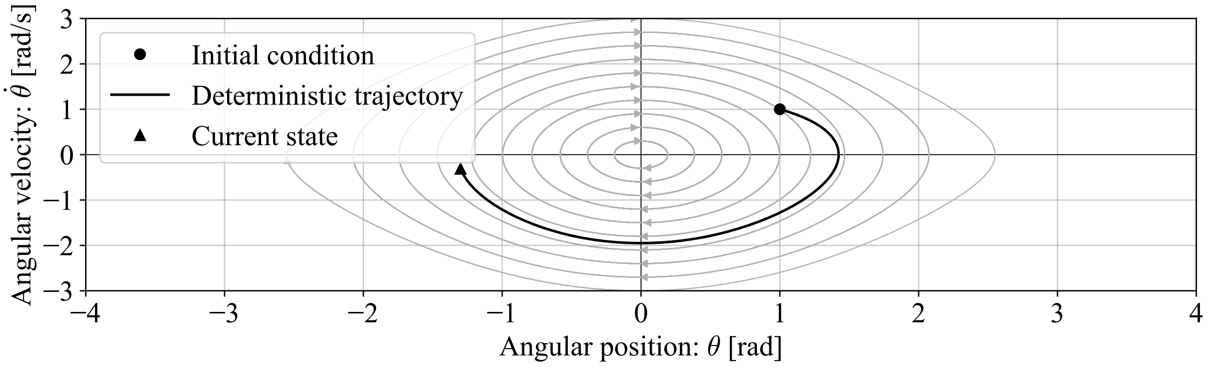


Figure 4: Drift field of a pendulum system for $\omega^2(t) = 1 + \sin(t/4) + \sin(3t/4)$. A realization is integrated from the initial condition $\underline{X}_0 = (1, 1)$ until $T = 3$ [s].

2.2.3 A note on chaotic systems

As a final note, let us treat the case of chaotic systems. Such systems are characterized by a deterministic drift function, which is highly sensitive to initial conditions. This means that two systems with very close initial conditions will diverge rapidly, leading to drastically different behaviors. Although the system's behavior may seem unpredictable with slight changes in initial conditions, it strictly remains deterministic, as the drift is uniquely defined at all times. For instance, a double pendulum exhibits a chaotic behavior, but its evolution is strictly the same for a given initial condition.

2.3 Stochastic processes and SDEs

2.3.1 Definition

As noted earlier, a process is deterministic if and only if $\underline{X}(t)$ is fully determined at any time t, given the drift function $\underline{f}(\underline{X}, t)$ and an initial condition $\underline{X}(t_0) = \underline{X}_0$. In contrast, stochastic processes can be defined as processes in which this predictability is no longer present. Instead, with the same knowledge about drift and an initial condition, $\underline{X}(t)$ is a random variable. In other words, a single stochastic process can lead to different realizations.

Such processes have real-life applications, which may be classified in two categories:
- inherently stochastic processes, which originate from probabilistic laws (e.g., quantum mechanics);
- deterministic processes modeled as stochastic processes, in which randomness is introduced to represent uncertainty or complexity in systems that are, in principle, deterministic (e.g., forces caused by turbulent flows).

The mathematical expression describing the evolution of a stochastic process can be formally written by adding a new term, $\underline{\gamma}(\underline{X}, t)$, to Equation 1.

$$\frac{\mathrm{d}\underline{X}}{\mathrm{d}t} = \underline{f}(\underline{X}, t) + \underline{\gamma}(\underline{X}, t) \tag{7}$$

In this expression, $\underline{\gamma}(\underline{X}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is the noise source term. It is a zero-mean random variable parametrized upon state and time; that is, for any given state $\underline{X}$ and time t,

$$E\left[\underline{\gamma}(\underline{X}, t)\right] = \underline{0}, \tag{8}$$

where E is the expectation functional.

In the present study, the noise source term is built upon m independent and Delta-correlated Gaussian White Noises (DGWNs) $\underline{w}(t) : \mathbb{R} \to \mathbb{R}^m$, with zero mean and unit variance. Mathematically, this means that $\underline{w}(t)$ is such that

$$w_i(t) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1) \quad \text{and} \quad E[w_i(t)w_i(t')] = \delta(t-t'), \quad \forall i \in \{1, \dots, m\}, \tag{9}$$

where $\delta(x)$ is the Dirac delta distribution defined as

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0, \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \int_{\mathbb{R}} \delta(x)\,\mathrm{d}V = 1. \tag{10}$$

The DGWN contributions of $\underline{w}$ are combined, thanks to $\underline{\underline{g}}(\underline{X}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n\times m}$ which is a nonlinear function of the state vector $\underline{X}$ and time t. Herein, this matrix is called the noise amplification matrix. The noise source term is then defined as $\underline{\gamma}(\underline{X}, t) = \underline{\underline{g}}(\underline{X}, t)\ \underline{w}(t)$ in such a way that Equation 7 becomes

$$\frac{\mathrm{d}\underline{X}}{\mathrm{d}t} = \underline{f}(\underline{X}, t) + \underline{\underline{g}}(\underline{X}, t)\ \underline{w}(t). \tag{11}$$

In reality, this equation only holds formally since the DGWN makes $\underline{X}(t)$ nowhere differentiable. To overcome this issue, one ought to use the framework of stochastic integration, such as Itō calculus (preferred in this work) or Stratonovich calculus [Ris89]. Refactoring Equation 7 as an SDE in the sense of Itō processes, one gets

Core Formula

$$\mathrm{d}\underline{X} = \underline{f}(\underline{X}, t)\,\mathrm{d}t + \underline{\underline{g}}(\underline{X}, t)\,\mathrm{d}\underline{W}(t) \tag{12}$$

where $\underline{W}(t) : \mathbb{R} \to \mathbb{R}^m$ is a vector of independent Gaussian Wiener processes [Hid03] defined as the stochastic integral of the DGWN vector $\underline{w}(t)$. Gaussian Wiener processes are continuous but nowhere differentiable stochastic processes which can be seen as a continuous extension of Markov random walks. In physics, a well-known example of a Gaussian Wiener process is Brownian motion, which describes the random motion of particles suspended in a fluid. More generally, Gaussian Wiener processes are Lévy processes characterized by the following properties [Dur19, Le 16, Shr04]:

- Initial condition:

$$P(\underline{W}(0) = \underline{0}) = 1. \tag{13}$$

- Independent increments: for any partition $0 \le t_0 < t_1 < \dots < t_n$, the increments

$$\underline{W}(t_1) - \underline{W}(t_0),\ \underline{W}(t_2) - \underline{W}(t_1),\ \dots,\ \underline{W}(t_n) - \underline{W}(t_{n-1}) \tag{14}$$

  are independent.

- Stationary Gaussian distribution: for any $t, s > 0$,

$$\underline{W}(t+s) - \underline{W}(t) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sqrt{s}). \tag{15}$$

- Continuity: $\underline{W}(t)$ is almost surely continuous in t.

Core Idea

Fundamentally, Equation 12 generates unique realizations of a stochastic process.

Caution

As such, Equation 12 can be integrated, in the Itō sense, from an initial condition to generate stochastic processes. It is important to emphasize that the only true source of randomness in this equation is the Gaussian Wiener process $\underline{W}(t)$. In contrast, the functions $\underline{f}(\underline{X}, t)$ and $\underline{\underline{g}}(\underline{X}, t)$ are predefined: for any given input, they yield consistent outputs. However, because the state $\underline{X}(t)$ is itself a random variable at any fixed time t, the evaluations $\underline{f}(\underline{X}, t)$ and $\underline{\underline{g}}(\underline{X}, t)$ also become random variables. Consequently, the Right Hand Side (RHS) of Equation 12 should not be interpreted as the sum of a deterministic term and a stochastic term; rather, the entire RHS is a stochastic quantity.

2.3.2 Practical example (continued)

Referring back to the pendulum example sketched in Figure 1, let us now consider that a DGWN perturbation is added to both $\theta(t)$ and $\dot{\theta}(t)$. This perturbation can be interpreted as any random disturbance in the environment of the pendulum. For this example, the equation of motion in Itō's sense reads

$$\mathrm{d}\begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}\dot{\theta}\\ -\omega^2\sin\theta\end{pmatrix}\mathrm{d}t + \underline{\underline{I}}_2\,\mathrm{d}\underline{W}(t) \tag{16}$$

Unlike in the deterministic case presented in Section 2.2.2, each realization of the system will now follow a different trajectory in state-space, even if they all start from the same initial condition and are subject to the same drift field, as shown in Figure 5.

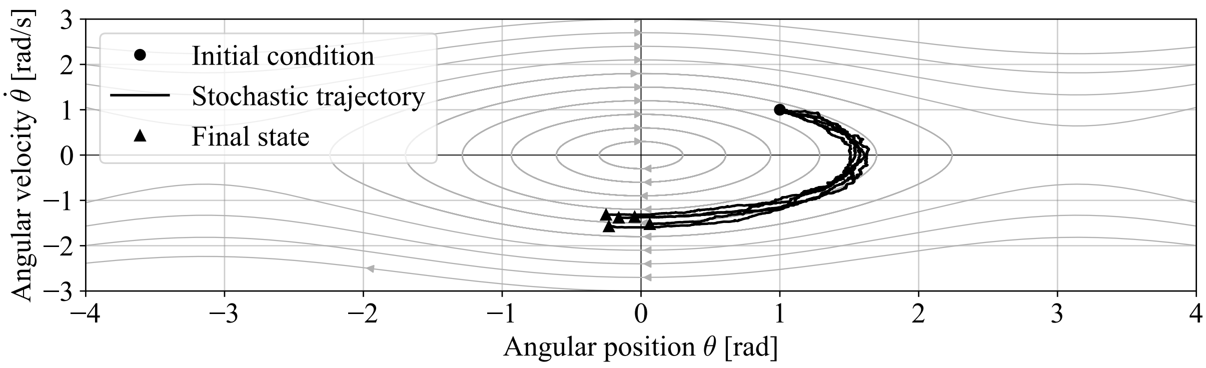


Figure 5: Drift field of a pendulum system for $\omega = 1$ [rad/s]. Five realizations of the pendulum system are shown, integrated from the initial condition $\underline{X}_0 = (1, 1)$ until $T = 3$ [s]. A continuous DGWN perturbation is considered on both $\theta(t)$ and $\dot{\theta}(t)$.

Ultimately, Figure 5 shows that each realization follows more or less the same trajectory. However, there exists a clear tendency for the realizations to diverge from each other as time progresses. Stated otherwise, the realizations tend to spread around a common deterministic trajectory. This observation is at the heart of the FPK equation, which is derived in section Section 2.5.

2.4 Monte-Carlo Simulation

Instead of considering individual realizations of a stochastic system, one can instead focus on the PDF $\psi(\underline{X}, t)$ which describes the probability density that a stochastic process occupies the state $\underline{X}$ at time t. To gather information regarding ψ, the MCS method [RC04] is often the default choice, favored for its conceptual simplicity, ease of implementation, and broad applicability across a wide range of stochastic models.

Simply put, the MCS method involves generating a large number of realizations of the stochastic process governed by the SDE of interest. An empirical PDF can then be constructed from these realizations. As the number of sampled realizations increases, the empirical PDF is expected to converge towards the true PDF. Likewise, empirical estimators can also be inferred from the MCS realizations.

To generate the different realizations of the stochastic process of interest, the MCS method needs to integrate the corresponding SDE numerically. In that extent, the Euler-Maruyama method serves as a first-order integrator for integrating the SDE in Itō's sense. Given the finite integration time step Δt, the Euler-Maruyama method provides the following update formula [KP92]:

$$\underline{X}^{[k+1]} = \underline{X}^{[k]} + \underline{f}\big(\underline{X}^{[k]}, t^{[k]}\big)\Delta t + \underline{\underline{g}}\big(\underline{X}^{[k]}, t^{[k]}\big)\Delta \underline{W}^{[k]}, \tag{17}$$

where $\Delta \underline{W}^{[k]}$ is the increment of the Wiener process over the time step Δt:

$$\Delta \underline{W}^{[k]} = \underline{W}\big(t^{[k+1]}\big) - \underline{W}\big(t^{[k]}\big). \tag{18}$$

Thanks to the properties of Wiener processes, the increments $\Delta \underline{W}^{[k]}$ are identically and independently distributed according to a normal distribution with mean zero and variance Δt:

$$\Delta W_i^{[k]} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\Big(0, \sqrt{\Delta t}\Big), \quad \forall i \in \{1, ..., m\}. \tag{19}$$

Then, defining random standard normal variables $w_i^{[k]} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \forall i \in \{1, ..., m\}$, the Euler-Maruyama integration scheme is finally expressed as

$$\underline{X}_{n+1} = \underline{X}^{[k]} + \underline{f}\big(\underline{X}^{[k]}, t^{[k]}\big)\Delta t + \underline{\underline{g}}\big(\underline{X}^{[k]}, t^{[k]}\big)\underline{w}^{[k]}\sqrt{\Delta t}. \tag{20}$$

Interestingly, unlike classical integration schemes where the time increments appear linearly, the Euler-Maruyama time integration scheme involves a stochastic term which scales with the square root of time increments [Shr04]. This peculiar aspect breaks additivity of the time steps and highlights the fact that stochastic systems are fundamentally different from deterministic ones, and must be treated with special care.

2.5 Fokker-Planck-Kolmogorov equation

2.5.1 From Itō processes to the FPK equation

An alternative way for deriving the PDF associated to a stochastic process is to establish its corresponding FPK equation. In contrast to the MCS method, which simulates realizations and then infers the underlying PDF statistically, the FPK approach develops the SDE into a deterministic PDE describing the evolution of the PDF over time. The resulting FPK equation is then solved numerically, as detailed in Section 3, and statistical quantities can be extracted from the computed PDF.

Core Idea

The FPK equation is a deterministic PDE which describes the evolution of the PDF $\psi(\underline{X}, t)$ of occupying a state $\underline{X}$ at time t.

Let us now derive the FPK equation from the expression of the SDE at hand, in the sense of Itō. The proof is given in the context of a one-dimensional system, but it can be extended to higher dimensions. For more information, please refer to [Ris89].

First, recall that the SDE considered here is

$$\mathrm{d}X(t) = f(X,t)\,\mathrm{d}t + g(X,t)\,\mathrm{d}W(t). \tag{21}$$

Let $\psi(X', t+\tau \mid X, t)$ be the probability density of transitioning from the state X at time t to the state X' at time $t+\tau$. By definition of conditional probability density [Pre90], one has

$$\begin{aligned}\psi(X', t+\tau) &= \int_{\mathbb{R}} \psi(X,t)\psi(X', t+\tau \mid X, t)\,\mathrm{d}X & (22)\\ &= \int_{\mathbb{R}} \psi(X,t) \int_{\mathbb{R}} \psi(Y, t+\tau \mid X, t)\delta(Y-X')\,\mathrm{d}Y\,\mathrm{d}X. & (23)\end{aligned}$$

In this expression, the delta function's Taylor expansion can be determined using distributional derivatives. This leads to

$$\begin{aligned}\delta(Y-X') &= \delta(X-X'+Y-X) & (24)\\ &= \sum_{n=0}^{+\infty} \frac{(Y-X)^n}{n!}\delta^{(n)}(X-X'), & (25)\end{aligned}$$

where $\delta^{(n)}$ denotes the nth distributional derivative of the Dirac delta distribution.

Injecting back,

$$\begin{aligned}\psi(X', t+\tau) &= \int_{\mathbb{R}} \psi(X,t) \int_{\mathbb{R}} \psi(Y, t+\tau \mid X, t) \sum_{n=0}^{+\infty} \frac{(Y-X)^n}{n!}\delta^{(n)}(X-X')\,\mathrm{d}Y\,\mathrm{d}X & (26)\\ &= \int_{\mathbb{R}} \delta^{(n)}(X-X')\psi(X,t) \int_{\mathbb{R}} \psi(Y, t+\tau \mid X, t) \sum_{n=0}^{+\infty} \frac{(Y-X)^n}{n!}\,\mathrm{d}Y\,\mathrm{d}X. & (27)\end{aligned}$$

Considering that X_t is a known, sharp realization of X at time t, one can write the nth raw moment of the state variation from time t to $t+\tau$ as

$$m_n(X_t, t, \tau) = E[(X(t+\tau) - X_t)^n] = \int_{\mathbb{R}} P(X, t+\tau \mid X_t, t)\ (X - X_t)^n\,\mathrm{d}X. \tag{28}$$

Injecting Equation 28 into Equation 27 yields

$$\begin{aligned}\psi(X', t+\tau) &= \int_{\mathbb{R}} \delta^{(n)}(X-X')\psi(X,t) \sum_{n=0}^{+\infty} \frac{1}{n!} m_n(X,t,\tau)\,\mathrm{d}X & (29)\\ &= \sum_{n=0}^{+\infty} \frac{1}{n!} \int_{\mathbb{R}} \delta^{(n)}(X-X')\psi(X,t) m_n(X,t,\tau)\,\mathrm{d}X. & (30)\end{aligned}$$

Each term of the sum is then integrated by parts n times. This way, the distributional derivatives of the delta function are transformed into classical derivatives applied to the rest of the integrand.

$$\psi(X', t+\tau) = \sum_{n=0}^{+\infty} \frac{1}{n!} \int_{\mathbb{R}} \delta(X-X')(-\partial_X)^n(\psi(X,t) m_n(X,t,\tau))\,\mathrm{d}X \tag{31}$$

$$= \sum_{n=0}^{+\infty} (-\partial_{X'})^n \left(\frac{m_n(X', t, \tau)}{n!} \psi(X', t) \right) \tag{32}$$

Rearranging the terms and dividing by τ results in

$$\frac{\psi(X', t+\tau) - \psi(X', t)}{\tau} = \sum_{n=1}^{+\infty} (-\partial_{X'})^n \left(\frac{1}{\tau} \frac{m_n(X', t, \tau)}{n!} \psi(X', t) \right). \tag{33}$$

Finally, taking the limit $\tau \to 0$ (which also implies $X' \to X$) yields the Kramers-Moyal expansion of ψ:

$$\partial_t \psi = \sum_{n=1}^{+\infty} (-\partial_X)^n (K_n \psi), \tag{34}$$

where the

$$K_n(X, t) = \lim_{\tau \to 0} \frac{1}{\tau} \frac{m_n(X, t, \tau)}{n!} \tag{35}$$

are the Kramers-Moyal expansion coefficients. In Itō's sense, these coefficients are identified as

$$K_1(X, t) = f(X, t) \quad \text{and} \quad K_2(X, t) = \frac{1}{2} g^2(X, t) \quad \text{and} \quad K_{n \geq 3} = 0. \tag{36}$$

Injecting these expressions back into Equation 34, the FPK equation, also known as the *Fokker-Planck* equation or the *forward Kolmogorov* equation, is finally obtained:

$$\partial_t \psi = -\partial_X (f \psi) + \frac{1}{2} \partial_X^2 (g^2 \psi). \tag{37}$$

As a side note, if the Kramers-Moyal coefficients were defined using Stratonovich calculus instead of Itō calculus, K_1 would include an additional term: $\partial_X g^2/2$. This term accounts for the "noise-induced drift," also known as the "spurious drift," "Stratonovich correction," or "Wong-Zakai correction".

Extending the one-dimensional FPK equation expressed by Equation 37 to higher dimensions, one can write the FPK equation as

Core Formula

$$\partial_t \psi = -\nabla \cdot \left(\underline{f} \psi \right) + \frac{1}{2} (\nabla \otimes \nabla) : \left(\underline{\underline{g}}\ \underline{\underline{g}}^T \psi \right) \equiv -\nabla \cdot \left(\underline{f} \psi \right) + (\nabla \otimes \nabla) : \left(\underline{\underline{D}} \psi \right) \tag{38}$$

or in index notation,

Core Formula

$$\partial_t \psi = -\sum_{i=1}^{n} \partial_i (f_i \psi) + \sum_{i,j,k=1}^{n} \frac{1}{2} \partial_{ij} (g_{ik} g_{jk} \psi) \equiv -\sum_{i=1}^{n} \partial_i (f_i \psi) + \sum_{i,j=1}^{n} \partial_{ij} (D_{ij} \psi), \tag{39}$$

where $\underline{\underline{D}}(\underline{X}, t) = \underline{\underline{g}}\ \underline{\underline{g}}^T/2 : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n \times n}$ is the diffusion matrix.

2.5.2 Auxiliary conditions

It is worth noting that the problem is transient, which means that some initial and boundary conditions on ψ must be given. These auxiliary conditions are usually given in the form of a PDF $\psi(\underline{X}, t_0)$ at time t_0, which must satisfy the

- positivity condition:

$$\psi(\underline{X}, t_0) \geq 0, \tag{40}$$

- normalization condition:

$$\int_{\mathbb{R}^n} \psi(\underline{X}, t_0)\, \mathrm{d}V = 1, \tag{41}$$

- and far-field condition:

$$\lim_{\|\underline{X}\| \to +\infty} \psi(\underline{X}, t_0) = 0. \tag{42}$$

2.5.3 Parallel with fluid dynamics

To get a better grasp of the FPK equation, let us draw a parallel with the domain of fluid dynamics. More precisely, let us consider the example of a liquid dye being introduced in a known flow. Denoting $\rho(\underline{x}, t)$ the density of the dye at any given position $\underline{x} \in \mathbb{R}^3$ and time $t \in \mathbb{R}$, the evolution of the dye's density is governed by the advection-diffusion equation

$$\partial_t \rho = -\nabla \cdot \left(\underline{f} \rho\right) + (\nabla \otimes \nabla) : \left(\underline{\underline{D}} \rho\right) = -\sum_{i=1}^{3} \partial_i (f_i \rho) + \sum_{i=1}^{3} \partial_{ij} \left(D_{ij} \rho\right), \tag{43}$$

where $\underline{f}(\underline{x}, t)$ is the velocity field of the flow and $\underline{\underline{D}}(\underline{x}, t)$ is the molecular diffusion coefficient. Most often, the molecular diffusion coefficient is assumed constant and isotropic, yielding

$$\partial_t \rho = -\nabla \cdot \left(\underline{f} \rho\right) + D \nabla^2 \rho = -\sum_{i=1}^{3} \partial_i (f_i \rho) + D \sum_{i=1}^{3} \partial_{ii} \rho. \tag{44}$$

As such, Equation 43 and Equation 44 clearly follow the form of the FPK equation given by Equation 39. Let us now show that they can in fact be derived directly from stochastic calculus. The trajectory of a single particle of dye is dictated by advection in the ambient flow $\underline{f}$, along with molecular agitation, which can be modeled as Brownian motion. Since Brownian motion is simply the physical counterpart of the more abstract Wiener process, one obtains the following SDE.

$$\mathrm{d}\underline{x} = \underline{f}(\underline{x}, t)\, \mathrm{d}t + \mathrm{d}\underline{W} \tag{45}$$

The corresponding FPK equation then reads

$$\partial_t \psi = -\nabla \cdot \left(\underline{f} \psi\right) + \frac{1}{2} \nabla^2 \psi = -\sum_{i=1}^{3} \partial_i (f_i \psi) + \frac{1}{2} \sum_{i=1}^{3} \partial_{ii} \psi. \tag{46}$$

In this formulation, $\psi(\underline{x}, t)$ denotes the PDF for finding a dye particle at location $\underline{x}$ and time t. Intuitively, regions where ψ is larger correspond to regions where particles are more likely to be found, and hence where the dye is more concentrated, as illustrated by Figure 6.

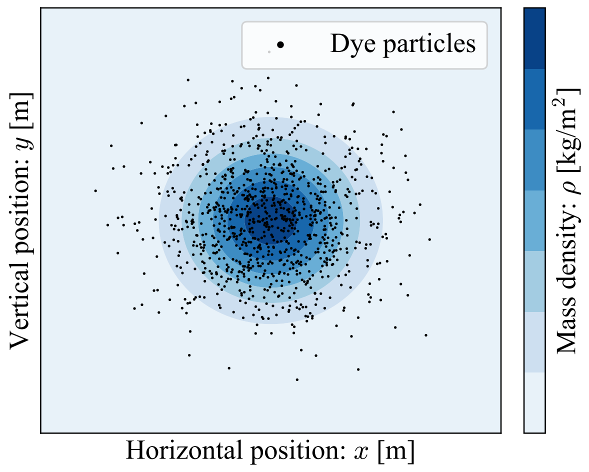


Figure 6: Illustration of the similarity between particle concentration and mass density.

Since each particle contributes equally to the total mass of the dye, the mass density field $\rho(\underline{x}, t)$ should be proportional to $\psi(\underline{x}, t)$. Specifically, if M denotes the total mass of dye in the system, one can express this relationship as

$$\rho(\underline{x}, t) = M\psi(\underline{x}, t). \tag{47}$$

Substituting this into Equation 46, one finally recovers Equation 44:

$$\partial_t \rho = -\nabla \cdot \big(\underline{f}\rho\big) + \frac{1}{2}\nabla^2 \rho = -\sum_{i=1}^{3} \partial_i (f_i \rho) + \frac{1}{2}\sum_{i=1}^{3} \partial_{ii}\rho, \tag{48}$$

which corresponds to $D = 1/2$ [m^2/s].

2.5.4 Practical example (continued)

Let us consider one last time the pendulum example described by Figure 1 and, more precisely, its stochastic version defined in Section 2.3.2. One can directly derive the FPK equation for this system and solve it to obtain the evolution of $\psi(\underline{X}, t)$.

As a reminder, the pendulum, in its stochastic form, was governed by the following SDE:

$$\mathrm{d}\begin{pmatrix}\theta \\ \dot{\theta}\end{pmatrix} = \underbrace{\begin{pmatrix}\dot{\theta} \\ -\omega^2 \sin\theta\end{pmatrix}}_{\underline{f}(\underline{X},t)} \mathrm{d}t + \underbrace{\begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}}_{\underline{\underline{g}}(\underline{X},t)} \mathrm{d}\underline{W}(t). \tag{49}$$

Instead of using Equation 49 to generate individual realizations of the system, one can construct the equivalent FPK equation and directly solve for the evolution of the PDF of the system's state. Thanks to Equation 39, the following FPK equation is obtained for the problem at hand.

$$\partial_t \psi = \begin{pmatrix}-\dot{\theta} \\ \omega^2 \sin\theta\end{pmatrix} \cdot \nabla\psi + \frac{1}{2}\nabla^2\psi \tag{50}$$

$$= \Big(-\dot{\theta}\partial_\theta + \omega^2 \sin\theta \; \partial_{\dot{\theta}} + \frac{1}{2}\partial_{\theta\theta} + \frac{1}{2}\partial_{\dot{\theta}\dot{\theta}}\Big)\psi \tag{51}$$

This new equation is solved for the same initial condition $\underline{X}_0 = (1, 1)$ as previously. This time however, it must be expressed in terms of probability density:

$$\psi(\underline{X}_0, 0) = \delta(\theta - 1)\delta\big(\dot{\theta} - 1\big). \tag{52}$$

In Figure 7, a rough illustration of the solution of Equation 51 is shown. What is important to understand here is that the FPK solution offers a comprehensive description of the PDF of states at any time t. Hence, the individual realizations considered earlier are just samples drawn from this PDF.

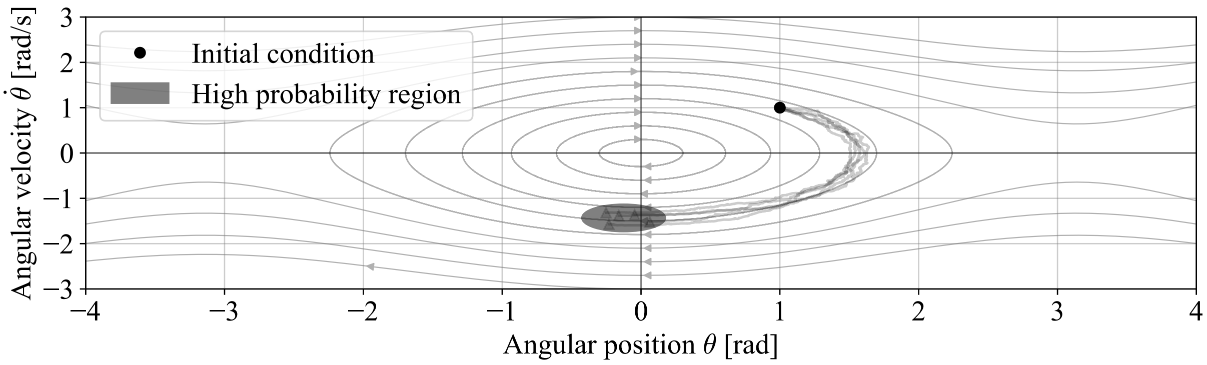


Figure 7: Drift field of a pendulum system for $\omega = 1$ [rad/s]. A rough illustration of the PDF at $t = 3$ [s] is proposed for a DGWN on both $\theta(t)$ and $\dot{\theta}(t)$. Some realizations are shown in light gray, for comparison with Figure 5.

2.6 Comparison between the MCS and FPK approaches

2.6.1 Theoretical discussion

To conclude, let us summarize and compare the two main approaches introduced in this part: the MCS and FPK methods. An overview of this discussion is presented in Table 1.

On the one hand, the SDE is formulated by modeling a stochastic process as the combination of a deterministic process and a scaled Gaussian Wiener process. By applying Itō calculus, the SDE can be integrated to generate multiple realizations of the stochastic process. Rather than focusing on individual sample paths, one may instead be interested in extracting information about the underlying statistical properties of the process. In such a case, statistical techniques, such as the MCS method, the maximum likelihood method or the method of moments are used to generate statistical estimators. Overall, the realizations are first simulated and then statistical quantities are estimated.

On the other hand, the FPK approach is obtained by applying Itō calculus to the original SDE describing the stochastic process of interest. This way, one gets an equation describing the evolution of the PDF itself. Once the PDF is known, any statistical information can be derived directly.

MCS	**FPK**
• System of coupled scalar SDEs	• Single scalar PDE
• Stochastic perturbations	• Deterministic diffusion
• Sharp realizations	• Probabilistic quantity
• Euler-Maruyama integration	• Deterministic resolution
• Inferred statistics	• Computed statistics

Table 1: Comparison between the MCS and FPK approaches to stochastic processes.

2.6.2 Practical example

To illustrate one last time the difference between the MCS and FPK approaches, let us consider a toy model. In particular, let $x(t)$ denote the position at time t of a particle subject to Brownian motion with drift. Considering a constant drift $f(x,t) = 1$ [m/s] and a noise amplification factor $g(x,t) = 1/2$ [m/$\sqrt{\text{s}}$], one can express the evolution of $x(t)$ through the following SDE:

$$\mathrm{d}x = f(x,t)\,\mathrm{d}t + g(x,t)\,\mathrm{d}W(t) = \mathrm{d}t + \frac{1}{2}\,\mathrm{d}W(t). \tag{53}$$

Moreover, a probabilistic initial condition can be considered. In particular, one can impose

$$x(0) \overset{\text{i.i.d.}}{\sim} \text{Unif}\left(-\frac{1}{2}, \frac{1}{2}\right), \tag{54}$$

where Unif(a,b) denotes a uniform distribution spanning from a to b.

Through repeated numerical stochastic integrations, performed via the Euler-Maruyama integration scheme, Equation 53 generates multiple realizations illustrated in Figure 8 (left). The corresponding underlying PDF is estimated in Figure 8 (right) by constructing a normalized histogram of these realizations, representing their empirical PDF.

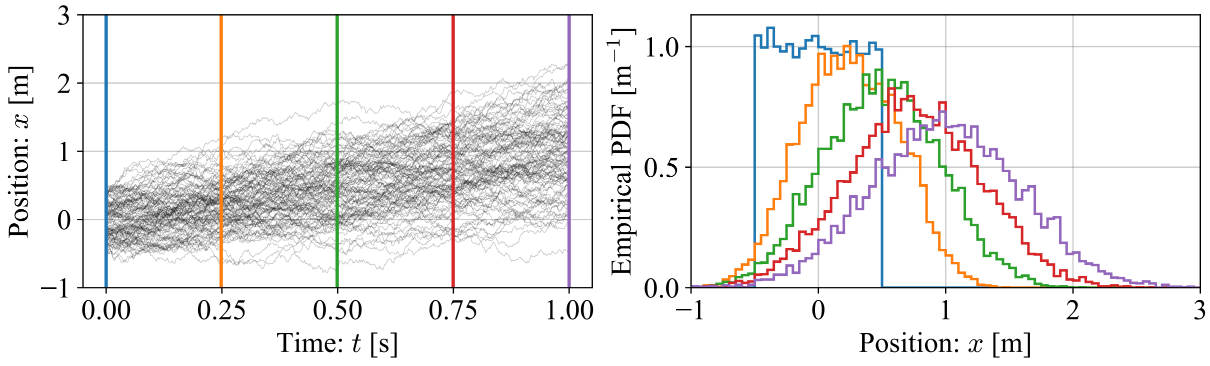


Figure 8: Illustration of how the MCS method is used to approximate the PDF $\psi(x,t)$ of finding the particle at some position x at the time t. **Left pane**: 100 realizations $x(t)$ are shown for a time horizon $T = 1$ [s]. **Right pane**: Empirical PDFs derived by MCS for 10000 realizations. The empirical PDFs are shown at $t = 0$ [s] (blue), $t = 0.25$ [s] (orange), $t = 0.5$ [s] (green), $t = 0.75$ [s] (red) and $t = 1$ [s] (purple).

Alternatively, one can formulate and solve the problem by employing the FPK formalism. Thanks to Equation 39, the process defined by Equation 53 is now described by

$$\begin{aligned}\partial_t \psi(x,t) &= -\partial_x (f(x,t)\psi(x,t)) + \frac{1}{2}\partial_{xx}(g^2(x,t)\psi(x,t)) &(55)\\ &= -\partial_x \psi(x,t) + \frac{1}{8}\partial_{xx}\psi(x,t). &(56)\end{aligned}$$

It is worth noting that Equation 56 is an advection-diffusion equation. Omitting the first term of the RHS yields Fick's law of diffusion. This illustrates once more how the FPK formalism bridges microscopic Brownian motion with macroscopic diffusion.

Regarding the initial condition, Equation 54 is reformulated as

$$\psi(x,0) = \begin{cases} 1 & \text{if } x \in \left[-\frac{1}{2}; \frac{1}{2}\right], \\ 0 & \text{otherwise} \end{cases}, \tag{57}$$

and the analytical solution for $t > 0$ is found to be

$$\psi(x,t) = \frac{1}{2}\left(\operatorname{erf}\left(\frac{x-t+1/2}{\sqrt{t/2}}\right) - \operatorname{erf}\left(\frac{x-t-1/2}{\sqrt{t/2}}\right)\right), \tag{58}$$

where $\operatorname{erf}(y)$ denotes the error function, defined as

$$\operatorname{erf}(y) = \frac{2}{\sqrt{\pi}}\int_0^y e^{-z^2}\,\mathrm{d}z. \tag{59}$$

The results are shown in Figure 9.

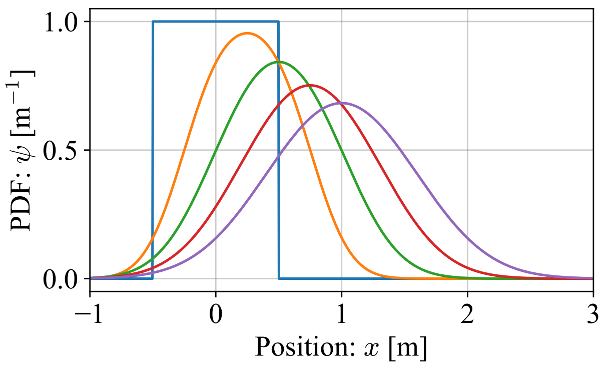


Figure 9: Solution of the FPK equation defined by Equation 56, for $t = 0$ [s] (blue), $t = 0.25$ [s] (orange), $t = 0.5$ [s] (green), $t = 0.75$ [s] (red) and $t = 1$ [s] (purple).

Overall, both approaches to the problem are valid since the empirical PDF illustrated in Figure 8 (right) converges to the exact PDF illustrated in Figure 9 as the number of realizations tends to infinity. For this reason, both approaches are used in the present work, for comparison purposes.

Ultimately, one key question remains: *"How is the deterministic resolution of the FPK equation performed?"*. This naturally arising question is the focus of the next chapter.

3 Smoothed Particle Hydrodynamics for the Fokker-Planck-Kolmogorov equation

3.1 Introduction

Simply put, the present work could be summarized as "Solving a specific *equation* with a specific *method*". In this part, the *method*, namely the SPH method, is introduced step by step.

The SPH method is a numerical method used to approximate continuous fields and solve PDEs. First introduced by [Luc77] and [GM77], it has since been widely adopted and improved in several fields of study. To this date, solving the FPK equation with the SPH formalism — which, as a whole, is called the FPK-SPH method in the following — remains largely underexplored. And yet, the SPH formalism holds several properties which make it specially suited for solving the FPK problem [Can14]:

- *Positivity*: by construction, the method ensures that the field of interest (i.e., the PDF) stays non-negative over the entire state space.
- *Far-field vanishing*: by construction, the SPH method ensures that the field of interest (i.e., the PDF) tends to zero in the far-field.
- *Conservation*: by construction, the SPH method conserves the total field of interest, and in particular the unity of total probability.
- *Focus*: particles inherently concentrate in regions of the state space where the quantity of interest is significant, whereas traditional Finite Element Method (FEM) and Finite Difference Method (FDM) require meshing the entire domain, including areas of lesser relevance. This property grows in interest when considering high-dimensional problems.

Traditionally, methods like the FEM or the FDM rely on a predefined mesh of sampling points, in such a way that the interconnections between a sampling point and its neighbors are fixed [CB15]. On the contrary, the SPH method is *meshless* (or *meshfree*), meaning that it relies on mobile scattered sampling points, referred to as *particles*, for which no predefined connectivity exists. To continuously approximate the field of interest with a set of scattered particles, the SPH method draws upon *kernel* functions [LLL03, Par62], which also define how the particles interact with their neighbors at each instant.

In this part, the SPH method is introduced from the ground up with the PDF $\psi(\underline{X}, t)$ being the field of interest. Then the FPK equation is recast in the Lagrangian formalism, upon which the method is based. Following this, the main theoretical improvements to the method are presented.

3.2 Integral representation and continuous approximation

There exist several ways to introduce the SPH method. Here, the integral representation of a continuous function $\psi(\underline{X}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ serves as a starting point. For a given instant t, the integral representation of ψ with respect to the state space consists in the following identity.

$$\psi(\underline{X}, t) = \int_{\mathbb{R}^n} \psi(\underline{X}', t)\ \delta(\underline{X} - \underline{X}')\,\mathrm{d}V' \equiv (\psi * \delta)(\underline{X}, t), \tag{60}$$

where $\mathrm{d}V'$ is the differential "volume"[(2)] element in $\mathbb{R}^n$ associated with the state $\underline{X}'$, and $\delta(\underline{X}) \equiv \delta(\|\underline{X}\|)$ is the Dirac delta distribution already defined in Equation 10. All in all, the integral representation simply states that $\psi(\underline{X}, t)$ is equal to its convolution (denoted by the $*$ operator) over the state space with the Dirac delta distribution.

The core idea of the SPH method relies on the approximation of Equation 60 by introducing two simplifications, in order to make the integral representation numerically tractable. First, the Dirac delta distribution is approached by a *kernel* function $W(\underline{X}, h)$. This function has a finite support, the extent of which is governed by some parameter h, called the *smoothing length* of the kernel. By construction, the kernel function is such that

$$\lim_{h \to 0^+} W(\underline{X}, h) = \delta(\underline{X}). \tag{61}$$

Applying this first approximation of the integral representation identity [Pri12], one has

$$\psi(\underline{X}, t) = \int_{\mathbb{R}^n} \psi(\underline{X}', t)\ W(\underline{X} - \underline{X}', h)\,\mathrm{d}V' + \mathcal{O}(h^2), \tag{62}$$

by considering a symmetric kernel $W(\underline{X} - \underline{X}', h) = W(\underline{X}' - \underline{X}, h)$. Other requirements are imposed for proper kernel definition, as summarized in Section 3.3.

The second approximation approaches the integral with a discrete sum, which is a common numerical practice. Doing so, the integrand is sampled at a set of N discrete sampling points $\{\underline{X}_i\}_{i=1}^{N}$. Consequently, the differential probability mass $\psi(\underline{X}', t)\,\mathrm{d}V'$ turns into the finite probability mass Ψ_i associated with the sampling point $\underline{X}_i$. The sampled probability masses Ψ_i remain constant over time in such a way that the unity of total probability mass, expressed by

$$\sum_i \Psi_i = 1 \tag{63}$$

[(2)]Here and after, the term "volume" should not be understood as a physical three-dimensional volume, but rather as its extension to a n-dimensional state space.

is respected throughout the entire simulation, without any need for additional treatment.

Caution

Note that Ψ_i is not a direct sampling of ψ. In other words, $\Psi_i \not\equiv \psi(\underline{X}_i, t)$.

Overall, the resulting approximation is called the *continuous approximation* [Kos+19] (or *kernel approximation*, or *particle approximation* [CD13]) of ψ and is denoted in this document as $\langle\psi\rangle$. It reads

Core Formula

$$\psi(\underline{X}, t) \approx \langle\psi(\underline{X}, t)\rangle = \sum_{i=1}^{N} \Psi_i W(\underline{X} - \underline{X}_i, h_i) \tag{64}$$

and is such that

$$\lim_{N\to+\infty} \langle\psi(\underline{X})\rangle = \psi(\underline{X}). \tag{65}$$

Let us note that a particle-dependent smoothing length h_i is hinted by Equation 64; more information on the calculation of h_i can be found in Section 3.7.

Thanks to the continuous approximation, derivatives of ψ can also be approximated with ease. In the present work, only the gradient of ψ is of interest. It is computed as

$$\nabla\psi(\underline{X}) \approx \nabla\langle\psi(\underline{X})\rangle = \sum_{i=1}^{N} \Psi_i \nabla W(\underline{X} - \underline{X}_i). \tag{66}$$

In the end, the function ψ and its gradient are now approximated thanks to a set of mobile sample points which carry some local information about ψ. In scientific literature, these sample points are often referred to as *particles*, as they consist in mobile entities carrying meaningful information and are often used to model matter. The overall approximation is an interpolation driven by the kernel functions, which extend the information carried by each particle to their neighborhood. Assembling the contribution of each particle is called the continuous approximation, as illustrated in Figure 10, and is the essence of the SPH method.

Caution

It is important to keep in mind that the so-called particles are not physical entities, but rather mathematical constructs used to approximate continuous functions [Kos+19].

In Figure 10, the SPH continuous approximation is represented qualitatively for some one-dimensional function of interest. As illustrated, the continuous approximation is constructed by summing the individual contribution $\Psi_i W(\underline{X} - \underline{X}_i, h_i)$ of each particle i. The quality of the approximation clearly improves as the number of particles increases. The illustration also shows that the continuous approximation suffers from discontinuities in the target function, due to the limited resolution size imposed by the finite smoothing lengths. However, considering even more particles and reducing the smoothing length in consequence, the error caused by imperfect discontinuity representation becomes very small.

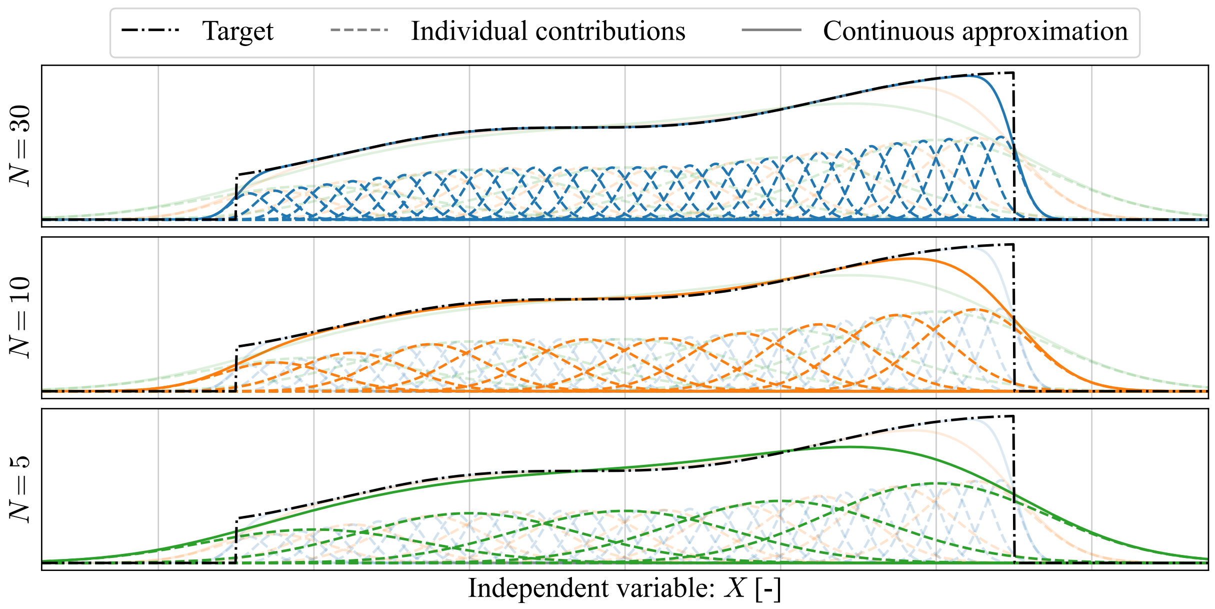


Figure 10: Qualitative representation of the continuous approximation of an arbitrary target function $\psi(X) : \mathbb{R} \to \mathbb{R}$, using a Gaussian kernel (see Section 3.3.2). The results are compared for different values of N. The smoothing length is updated according to $h \sim N^{-1}$.

3.3 Kernel function

The continuous approximation still misses the definition of the kernel function W. Herein, the general definition and properties of a proper kernel function are presented, followed by the definition of three specific kernel functions in $\mathbb{R}$. Additionally, the generalization of kernels to $\mathbb{R}^n$ is proposed, as the literature often limits itself to $\mathbb{R}$, $\mathbb{R}^2$ and $\mathbb{R}^3$.

3.3.1 Mathematical properties

As mentioned in the previous section, the kernel function $W(\underline{X}, h_i)$ is particle-dependent, as indicated by its parameter h_i. This dependency arises because the extent of the kernel can vary from one particle to another, while its shape remains unchanged. Nonetheless, the bivariate kernel function $W(\underline{X}, h)$ itself is considered a single, shared mathematical object, which is then individualized for each particle through the parameter h. The main properties [LLL03] that a proper kernel function must satisfy in the modern SPH framework are the following.

- *Dirac delta limit*: as already discussed in Section 3.2, the kernel function must converge to the Dirac-delta distribution when the smoothing length h tends to zero.

$$\lim_{h \to 0^+} W(\underline{X}, h) = \delta(\underline{X}). \tag{67}$$

- *Normalization*: the kernel function must be normalized to ensure that the integral of the kernel function over the whole space equals unity:

$$\int_{\mathbb{R}^n} W(\underline{X}, h) \, \mathrm{d}V = 1. \tag{68}$$

- *Compact support*: the kernel function must be zero outside a certain portion of the state space, which is defined by the smoothing length h and a kernel-dependent *scale factor* κ:

$$W(\underline{X}, h) = 0 \quad \text{if } \|\underline{X}\| > \kappa h, \tag{69}$$

  which results in a support radius κh.

- *Positivity*: the kernel function must be non-negative, that is
$$W(\underline{X}, h) \geq 0. \tag{70}$$
- *Smoothness*: the kernel function must be sufficiently smooth (at least continuously differentiable with respect to $\underline{X}$) within its support.
- *Isotropy* (implying *symmetry*): the kernel function must be of the form
$$W(\underline{X}, h) = C_n(h) W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \equiv C_n(h) W_{1\mathrm{D}}(\xi), \tag{71}$$
where $C_n(h)$ is a normalization factor depending on the dimension n of the state space and on the smoothing length h, while $W_{1\mathrm{D}}$ is a valid one-dimensional kernel function. The dimensionless variable $\xi \equiv \|\underline{X}\|/h$, will be used to simplify the notations.
- *Monotonous decrease*: the kernel function must be a monotonously decreasing function with respect to $\|\underline{X}\|$, that is
$$W(\underline{X}_1, h) \geq W(\underline{X}_2, h) \quad \text{if } \|\underline{X}_1\| \leq \|\underline{X}_2\|. \tag{72}$$

Any function that satisfies the aforementioned requirements can be used as a kernel function [LLL03], although other properties can be added depending on the specific problem [Mon05].

3.3.2 Specific kernel functions in $\mathbb{R}$

In the present work, three different kernels are considered: the cubic spline kernel, the Lucy kernel and the Gaussian kernel. Of course, many other kernels exist and can be used depending on the specific problem [LLL03, Pri12].

First, the Lucy kernel [Luc77] is employed, as it is used in the main reference [Can14] for the study. It has a scale factor $\kappa = 1$ and is characterized in $\mathbb{R}$ by the formula
$$W^{\mathrm{L}}(X, h) = C_1(h) W_{1\mathrm{D}}^{\mathrm{L}}(\xi) = \frac{5}{4h} \begin{cases} (1+3\xi)(1-\xi)^3 & \text{if } 0 \leq \xi < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{73}$$

Its X-derivative is
$$\partial_X W^{\mathrm{L}}(X, h) = \frac{\operatorname{sgn}(X)}{h} C_1(h) \big(W_{1\mathrm{D}}^{\mathrm{L}}\big)'(\xi) = \frac{5 \operatorname{sgn}(X)}{4h^2} \begin{cases} -12\xi(1-\xi)^2 & \text{if } 0 \leq \xi < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{74}$$

Second, the cubic spline kernel seems to be a natural choice since it is relatively standard in fluid mechanics [Kos+19], from which the SPH method partly originates [ML85]. Moreover, it has shown good results in a precedent work [BBD24] in which the author took part. The cubic spline kernel has a scale factor $\kappa = 2$ and is characterized in $\mathbb{R}$ by the following *B-spline*:
$$W^{\mathrm{C}}(X, h) = C_1(h) W_{1\mathrm{D}}^{\mathrm{C}}(\xi) = \frac{2}{3h} \begin{cases} 1 - \frac{3}{2}\xi^2 + \frac{3}{4}\xi^3 & \text{if } 0 \leq \xi < 1, \\ \frac{1}{4}(2-\xi)^3 & \text{if } 1 \leq \xi < 2, \\ 0 & \text{otherwise.} \end{cases} \tag{75}$$

Its X-derivative is

$$\partial_X W^{\mathrm{C}}(X,h) = \frac{\operatorname{sgn}(X)}{h} C_1(h) \big(W^{\mathrm{C}}_{\mathrm{1D}}\big)'(\xi) = \frac{2\operatorname{sgn}(X)}{3h^2} \begin{cases} -3\xi + \frac{9}{4}\xi^2 & \text{if } 0 \le \xi < 1, \\ -\frac{3}{4}(2-\xi)^2 & \text{if } 1 \le \xi < 2, \\ 0 & \text{otherwise.} \end{cases} \tag{76}$$

Third and lastly, the (truncated) Gaussian kernel is employed, as it is intrinsically linked to probability theory. This consideration aside, the Gaussian kernel is also one of the most popular kernels [Pri12] and has the property of being infinitely smooth within its support. It is introduced here for comparison purposes, given that applying the SPH method to the FPK equation is a relatively novel application. It has a scale factor $\kappa = 3$ and is characterized in $\mathbb{R}$ by the formula

$$W^{\mathrm{G}}(X,h) = C_1(h) W^{\mathrm{G}}_{\mathrm{1D}}(\xi) = \frac{1}{\operatorname{erf}(3) h \sqrt{\pi}} \begin{cases} \exp(-\xi^2) & \text{if } 0 \le \xi < 3, \\ 0 & \text{otherwise.} \end{cases} \tag{77}$$

Its X-derivative is

$$\partial_X W^{\mathrm{G}}(X,h) = \frac{\operatorname{sgn}(X)}{h} C_1(h) \big(W^{\mathrm{G}}_{\mathrm{1D}}\big)'(\xi) = \frac{\operatorname{sgn}(X)}{\operatorname{erf}(3) h^2 \sqrt{\pi}} \begin{cases} -2\xi\exp(-\xi^2) & \text{if } 0 \le \xi < 3, \\ 0 & \text{otherwise.} \end{cases} \tag{78}$$

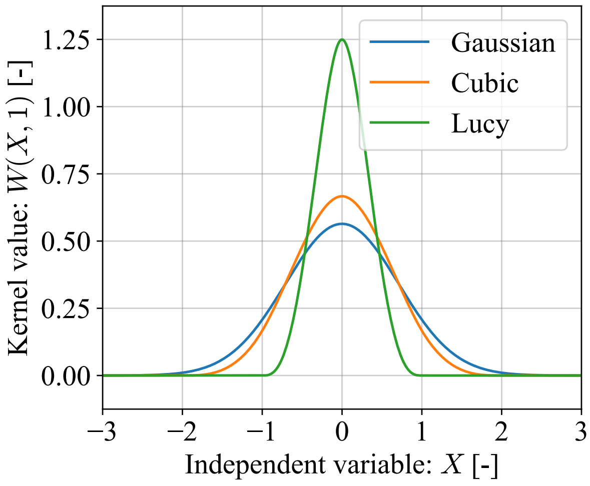


Figure 11: Comparison of the Gaussian, cubic and Lucy kernels for $h = 1$ [-].

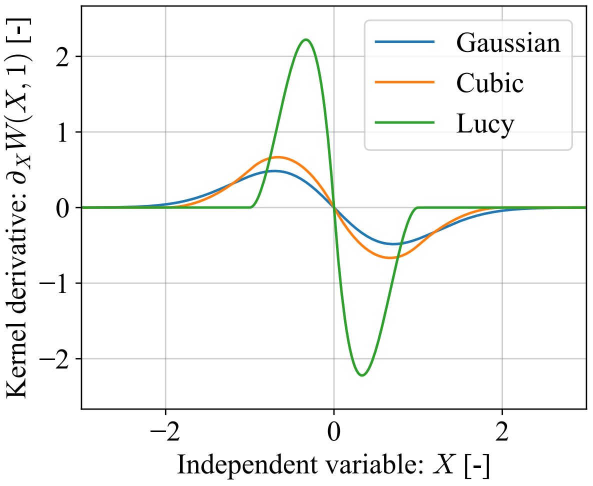


Figure 12: Comparison of the X-derivative of the Gaussian, cubic and Lucy kernels for $h = 1$ [-].

3.3.3 Kernel generalization in $\mathbb{R}^n$

Remembering Section 3.3.1, the isotropy requirement expressed by Equation 71 states that a kernel function W in n-dimensional state space must be of the form

$$W(\underline{X},h) = C_n(h) W_{\mathrm{1D}}\left(\frac{\|\underline{X}\|}{h}\right) \equiv C_n(h) W_{\mathrm{1D}}(\xi), \tag{79}$$

where W_{1D} is a known one-dimensional kernel, such as those presented in Section 3.3.2.

As such, the last unknown one still has to identify is the normalization factor $C_n(h)$. In scientific literature, the determination of $C_n(h)$ is often limited to $C_1(h)$, $C_2(h)$ and $C_3(h)$. However, the FPK-SPH method is not limited to physical space and can theoretically operate in any n-dimensional state space. In response, we present in this section a general methodology for deriving the analytical expression of $C_n(h)$.

By definition, the normalization factor is such that the kernel function W satisfies the normalization requirement of Equation 68 in $\mathbb{R}^n$, that is

$$\int_{\mathbb{R}^n} W(\underline{X}, h)\,\mathrm{d}V = C_n(h) \int_{\mathbb{R}^n} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \mathrm{d}V = 1 \tag{80}$$

$$\Leftrightarrow C_n(h) = \left(\int_{\mathbb{R}^n} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \mathrm{d}V\right)^{-1}. \tag{81}$$

First, owing to the isotropy property of the kernel function, $W_{1\mathrm{D}}$ is a radial function, whose integral over $\mathbb{R}^n$ is given by [Bak99]

$$\int_{\mathbb{R}^n} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \mathrm{d}V = nV_n \int_0^{+\infty} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \;\|\underline{X}\|^{n-1}\,\mathrm{d}\|\underline{X}\| \tag{82}$$

$$= nV_n h^n \int_0^{+\infty} W_{1\mathrm{D}}(\xi)\xi^{n-1}\,\mathrm{d}\xi, \tag{83}$$

where V_n is the volume of the unit ball in $\mathbb{R}^n$. Doing so, the n-dimensional integral is converted to a simple one-dimensional integral.

Second, the compact support property allows to restrict the integral to a finite domain.

$$\int_{\mathbb{R}^n} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \mathrm{d}V = nV_n h^n \int_0^{\kappa} W_{1\mathrm{D}}(\xi)\;\xi^{n-1}\,\mathrm{d}\xi \equiv nV_n h^n I_n. \tag{84}$$

Finally, introducing the expression of the volume of the unit ball in $\mathbb{R}^n$ and applying the Gamma function identity $\Gamma(z+1) = z\Gamma(z)$ yields

$$\int_{\mathbb{R}^n} W_{1\mathrm{D}}\left(\frac{\|\underline{X}\|}{h}\right) \mathrm{d}V = n\;\frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} h^n I_n = \frac{2\pi^{n/2}h^n}{\Gamma(n/2)} I_n. \tag{85}$$

In the end, coming back to Equation 81, the normalization factor $C_n(h)$ is simply

$$C_n(h) = \frac{\Gamma(n/2)}{2\pi^{n/2}h^n} I_n^{-1}. \tag{86}$$

In this final result, every factor is known. The fraction depends on the dimension n of the state variables but not on the kernel function, whereas the I_n factor can be computed analytically for the desired kernel function W, from its one-dimensional component $W_{1\mathrm{D}}$.

Applying the generalization to the aforementioned kernels, one can find the normalization factor C_n for the Lucy, cubic and Gaussian kernels. The results are

$$C_n^{\mathrm{L}}(h) = \frac{\Gamma(n/2)}{48\pi^{n/2}h^n}\; n(n+2)(n+3)(n+4), \tag{87}$$

$$C_n^{\mathrm{C}}(h) = \frac{\Gamma(n/2)}{12\pi^{n/2}h^n}\;\frac{n(n+1)(n+2)(n+3)}{2^{n+1}-1}, \tag{88}$$

$$C_n^{\mathrm{G}}(h) = \frac{\Gamma(n/2)}{\gamma(n/2,9)\pi^{n/2}h^n}\left(\approx \frac{1}{\pi^{n/2}h^n} \;\text{ for small } n\right), \tag{89}$$

where γ is the lower incomplete gamma function.

Fortunately, the same normalization factor can be used for the state derivative of the kernel function. Indeed, the gradient of the kernel function is simply given by

$$\nabla W(\underline{X}, h) = \frac{C_n(h)}{h}(W_{1\mathrm{D}})'(\xi)\ \underline{e}_X, \tag{90}$$

where $\underline{e}_X = \underline{X}/\|\underline{X}\|$ denotes the unit vector pointing in the direction of $\underline{X}$.

3.4 Initial condition

One variable still has to be determined in the continuous approximation formula, and that is the quantity Ψ_i carried by each particle. It is set thanks to an initial condition $\psi(\underline{X}, 0)$ on the PDF. In this work, two methods are presented for computing Ψ_i.

1. *Riemann approach*: The probability mass linked to particle i can be approximated by

$$\Psi_i = \psi(\underline{X}_i)V_i, \tag{91}$$

which comes to a Riemann approximation of the integral of ψ over the volume associated to the particle i, as depicted in Figure 13. Of course, some error is introduced by this approximation, in such a way that one must ensure normalization of total probability by scaling all the Ψ_i to enforce

$$\sum_{i=1}^{N} \Psi_i = 1. \tag{92}$$

2. *Collocation points approach*: the Ψ_i are chosen such that the corresponding continuous approximation of ψ is exact at the states occupied by particles.

$$\psi(\underline{X}_j, 0) = \langle \psi(\underline{X}_j, 0) \rangle = \sum_{i=1}^{N} \Psi_i W(\underline{X}_j - \underline{X}_i, h_i), \quad \forall j \in \{0, 1, ..., N\}. \tag{93}$$

This set of linear equations can of course be solved as a simple matrix equation. Yet again, normalization of total probability must be ensured afterward.

Amongst these two methods, the former is chosen, as the latter raises questions of ill-conditioning, according to [Can14].

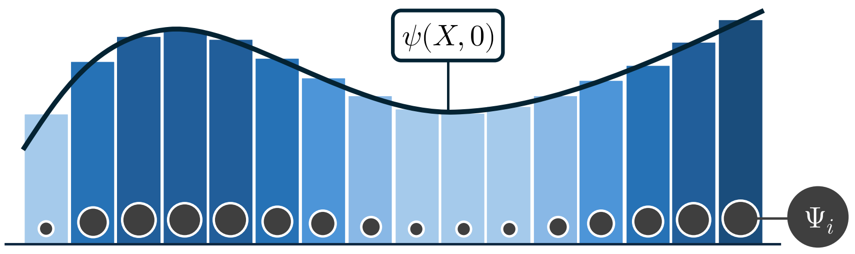


Figure 13: Illustration of the Riemann-based approach for initializing probability masses. The gray disks represent SPH particles, and their respective sizes correspond to the associated probability masses Ψ_i.

Regardless of the method employed to define the initial condition, a fundamental limitation of the approach relies in its inability to represent distributions with zero support, such as the Dirac delta distribution. Representing such singular distributions would require having infinitely close particles, which is not allowed by the method. In practice, such distributions can be approximated with regular kernels having a narrow but finite support. Nevertheless, this inevitably comes with a cost in terms of accuracy and computational power.

3.5 Lagrangian expression of the FPK equation

In this section, the aim is to convert the Eulerian form of the FPK equation into a Lagrangian form, which corresponds to the SPH framework. This requires deriving an expression for the advection field $\underline{v}$, which governs the transport of probability density through state space. To be more precise, $\underline{v}$ represents the *total* advection field, encapsulating both the influence of the drift $\underline{f}$ and the diffusion $\underline{\underline{D}}$.

By conservation of probability, the following continuity relation holds [Can14, CF86]:

$$\partial_t \psi = -\sum_{i=1}^{n} \partial_i (\psi v_i). \tag{94}$$

The Left Hand Side (LHS) of this equation can then be calculated using the FPK equation

$$\begin{aligned}\partial_t \psi &= -\sum_{i=1}^{n} \partial_i (f_i \psi) + \sum_{i=1}^{n} \sum_{j=1}^{n} \partial_{ij} (D_{ij} \psi) \qquad (95)\\ &= -\sum_{i=1}^{n} \partial_i \left(\psi \left(f_i - \sum_{j=1}^{n} \partial_j D_{ij} - \frac{1}{\psi} \sum_{j=1}^{n} D_{ij} \partial_j \psi \right) \right). \qquad (96)\end{aligned}$$

By identification between Equation 94 and Equation 96, the advection field is found to be

$$v_i = f_i - \sum_{j=1}^{n} \partial_j D_{ij} - \frac{1}{\psi} \sum_{j=1}^{n} D_{ij} \partial_j \psi. \tag{97}$$

Of course, in practice, ψ is unknown, so the advection field is computed using the continuous approximation of ψ, that is

Core Formula

$$v_i(\underline{X}, t) = f_i(\underline{X}, t) - \sum_{j=1}^{n} \partial_j D_{ij}(\underline{X}, t) - \frac{1}{\langle \psi(\underline{X}, t) \rangle} \sum_{j=1}^{n} D_{ij}(\underline{X}, t) \partial_j \langle \psi(\underline{X}, t) \rangle. \tag{98}$$

Finally, each particle i follows the advection field $\underline{v}(\underline{X}_i, t)$ through

$$\mathrm{d}_t \underline{X}_i(t) = \underline{v}(\underline{X}_i, t). \tag{99}$$

Ultimately, Equation 99 is integrated over time to compute the trajectories of the SPH particles. However, its RHS, defined by Equation 98, depends itself on $\langle \psi \rangle$, which in turn depends on the current state of the particles. This intricate relationship, summarized in Figure 14, constitutes the core computational loop of the method.

$$\begin{tikzcd} \text{Approximate } \psi \text{ with Equation 64} \arrow[r] & \text{Compute } \underline{v} \text{ with Equation 98} \arrow[dl] \\ \text{Particles } \underline{X}_i(t) \text{ drift along } \underline{v} \text{ with Equation 99} \arrow[u] & \end{tikzcd}$$

Figure 14: Minimal computational loop of the FPK-SPH method.

3.6 Time stepping

3.6.1 Euler time integration scheme

In order to update the particle states according to Equation 99, a time integration scheme is required. A straightforward approach is to use the explicit Euler method, which simply consists in a first order Taylor approximation of $\underline{X}(t)$. Given a partition of time $\left\{t^{[0]}, t^{[1]}, ..., t^{[K]}\right\}$ characterized by (possibly variable) increments $\Delta t^{[k]} = t^{[k+1]} - t^{[k]}$, the state of any particle i is updated according to

$$\underline{X}_i\left(t^{[k+1]}\right) = \underline{X}_i\left(t^{[k]}\right) + \mathrm{d}_t\underline{X}\left(t^{[k]}\right)\Delta t^{[k]} + \mathcal{O}\Big(\left(\Delta t^{[k]}\right)^2\Big) \tag{100}$$

$$\Leftrightarrow \underline{X}_i^{[k+1]} \approx \underline{X}_i^{[k]} + \underline{v}_i^{[k]}\Delta t^{[k]}. \tag{101}$$

3.6.2 Midpoint time integration scheme

The explicit midpoint time integration method is a second-order accurate scheme belonging to the family of second order Runge-Kutta methods. It consists in updating the state of each particle i at time $t^{[k+1]}$ by computing an intermediate state at time $t^{[k+\frac{1}{2}]} \equiv t^{[k]} + \Delta t^{[k]}/2$. A whole integration step is performed by the following successive updates [Can14]:

$$\underline{X}_i^{[k+\frac{1}{2}]} = \underline{X}_i^{[k]} + \underline{v}_i^{[k]}\frac{\Delta t^{[k]}}{2}, \tag{102}$$

$$\underline{v}_i^{[k+\frac{1}{2}]} = \underline{v}\Big(\underline{X}_i^{[k+\frac{1}{2}]}, t^{[k+\frac{1}{2}]}\Big), \tag{103}$$

$$\underline{X}_i^{[k+1]} = \underline{X}_i^{[k]} + \underline{v}_i^{[k+\frac{1}{2}]}\Delta t^{[k]}. \tag{104}$$

3.6.3 Adaptive time stepping

Both time integration methods rely on finite time increments $\Delta t^{[k]}$, which can be either constant or adaptive. Clearly, setting a constant time step $\Delta t^{[k]} = \Delta t, \forall k$ is not the best option since it requires the time step to be chosen small enough to ensure numerical stability at the most unstable point in the simulation. Consequently, the entire simulation is constrained by the strictest stability requirement, even during phases where a larger time step would be acceptable. Ideally, one would prefer having a variable time step which continuously adapts throughout the simulation, in such a way that the time step is small enough to ensure stability, but large enough to avoid excessive computational time.

At a given instant, a good candidate for the time step is given by a fraction of the minimal time step $\Delta t_c^{[k]}$ needed for any particle i to reach its closest neighbor with its current speed, as proposed by [Can14]. This criterion has a clear physical interpretation: if the time step is too large, particles may move too close to each other in a single time step, causing their interaction to become too strong and leading to numerical instability.

The time step is thus given by:

$$\left(\Delta t^{\text{abs}}\right)^{[k]} = S\,\left(\Delta t_c^{\text{abs}}\right)^{[k]} = S\min_{j\neq i}\frac{\|\underline{X}_i^{[k]} - \underline{X}_j^{[k]}\|}{\|\underline{v}_i^{[k]}\|}, \tag{105}$$

where S is a CFL-like control parameter (typically $S < 1$) that can be adjusted to control the stability of the method.

We propose to alter this formula by considering the relative velocity between particles rather than the absolute velocity. Since particles approach one another at their relative speed, using this new criterion provides a finer estimate of the time needed for potential collisions to occur:

$$\left(\Delta t^{\mathrm{rel}}\right)^{[k]} = S\,\left(\Delta t_c^{\mathrm{rel}}\right)^{[k]} = S \min_{j \neq i} \frac{\|\underline{X}_i^{[k]} - \underline{X}_j^{[k]}\|}{\|\underline{v}_i^{[k]} - \underline{v}_j^{[k]}\|}. \tag{106}$$

Let us note that both criteria are based on the conservative assumption that the particles are moving straight towards each other. In reality, setting $S = 1$ will most unlikely cause any pair of particles to collide.

3.7 Adaptive smoothing length

3.7.1 Motivations

As it is said in Section 3.3.1, the smoothing length parameter h_i is particle-dependent. To illustrate why this is important, let us consider a pure diffusion problem in $\mathbb{R}$ discretized using the FPK-SPH method with Gaussian kernels. By nature, diffusion tends to spread out the SPH particles, in such a way that the fixed kernel supports become too small to cover the ever increasing gaps between the particles. This is the cause for several issues, the two main ones, illustrated in Figure 15, are explained herein.

- As the particles spread, some locations are not covered by any kernel, which yields a continuous approximation of zero at these locations. This does not make sense as the quantity of interest is expected to spread, not to disappear between particles.
- The continuous approximation at a state X_i can never be lower than $\Psi_i W(0, h_i)$, which is constant. Therefore, by construction, an unphysical lower bound exists for the approximated field at the particle states.

Fortunately, these issues can be avoided by adapting the smoothing length of each particle during the time integration. Doing so, the kernel of each particle spreads out according to the local density of the particles.

Let us now confront the issue of the constant smoothing length with a practical example. Consider a diffusion problem in $\mathbb{R}$ with a Gaussian kernel, given by

$$\partial_t \psi = \frac{1}{2} \partial_{XX} \psi \qquad \text{with } \psi(X, 0) = \begin{cases} 0.05 & \text{if } X \in [-10, 10], \\ 0 & \text{otherwise} \end{cases} \tag{107}$$

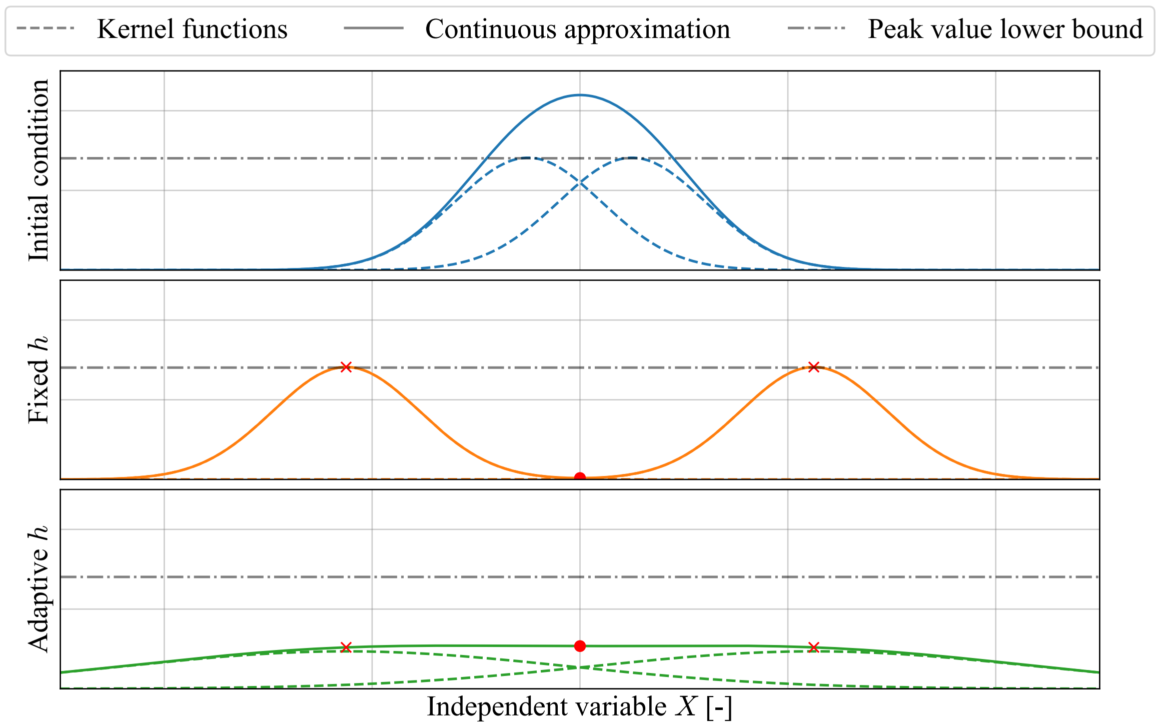


Figure 15: Illustration of a toy diffusion problem in $\mathbb{R}$ with a Gaussian kernel. Starting from an initial condition (top), two cases are considered: a constant smoothing length (middle) and an adaptive smoothing length (bottom). The constant smoothing length causes deserted regions (red spot) and fixed peak values (red cross) at the particle states. Using adaptive smoothing length, the kernels spread out and the continuous approximation is as expected.

A comparison of the results, with or without an adaptive smoothing length, is displayed in Figure 16.

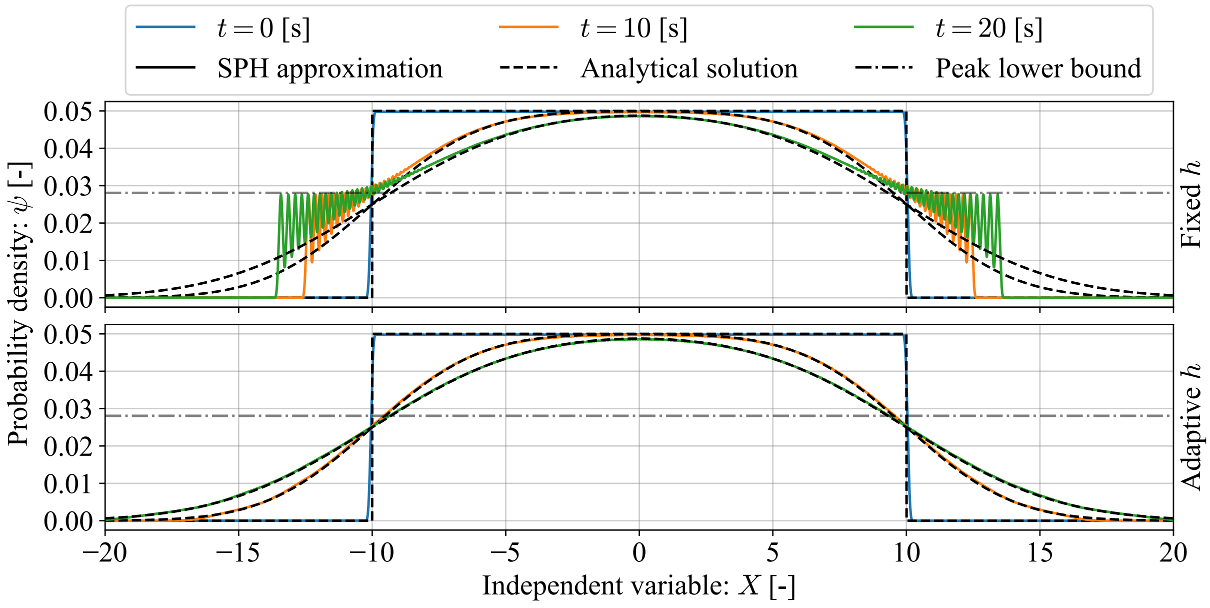


Figure 16: Comparison of the results produced when solving Equation 107 using a Gaussian kernel, with and without adaptive smoothing length. Initially, $h_i = 0.1$ [-] for all $N = 200$ particles, which sets the peak lower bound to $\Psi_i W^{\mathrm{G}}(0, h_i) = 0.05/(\mathrm{erf}(3)\sqrt{\pi}) \approx 0.028$ [-] in the case of fixed smoothing length.

3.7.2 Update formula

Clearly, the adaptive smoothing length is a must-have when applying the SPH method to the FPK equation, which intrinsically holds a diffusive component.

Let us consider, at some initial time step index $k = 0$, a regular lattice of particles in $\mathbb{R}^n$ with a constant spacing s. The corresponding smoothing length $h_i^{[0]}$ and the (constant) volume V_i of each particle i is set to

$$h_i^{[0]} = \eta s \quad \text{and} \quad V_i = s^n = \left(\frac{h_i^{[0]}}{\eta}\right)^n, \tag{108}$$

as illustrated in Figure 17. In the previous relations, η is the *coupling factor*, commonly chosen such that $\eta \in [1.2, 1.5]$ [Mon05].

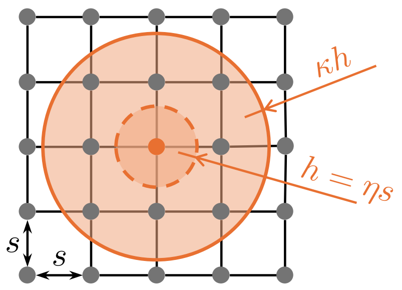


Figure 17: Illustration of the link between the smoothing length h and the spacing s of the SPH particles in the initial configuration (lattice).

By definition of the particle probability mass Ψ_i, one has

$$\Psi_i \approx \langle \psi^{[0]}(\underline{X}_i) \rangle V_i = \langle \psi^{[0]}(\underline{X}_i) \rangle \left(\frac{h_i^{[0]}}{\eta}\right)^n, \tag{109}$$

where the quality of the approximation depends on the refinement of the initial arrangement of particles, governed by the parameter s. Extending this relation for any time step k, one can link the corresponding smoothing length $h^{[k]}$ to the continuous approximation of the PDF by rearranging the terms. This yields the update formula

$$h_i^{[k]} = \eta \left(\frac{\Psi_i}{\langle \psi^{[k]}(\underline{X}_i) \rangle}\right)^{1/n}, \tag{110}$$

which virtually changes the extent of each particle i such that it keeps the same level of interaction with its neighbors [Can14, LLL03], compared to the initial configuration.

3.7.3 A new continuous approximation

This new definition of the smoothing length as a variable quantity raises two questions, the first of which is “how is the continuous approximation of the PDF affected?” The answer comes from a symmetry argument. For the interaction between two particles i and j to be reciprocal, the continuous approximation formula must be symmetric with respect to i and j, which is not the case yet. Instead, one can consider the average W_{ij} of the kernel evaluations using h_i and h_j. The approach, presented by [HK89] and employed by [Mon05], is utilized herein. The new continuous approximation formula reads

$$\left\langle \psi^{[k]}(\underline{X}_i) \right\rangle = \sum_{j=1}^{N} \Psi_j W_{ij}^{[k]} \equiv \sum_{j=1}^{N} \Psi_j \frac{W\left(\underline{X}_j - \underline{X}_i, h_i^{[k]}\right) + W\left(\underline{X}_j - \underline{X}_i, h_j^{[k]}\right)}{2}. \tag{111}$$

Similarly, the gradient is given by

$$\nabla \left\langle \psi^{[k]}(\underline{X}_i) \right\rangle = \sum_{j=1}^{N} \Psi_j \nabla W_{ij}^{[k]} \equiv \sum_{j=1}^{N} \Psi_j \frac{\nabla W\left(\underline{X}_j - \underline{X}_i, h_i^{[k]}\right) + \nabla W\left(\underline{X}_j - \underline{X}_i, h_j^{[k]}\right)}{2}. \tag{112}$$

3.7.4 Numerical computation of the smoothing length

The other question raised by Equation 110 comes from the numerical computation of the smoothing length. Indeed, the update of the smoothing length is not trivial, as it requires the continuous approximation of the PDF at each particle state, a quantity that depends itself on the smoothing length. To circumvent this issue, two iterative methods are explored: the fixed point method and the Newton-Raphson method. These two methods have their own advantages and disadvantages in terms of convergence speed and robustness, especially regarding the initial guess of the smoothing length. Both methods are implemented in the code, and the choice of the method to use is left to the user, as the optimal method may depend on the specific problem at hand. The two methods are presented in the following sections.

3.7.4.1 Fixed point method

The fixed point consists in iteratively updating the smoothing length via Equation 110 until its value converges. In other words, the process starts from an initial guess $h^{[k,0]}$, then computes the corresponding continuous approximation of the PDF and finally uses Equation 110 to compute the new smoothing length $h^{[k,1]}$. This process is repeated until the relative difference between two consecutive smoothing lengths meets a tolerance criterion, typically 10^{-3} [-] in this work. Mathematically, with l the iteration index of the fixed point method, the update formula is

$$h_i^{[k,l+1]} = \eta \left(\frac{\Psi_i}{\left\langle \psi^{[k,l]}(\underline{X}_i) \right\rangle} \right)^{1/n}, \tag{113}$$

where $\left\langle \psi^{[k,l]}(\underline{X}_i) \right\rangle$ uses the smoothing lengths of the iteration l. A natural choice for the initial guess is the value of the smoothing length at the previous time step, $h_i^{[k,0]} = h_i^{[k-1]}$.

3.7.4.2 Newton-Raphson method

The Newton-Raphson method is a numerical method used to find the roots of a function. In the present case, the function to solve is the error between the density according to Equation 110 and the continuous approximation of the density [Mon05], that is for particle i

$$\varepsilon_i = \Psi_i \left(\frac{\eta}{h_i} \right)^n - \sum_{j=1}^{N} \Psi_j W_{ij}. \tag{114}$$

Let l be the iteration index of the Newton-Raphson method. The update formula for the smoothing length at the time step k is

$$h_i^{[k,l+1]} = h_i^{[k,l]} - \varepsilon_i^{[k,l]} \left(\frac{\partial \varepsilon_i^{[k,l]}}{\partial h_i} \right)^{-1}, \tag{115}$$

with typically the initial guess $h_i^{[k,0]} = h_i^{[k-1]}$ and a relative tolerance criterion of 10^{-3}. After some development, the update formula becomes

$$h_i^{[k,l+1]} = h_i^{[k,l]} + \frac{\Psi_i\left(\eta/h_i^{[k,l]}\right)^n - \sum_{j=1}^N \Psi_j W_{ij}^{[k,l]}}{\frac{n\Psi_i}{h_i^{[k,l]}}\left(\eta/h_i^{[k,l]}\right)^n + \sum_{j=1}^N \Psi_j \frac{\partial W_{ij}^{[k,l]}}{\partial h_i}} \tag{116}$$

$$= h_i^{[k,l]}\left(1 + \frac{\Psi_i\left(\eta/h_i^{[k,l]}\right)^n - \sum_{j=1}^N \Psi_j W_{ij}^{[k,l]}}{n\Psi_i\left(\eta/h_i^{[k,l]}\right)^n + \sum_{j=1}^N \Psi_j\left(nW_{ij}^{[k,l]} + \left(\underline{X}_j^{[k]} - \underline{X}_i^{[k]}\right)\cdot\nabla W_{ij}^{[k,l]}\right)}\right). \tag{117}$$

3.8 Neighbor search

As already discussed when considering the core SPH computation loop, summarized in Figure 14, the SPH method strongly relies on the sampling of $\langle\psi\rangle$ at each particle state:

$$\langle\psi(\underline{X}_i, t)\rangle = \sum_{j=1}^N \Psi_j W_{ij}, \qquad \forall i \in \{1, ..., N\}. \tag{118}$$

Since each sampling of $\langle\psi\rangle$ is computed as a sum over all N particles, the overall computational cost is $\mathcal{O}(N^2)$. As a result, attempting to solve the FPK equation numerically becomes quickly intractable when the number of particles is large. To circumvent this issue, the SPH method defines the concept of *neighbors.* Let

$$\mathcal{N}_i = \left\{j \in \{1, ..., N\} : \|\underline{X}_i - \underline{X}_j\| < \kappa h_i \vee \|\underline{X}_i - \underline{X}_j\| < \kappa h_j\right\} \tag{119}$$

$$= \left\{j \in \{1, ..., N\} : \|\underline{X}_i - \underline{X}_j\| < \kappa \max\left(h_i, h_j\right)\right\} \tag{120}$$

be the set of neighbor particles of particle i. In other words, $\mathcal{N}_i$ is the set of all particles j such that j lies within the kernel support of i or vice versa. Thanks to the compact support property of the kernel, it follows that

$$W_{ij} = 0 \quad \text{if} \quad j \notin \mathcal{N}_i, \tag{121}$$

in such a way that the sampling of the continuous approximation given by Equation 111 can be written as

$$\langle\psi(\underline{X}_i, t)\rangle = \sum_{j=1}^N \Psi_j W_{ij} = \sum_{j\in\mathcal{N}_i} \Psi_j W_{ij}. \tag{122}$$

Only a limited number of terms have to be considered in the sum, thus sensibly reducing the computational cost of the operation. However, effort must still be invested in an efficient neighbor search algorithm, which is the purpose of this section.

The first step of the neighbor search is the *cell mapping.* It consists in dividing the state space at $t = 0$ into cells of size $\alpha\kappa h^{[0]}$ — the initial smoothing length is constant across all particles — with κ the already encountered scale factor and α (often ≥ 1 [Kos+19]) a user-defined *cell scale factor.* Each particle is then assigned to the cell it belongs to. All in all, this process has a computational cost of $\mathcal{O}(N)$ [Kos+19].

The second step, *neighbor pairing*, consists in finding for each particle its neighbors by searching only within the current and adjacent cells. For fixed smoothing lengths, the explanation stops here. However, for adaptive smoothing lengths, the neighbors might not all be found in the current and directly adjacent cells. In that case, the search must be extended to further cells, in such a way that several *layers* of neighbor cells are considered. The number of layers is given by

$$\lambda_i^{[k]} = \left\lceil \frac{h_i^{[k]}}{\alpha h_i^{[0]}} \right\rceil. \tag{123}$$

It is worth noting that in the case of decreasing smoothing lengths, the original neighbor search algorithm with one layer of neighbor cells is still valid.

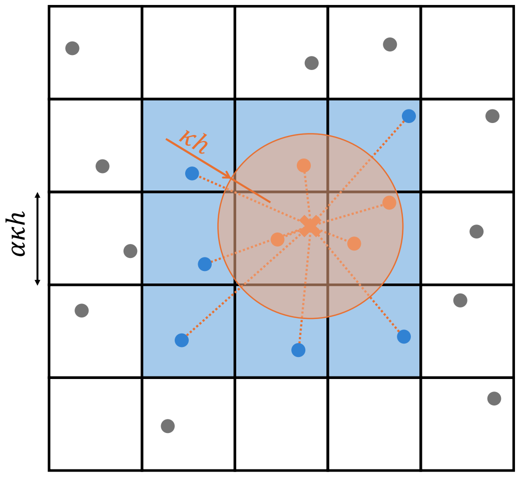


Figure 18: Illustration of the neighbor search algorithm. The neighbor cells (light blue) mark potential neighbors (dark blue), which are the only particles for which the distance (dashed orange) to the current particle (orange cross) is computed.

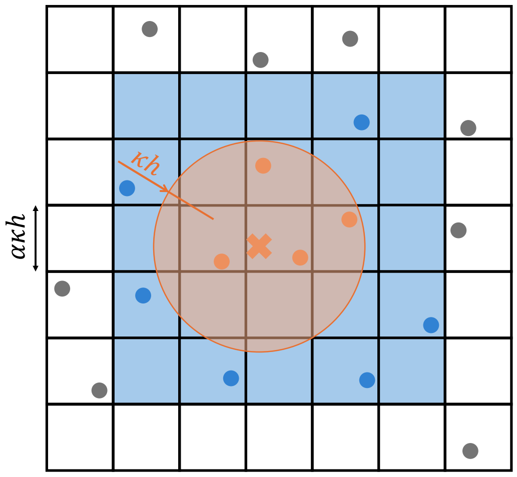


Figure 19: Illustration of the neighbor search algorithm with adaptive smoothing length. The neighbor cells (light blue) are extended to several layers to account for the variable smoothing length.

```
1  CELL MAPPING
2  for each particle $i$
3      assign particle $i$ to the cell it belongs to
4  NEIGHBOR PAIRING
5  for each particle $i$
6      compute the number of layers $\lambda_i^{[k]}$
7      for each neighbor cell $c$ in the $\lambda_i^{[k]}$ layers
8          for each particle $j$ in cell $c$
9              if $\|\underline{X}_j^{[k]} - \underline{X}_i^{[k]}\| < \kappa h_i^{[k]}$ or $\|\underline{X}_j^{[k]} - \underline{X}_i^{[k]}\| < \kappa h_j^{[k]}$
10                 add particle $j$ to the list of neighbors of particle $i$
```

Algorithm 1: Simplified neighbor search algorithm with adaptive smoothing length

Caution

Since the number of layers $\lambda_i^{[k]}$ is particle-dependent, a particle i may count a particle j as a neighbor, while the particle j has a too small smoothing length to perceive the particle i as a neighbor. In such an asymmetrical case, the particle i must add itself to the list of neighbors of particle j as well.

3.9 Boundary conditions

As discussed in Section 2.5.2, the FPK equation comes with a set of prescribed boundary conditions. Thanks to how the FPK-SPH method is formulated, all the boundary conditions are fulfilled. Specifically,

- positivity of ψ is ensured by the positivity of the kernels and probability masses,
- normalization of ψ is ensured by $\sum_{i=1}^{N} \Psi_i = 1$,
- far-field evanescence of ψ is ensured by the finite support of kernels.

Beyond the question of boundary conditions, we investigate different types of domains, namely the bounded, periodic and infinite domains, which are implemented in the solver. These types of domains come with potential adaptations to the cell mapping and neighbor pairing algorithms, which are also discussed in the following.

3.9.1 Bounded domain

The bounded domain is a fixed, regular region of state space in which the simulation is carried out, as shown in Figure 18. It was the first boundary condition implemented in this work because it is simple and computationally efficient. This method assumes that particles remain inside the domain at all times. If any particle reaches the boundary, the solver stops. As a result, the bounded domain is mainly used here for illustrative purposes, since it is easy to partition and requires no changes to the basic neighbor search algorithm. While it is limited for practical use, it can still be applied effectively if the domain is chosen large enough to ensure that no particle reaches the boundary during the simulation.

3.9.2 Periodic domain

The periodic domain is a type of domain in which particles that leave the domain on one side re-enter from the opposite side, making the domain topologically equivalent to an n-torus. This boundary condition is especially useful for intrinsically periodic problems, such as the one illustrated in Figure 20. It is also commonly used to simulate a small, representative patch of a virtually infinite domain by avoiding edge effects.

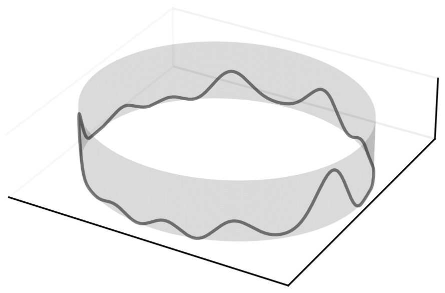


Figure 20: Sketch of a one-dimensional periodic domain, here a circle. The field of interest lives on the circle, making it periodic by nature.

When considering a periodic domain, attention must be paid to several points. First, the state of the particles must be corrected when they cross the boundary, as illustrated in Figure 21. Second, the neighbor search algorithm must be adapted to account for the periodicity of the domain. In particular, neighbor cells which are adjacent to the boundary must be considered as neighbors of the cells on the opposite side of the domain, as illustrated in Figure 22. Third and lastly, the distance computations must take into account the periodicity of the domain both when identifying the neighbors and when computing the continuous approximation of the PDF.

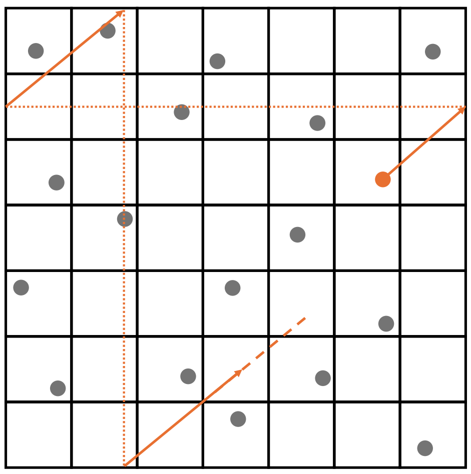


Figure 21: Illustration of the effect of the periodic boundary condition on a particle trajectory. The particle (orange) is brought back to the other side of the domain when it reaches a boundary.

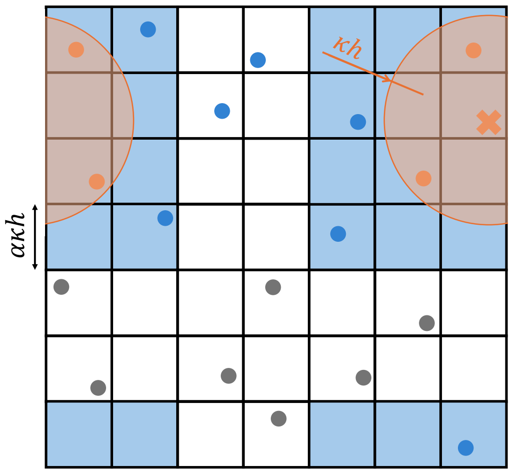


Figure 22: Illustration of the neighbor search algorithm with periodic boundary conditions. The neighbor cells (light blue) are extended to the other side of the domain and the distance computation (orange) takes into account the periodicity of the domain.

3.9.3 Infinite topology

The last boundary condition which is implemented in this work consists in an extension of the periodic boundary condition to simulate an infinite domain. The core idea is explained herein and represented in Figure 23.

1. At the beginning of the simulation, a finite region of state space is partitioned into cells, as it is done for the other topologies. This is represented with solid black lines.
2. During the simulation, the particles can move freely out of the finite domain.
3. During the cell mapping phase, each particle is virtually mapped back to the finite domain, using a modulo operation.
4. Potential neighbors are marked based on this virtual mapping of particles.
5. The neighbor pairing algorithm is then performed as usual, computing the true distance between the particles which are marked as potential neighbors.

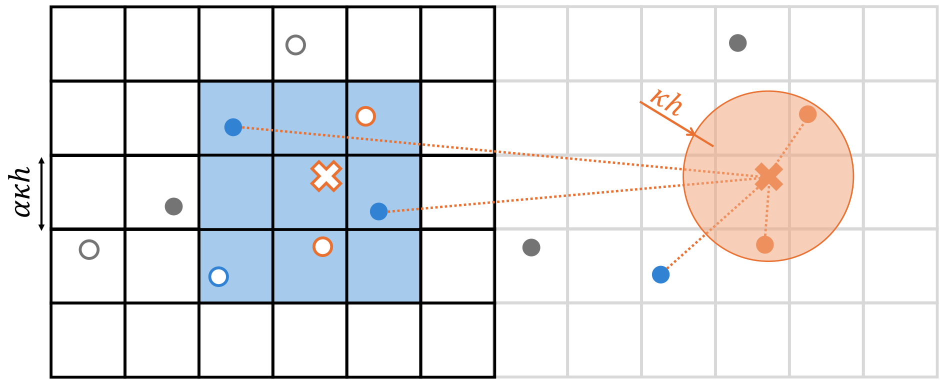


Figure 23: Illustration of the neighbor search algorithm with infinite boundary conditions. All the particles are mapped to cells of a finite grid (solid black lines), eventually through virtual particles (hollow shapes). The neighbor cells (light blue) contain the potential neighbors (dark blue and orange) for which the distance computation (dashed orange lines) is performed on the true particles (filled shapes). In the end, the neighbors (orange) are determined for the current particle (orange cross).

3.9.4 Redundancy

In the case of periodic and infinite boundary conditions, another adaptation must be added to the neighbor search algorithm. In particular, the number of layers λ cannot be too large, or it will cause some cells to be identified as neighbors several times, as illustrated in Figure 24. This caveat must be addressed when using the adaptive smoothing length, as the number of layers can grow large.

In the present work, this issue is addressed by

- removing neighbor cell duplicates to ensure that each neighbor cell is counted only once,
- enforcing an upper bound on the number of layers, specifically limiting λ to λ_{max}, defined as half the maximum number of cells along any dimension; this limits the memory usage when storing the reference to all the neighbor cells, before removing duplicates.

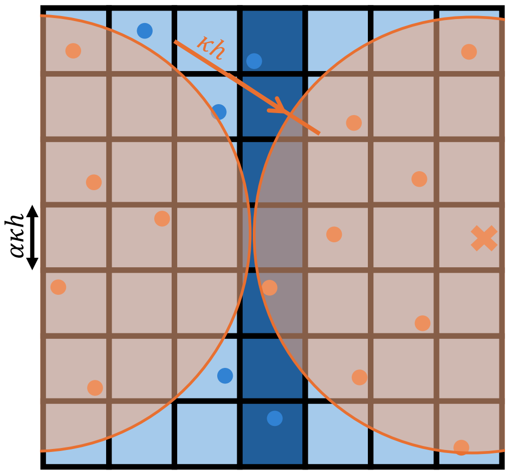


Figure 24: Some cells (dark blue) are erroneously identified several times as being neighbors of the current cell (containing the orange cross).

3.10 Approximation of statistical estimators

Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a deterministic function of a stochastic vector variable $\underline{X}$, and let E be the expectation functional. The expected value of φ [Pre90] is defined as

$$E[\varphi(\underline{X}(t))] = \int_{\mathbb{R}^n} \varphi(\underline{X}(t))\psi(\underline{X}, t)\, \mathrm{d}V \approx \sum_{i=1}^{N} \varphi(\underline{X}_i(t))\Psi_i. \tag{124}$$

In particular, this formula can be used to compute raw statistical cross-moments of the PDF via

Core Formula

$$m(a_1, a_2, ..., a_n) \equiv E[X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n}] = E\left[\prod_{j=1}^{n} X_j^{a_j}\right] \tag{125}$$

$$\approx \sum_{i=1}^{N} \Psi_i [\underline{X}_i]_1^{a_1} [\underline{X}_i]_2^{a_2} \cdots [\underline{X}_i]_n^{a_n} = \sum_{i=1}^{N} \Psi_i \prod_{j=1}^{n} [\underline{X}_i]_j^{a_j}. \tag{126}$$

In terms of central statistical cross-moments, this becomes

Core Formula

$$\overline{m}(a_1, a_2, ..., a_n) \equiv E\left[\prod_{j=1}^{n} \left(X_j - E\left[X_j\right]\right)^{a_j}\right] \tag{127}$$

$$\approx \sum_{i=1}^{N} \Psi_i \prod_{j=1}^{n} \left(\left[\underline{X}_i\right]_j - \sum_{i'=1}^{N} \Psi_{i'} \left[\underline{X}_{i'}\right]_j \right)^{a_j} \tag{128}$$

In a similar fashion, Equation 124 can be used to compute the probability of exceedance of a certain threshold $\nu(\underline{X}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$. For the sake of readability, let $\Omega(t) = \{\underline{X} \in \mathbb{R}^n : \nu(\underline{X}, t) > 0\}$ be the set of states for which the threshold is exceeded. The probability of exceedance is then

$$P(\underline{X} \in \Omega) = E[\mathbb{1}_\Omega(\underline{X})] \approx \sum_{i=1}^{N} \mathbb{1}_\Omega(\underline{X}_i)\Psi_i, \tag{129}$$

where the indicator distribution is defined as

$$\mathbb{1}_\Omega(\underline{X}) = \begin{cases} 1 & \text{if } \underline{X} \in \Omega(t), \\ 0 & \text{otherwise.} \end{cases} \tag{130}$$

4 Applications

4.1 Methodology

4.1.1 MCS moments computation

The MCS method, introduced in Section 2.4, is employed as a reference for evaluating the accuracy of the results produced by the FPK-SPH approach. In particular, selected statistical moments are computed from both the FPK-SPH (see Section 3.10) and MCS solutions. These moments are then compared to assess the consistency and performance of the proposed method.

The computation of moments for the MCS method is straightforward. Considering N samples, the raw cross-moments are computed as

$$m(a_1, a_2, ..., a_n) = E[X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n}] = E\left[\prod_{j=1}^{n} X_j^{a_j}\right] \tag{131}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} [\underline{X}_i]_1^{a_1} [\underline{X}_i]_2^{a_2} ... [\underline{X}_i]_n^{a_n} = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{n} [\underline{X}_i]_j^{a_j}, \tag{132}$$

while the central cross-moments are given by

$$\overline{m}(a_1, a_2, ..., a_n) = E\left[\prod_{j=1}^{n} \left(X_j - E[X_j]\right)^{a_j}\right] \tag{133}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{n}\left([\underline{X}_i]_j - \frac{1}{N}\sum_{i'=1}^{N}[\underline{X}_{i'}]_j\right)^{a_j}. \tag{134}$$

It is worth noting that Equation 132 and Equation 134 are very similar to their SPH counterparts, given by Equation 126 and Equation 128 respectively. The only difference resides in the fact that the MCS approach assumes $\Psi_i = 1/N$, meaning that each sample holds the same probability mass.

4.1.2 MCS confidence intervals

As is, directly considering the MCS statistical moments as a sharp reference would be misleading. In fact, the MCS method relies on statistical convergence, causing its estimations to come with some uncertainty, which must be quantified. In this section, the MCS samples are studied to derive a symmetric confidence interval $\mathcal{I}^{\mathrm{MC}}(t)$ which has a 95 [%] chance of containing $m(t)$ at any time t. Mathematically,

$$P\big(\mathcal{I}^{\mathrm{MC}}(t) \ni m(t)\big) = 0.95, \qquad \forall t. \tag{135}$$

To determine $\mathcal{I}^{\mathrm{MC}}$, the standard procedure consists in defining a *pivotal quantity* $Z = Z(m)$ which follows a well-known distribution $\mathcal{D}$. Once this pivotal quantity is established, a confidence interval $\mathcal{I}_Z^{\mathrm{MC}}$ for Z can be readily obtained by definition of the quantiles of $\mathcal{D}$. By inverting the relation linking Z to m, one can derive

$$\mathcal{I}^{\mathrm{MC}} = Z^{-1}\big(\mathcal{I}_Z^{\mathrm{MC}}\big). \tag{136}$$

Let us note that the procedure applies equally well to central statistical moments $\overline{m}(t)$ or any other statistical indicator derived from MCS samples. Focusing on the case of raw statistical moments, the following statistical pivot is defined:

$$Z(m) = \frac{m^{\mathrm{MC}} - m}{\Delta m^{\mathrm{MC}}/\sqrt{G}}, \tag{137}$$

where m^{MC} is the approximated value of m, computed thanks to Equation 132. The quantity Δm^{MC} is an empirical standard error on m^{MC}. To calculate it, the MCS samples are randomly separated into G groups. Then, for each group g, the value m_g^{MC} of the statistic of interest is computed. Finally, the unbiased standard deviation of m_g^{MC} across groups is given by

$$\Delta m^{\mathrm{MC}} = \sqrt{\frac{1}{G-1}\sum_{g=1}^{G}\big(m_g^{\mathrm{MC}} - m^{\mathrm{MC}}\big)^2}. \tag{138}$$

Thanks to this development, Z is known to be distributed according to a Student-t distribution with $G-1$ degrees of freedom [Fis25]. By definition of the 2.5 [%] quantile $Q_{t_{G-1}}(0.025)$ and 97.5 [%] quantile $Q_{t_{G-1}}(0.975)$ of the Student-t distribution, one has

$$P\Big(Z \in \big[Q_{t_{G-1}}(0.025); Q_{t_{G-1}}(0.975)\big]\Big) = 0.95. \tag{139}$$

Moreover, since the Student-t distribution is symmetric around the origin, one has $Q_{t_{G-1}}(0.025) = -Q_{t_{G-1}}(0.975)$ such that

$$P\left(\frac{m^{\mathrm{MC}} - m}{\Delta m^{\mathrm{MC}}/\sqrt{G}} \in \big[-Q_{t_{G-1}}(0.975); Q_{t_{G-1}}(0.975)\big]\right) = 0.95 \tag{140}$$

$$\Leftrightarrow P\left(m \in \left[m^{\mathrm{MC}} - \frac{Q_{t_{G-1}}(0.975)}{\sqrt{G}}\Delta m^{\mathrm{MC}}; m^{\mathrm{MC}} + \frac{Q_{t_{G-1}}(0.975)}{\sqrt{G}}\Delta m^{\mathrm{MC}}\right]\right) = 0.95. \quad (141)$$

Therefore, the confidence interval $\mathcal{I}^{\mathrm{MC}}$ can be chosen as

$$\mathcal{I}^{\mathrm{MC}} = \left[m^{\mathrm{MC}} - \frac{Q_{t_{G-1}}(0.975)}{\sqrt{G}}\Delta m^{\mathrm{MC}}; m^{\mathrm{MC}} + \frac{Q_{t_{G-1}}(0.975)}{\sqrt{G}}\Delta m^{\mathrm{MC}}\right]. \quad (142)$$

In the present case, the samples are divided into $G = 10$ groups, in such a way that the 95 [%] confidence interval particularizes to

$$\mathcal{I}^{\mathrm{MC}}(t) = \left[m^{\mathrm{MC}}(t) - 0.714\Delta m^{\mathrm{MC}}(t); m^{\mathrm{MC}}(t) + 0.714\Delta m^{\mathrm{MC}}(t)\right]. \quad (143)$$

For the sake of clarity and interpretability of the plots which appear in the different case studies, we define

$$\hat{Z} = \frac{Z}{Q_{t_{G-1}}(0.975)} = \frac{1}{Q_{t_{G-1}}(0.975)} \frac{m^{\mathrm{MC}} - m}{\Delta m^{\mathrm{MC}}/\sqrt{G}}, \quad (144)$$

which is a normalized version of Z, in the sense that its 95 [%] confidence interval is constantly $\mathcal{I}_{\hat{Z}}^{\mathrm{MC}} = [-1; 1]$:

$$P\Big(\hat{Z}(m(t)) \in [-1; 1]\Big) = 0.95. \quad (145)$$

In the following, the plots which show the evolution of the normalized pivotal quantity $\hat{Z}$ in the vicinity of a 95 [%] confidence interval are referred to as *confidence plots.*

Caution

It must be well-understood that the confidence interval defines a region where the exact solution likely lives, but the exact solution can eventually fall outside of it. By no means should the confidence interval plots which follow be interpreted as error plots.

4.1.3 Moment and cumulant equation methods

The present study led to a collaboration with a team of researchers at Tongji University working on an alternative method for simulating the evolution of statistical moments of the stochastic process $\underline{X}(t)$. They graciously agreed to share their results for comparison purposes with the FPK-SPH method. Specifically, the Duffing oscillator test case of Section 4.2 presents the results they obtained using their method, as well as runtime metrics. In this section, the Moment Equation Method (MEM) and Cumulant Equation Method (CEM) on which they work are briefly presented.

The MEM is a promising method used for estimating the evolution of the raw statistical moments of $\psi(\underline{X}, t)$ with a system of ODEs. In relatively recent studies, it has been developed further by [Den05, GF14, Lei+25] for nonlinear problems and non-Gaussian noise. Herein, it is used as a comparison point for assessing the capabilities of the FPK-SPH method. Basically, the MEM discretizes the PDF by transforming it into a finite number of statistical moments at discrete orders, whereas the FPK-SPH method discretizes the PDF as a combination of mobile particle kernels. Let us now briefly introduce the MEM.

Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a product function such that

$$\varphi(\underline{X}) = \prod_{r=1}^{n} X_r^{a_r}, \tag{146}$$

where, as a reminder, $\underline{X}(t)$ is an Itō stochastic process defined by

$$\mathrm{d}\underline{X} = \underline{f}(\underline{X}, t)\,\mathrm{d}t + \underline{\underline{g}}(\underline{X}, t)\,\mathrm{d}\underline{W}(t). \tag{147}$$

Thanks to Itō's formula [KT81], one can express the evolution of $\varphi(\underline{X}(t))$ as a stochastic Itō process governed by the equation

$$\mathrm{d}\varphi = \left(\sum_{i=1}^{n} f_i \partial_i \varphi + \frac{1}{2} \sum_{i,j,k=1}^{n} g_{ji} g_{ki} \partial_{jk} \varphi\right) \mathrm{d}t + \sum_{j=1}^{n} \left(\sum_{i=1}^{n} g_{ij} \partial_i \varphi\right) \mathrm{d}W_j. \tag{148}$$

Finally, due to the zero-mean property of the Wiener process (see Equation 15), applying the expectation functional on both members of the equation results in canceling the last term. Doing so, a regular ODE is recovered from the SDE of Equation 147 to determine the statistical moment $E[\varphi]$:

$$\mathrm{d}_t E[\varphi] = \sum_{i=1}^{n} E[f_i \partial_i \varphi] + \frac{1}{2} \sum_{i,j,k=1}^{n} E\left[g_{ji} g_{ki} \partial_{jk} \varphi\right]. \tag{149}$$

Rearranging the sums and substituting the expression of the derivatives of φ then yields

$$\mathrm{d}_t E[\varphi] = \sum_{i=1}^{n} a_i E[f_i X_i^{-1} \varphi] + \frac{1}{2} \sum_{\substack{i,j,k=1 \\ i \neq j}}^{n} a_i a_j g_{ik} g_{jk} E\left[X_i^{-1} X_j^{-1} \varphi\right] \tag{150(1)}$$

$$+ \frac{1}{2} \sum_{i,j=1}^{n} a_i (a_i - 1) g_{ij}^2 E[X_i^{-2} \varphi]. \tag{150(2)}$$

Now, let $\underline{m}_q$ be the vector containing all the raw statistical moments of order q. In other words, $E[\varphi]$ is an entry of $\underline{m}_q$ if and only if

$$\varphi(\underline{X}) = \prod_{r=1}^{n} X_r^{a_r} \quad \text{is such that} \quad \sum_{r=1}^{n} a_r = q \quad \text{and} \quad a_r \in \mathbb{N},\ \forall r. \tag{151}$$

If one assumes that the entries of $\underline{f}(\underline{X}, t)$ and $\underline{\underline{g}}(\underline{X}, t)$ are polynomials in $\underline{X}$ of at most order P_f and P_g respectively, then one can organize Equation 150 as

$$\mathrm{d}_t \underline{m}_q = \sum_{p=0}^{P_f} \underline{\underline{F}}_{q,p-1}(t) \underline{m}_{q+p-1} + \sum_{p=0}^{P_g} \underline{\underline{G}}_{q,2(p-1)}(t) \underline{m}_{q+2(p-1)}, \tag{152}$$

where the matrices $\underline{\underline{F}}$ and $\underline{\underline{G}}$ are determined from Equation 150 and the prior knowledge on $\underline{f}$ and $\underline{\underline{g}}$, respectively. In particular, it is worth noting that $\underline{\underline{F}}_{0,p-1}$, $\underline{\underline{G}}_{0,2(p-1)}$ and $\underline{\underline{G}}_{1,2(p-1)}$ are zero.

If $\underline{f}$ and $\underline{\underline{g}}$ are linear in $\underline{X}$, that is $P_f = P_g = 1$, one obtains the following system of first order ordinary differential equations.

$$\mathrm{d}_t \underline{m}_q = \underline{\underline{G}}_{q,-2}(t) \underline{m}_{q-2} + \underline{\underline{F}}_{q,-1}(t) \underline{m}_{q-1} + \left(\underline{\underline{F}}_{q,0}(t) + \underline{\underline{G}}_{q,0}(t)\right) \underline{m}_q, \tag{153}$$

in such a way that the LHS can be computed for each $q \in \{1, \ldots, Q\}$, where Q is the order for the method. By means of numerical integration, the evolution of all the moments up to order Q can be determined, starting from an initial condition $\underline{m}_q(0)$ which is directly computed from the analytical expression of $\psi(\underline{X}, 0)$. For the record, let us note that, unlike the FPK-SPH method, the MEM can easily describe a Dirac delta distribution since it can be defined with a limited number of statistical moments. On the flip side, several common distributions, such as the uniform distribution, require a large number of statistical moments to be defined accurately. For linear problems, this is not an issue since the analytical solution converges to a Gaussian distribution, which can be described exactly for $Q = 2$.

If $\underline{f}$ and $\underline{\underline{g}}$ are polynomials of arbitrary degree in terms of $\underline{X}$, the system becomes unclosed as the computation of $\mathrm{d}_t \underline{m}_Q$ typically depends on statistical moments of higher orders, which have not been solved yet. For example, if $P_f = 2$ and $P_g = 1$, one has for $q = Q$

$$\mathrm{d}_t \underline{m}_Q = \underline{\underline{G}}_{Q,-2}(t)\underline{m}_{Q-2} + \underline{\underline{F}}_{Q,-1}(t)\underline{m}_{Q-1} + \left(\underline{\underline{F}}_{Q,0}(t) + \underline{\underline{G}}_{Q,0}(t)\right)\underline{m}_Q + \underline{\underline{F}}_{Q,1}(t)\underline{m}_{Q+1}, \quad (154)$$

where $\underline{m}_{Q+1}$ is unknown. To circumvent this issue, one applies the cumulant-neglect closure technique [WSB96], which consists in assuming that the statistical cumulants of ψ are all 0 for orders larger than Q, thus providing enough information for closing the system.

As an enhancement, Equation 150 can be reformulated in terms of statistical cumulants instead of raw statistical moments, thus resulting in the CEM. This alternative approach strongly impacts the stability and the efficiency of the method for the best.

4.2 Duffing oscillator

4.2.1 Problem definition

Let us now consider a class of problems of increasing complexity. More precisely, one suggests to study a mass-damper-spring for which the spring is initially linear. In a second phase, the nonlinear regime is explored by progressively adding nonlinearity in the spring's stiffness. The setup is illustrated in Figure 25.

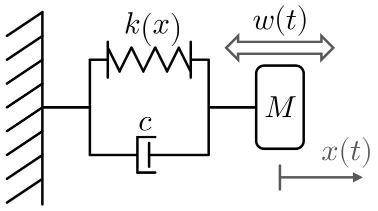


Figure 25: General mass-damper-spring oscillator with a nonlinear spring and an external excitation force.

The Duffing oscillator offers the perfect application for this approach. Setting $x = 0$ as being the equilibrium position in the absence of external excitation, the equation of motion reads

$$M\ddot{x}(t) + c\dot{x}(t) + k_0(1 + bx^2)x(t) = w(t), \quad (155)$$

where M is the mass of the moving object, c is the damping coefficient of the dashpot, and k is the stiffness of the spring. The b coefficient is a control parameter used to transition between the linear, weakly nonlinear and strongly nonlinear regimes. The DGWN w here acts as an external force on the system.

Considering a unit mass and the state vector $\underline{X} = (x, \dot{x})$, a proper definition of the problem is given by the following Itō SDE.

$$\mathrm{d}\underline{X} = \mathrm{d}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \underbrace{\begin{pmatrix} \dot{x} \\ -c\dot{x} - k_0(1+bx^2)x \end{pmatrix}}_{\underline{f}(\underline{X},t)} \mathrm{d}t + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}}_{\underline{\underline{g}}(\underline{X},t)} \mathrm{d}\underline{W}, \tag{156}$$

where the drift vector function $\underline{f}(\underline{X}, t)$ and the noise amplification matrix function $\underline{\underline{g}}(\underline{X}, t)$ are readily identified.

4.2.2 Steady state solution

This system converges to a steady state regime, characterized by the probability distribution

$$\psi_\infty(x, \dot{x}) = C(c, k_0, b) \exp\Big(-c\Big(\dot{x}^2 + k_0 x^2 + \frac{1}{2} k_0 b x^4\Big)\Big) \tag{157}$$

where C is a normalization constant.

Thanks to this theoretical result, the expected first and second moments of the system can be computed and used as reference for a large time horizon, which is here chosen as $T = 30$ [s]. The time horizon is chosen sufficiently large to capture the steady state configuration of the system, and thus permit comparison with ψ_∞. In order to quantify the accuracy of the different methods in terms of convergence towards the analytical steady state solution, one uses the (absolute) Steady State Relative Error (SSRE) defined as

$$\mathcal{E}[\varphi(\underline{X})] = \left| \frac{E[\varphi(\underline{X})] - E^{\text{method}}[\varphi(\underline{X})]}{E[\varphi(\underline{X})]} \right|. \tag{158}$$

4.2.3 FPK-SPH statement

For the problem at hand, the FPK-SPH method is applied to the FPK equivalent equation of Equation 156, and relies on the following configuration.

- Initial condition: one assumes an initial bivariate normal distribution with $\underline{0}$ mean and a standard deviation of 0.3 [m] for x and 0.3 [m/s] for $\dot{x}$, expressed by

  $$\psi(\underline{X}, 0) = C \exp\left(-\frac{x^2 + \dot{x}^2}{2 \times 0.3^2}\right), \tag{159}$$

  where C is a normalization factor. A Gaussian kernel is used.

- Initial particle arrangement: the $[-1.2, 1.2]^2$ interval of state space is covered with a regular lattice of particles. Amongst these particles, only those which lie within 4 times the standard deviation are kept, as the mass carried by the rest is negligible. This results in particles uniformly distributed in the disk $\Omega = \{\underline{X} : \|\underline{X}\| < 1.2\}$, as illustrated in Figure 26. Since the initial condition described in the previous point relies on an infinite support, only a truncated version of the initial condition is represented, which implies a normalization of the total probability held by the particles.

  For the Duffing model studied in this section, two different regimes are defined, in Section 4.2.6 and Section 4.2.7 respectively. For the first one, the *unimodal* regime ($k_0, b > 0$), the $[-1.2, 1.2]^2$ interval is uniformly covered with $49^2 = 2401$ SPH particles, resulting in $N = 1789$ particles effectively covering the disk Ω. For the second regime, the *bimodal* one ($k_0, b < 0$), $97^2 = 9409$ particles are uniformly distributed in the interval, which causes $N = 4049$ particles to be kept for covering Ω. This increase in the number of particles is justified by the fact that the evolution of ψ in the bimodal case is more complex and thus requires a better resolution in state space.

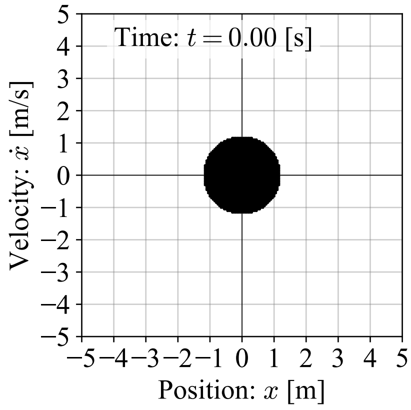


Figure 26: Initial arrangement of particles for all the Duffing test cases. N particles are arranged in a disk of radius 1.2, centered at the origin.

- Time integration: a time horizon of $T = 30[s]$ is simulated, using an adaptive time step and a forward Euler integration scheme. Regarding the adaptive time stepping, the *relative* and *absolute* conventions are utilized and compared. In both cases, the stability coefficient is set to $S = 0.1$ [-] and a maximum time step $\Delta t^{\max} = 5 \times 10^{-3}$ [s] is imposed.
- Efficiency: all tests are run on 64 cores and the total computation time is indicated in Table 4 for each variation of the problem at hand.

4.2.4 MCS statement

The MCS results are generated by direct stochastic integration of Equation 156, considering the following setup.

- Initial condition: the MCS samples start from a state $\underline{X}_0 = (x(0), \dot{x}(0))$ drawn from

$$x(0), \dot{x}(0) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.3). \tag{160}$$

- Sampling resolution: a total of $N = 10^4$ realizations is derived from the MCS.
- Time integration: as for the FPK-SPH simulation, a time horizon $T = 30$ [s] is considered, but using a fixed time step of $\Delta t = 10^{-3}$ [s]. The integration method is the Euler-Maruyama time integration scheme introduced in Section 2.4.

4.2.5 CEM statement

Without entering too much into details, let us only mention that the CEM is employed with a fixed time step $\Delta t = 0.05$ [s] and the raw statistical moments of

$$\psi(\underline{X}, 0) = C \exp\left(-\frac{x^2 + \dot{x}^2}{2 \times 0.3^2} \right), \tag{161}$$

where C is a normalization constant, are used as initial condition for the $\underline{m}_q(t)$ vectors.

For the sake of clarity, let us remind the reader that the CEM is not implemented in the present work, and the results presented in this section were obtained by collaborating with a team of researchers at Tongji University.

4.2.6 Unimodal regime

In this first part, the Duffing oscillator is studied in three scenarios, summarized in Table 2.

Scenario	c [Ns/m]	k_0 [N/m]	b [-]
Linear	0.4	1.0	0.0
Weakly nonlinear	0.4	1.0	0.1
Fully nonlinear	0.4	1.0	1.0

Table 2: Parameter values used in each Duffing unimodal scenario.

For these three test cases, the steady state PDF is unimodal (i.e., it presents a unique peak), as shown in Figure 27. This regime corresponds to the first set of tests presented in this section. Later, in Section 4.2.7, a bimodal steady state PDF is considered, causing significant effects on the system's dynamics and the solvers' performances.

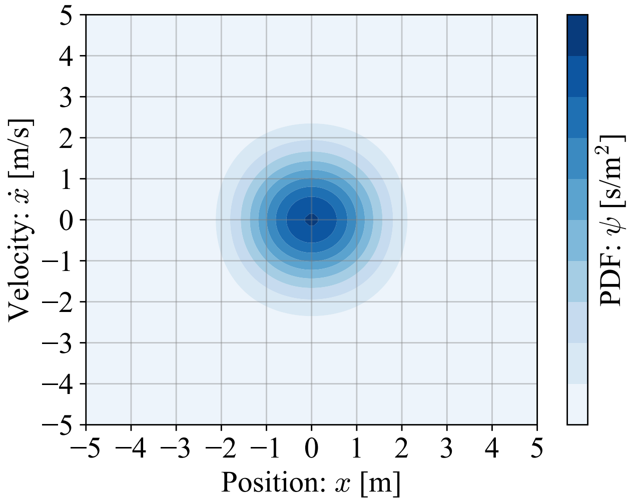


Figure 27: Contours of the steady state PDF of the unimodal Duffing oscillator, here obtained for the weakly nonlinear scenario ($b = 0.1$ [-]).

The results of the unimodal regime are presented under different forms. For each scenario, several figures are proposed:

- The first shows the evolution of the SPH particles, to give an appreciation of the underlying dynamics of the system.
- The second and fifth consist in plots of the evolution of the position and velocity second moments, respectively, for different methods, including MCS, FPK-SPH and CEM at different orders. The second moments are here assimilated to variances as the average position and velocity are known to be zero, by argument of symmetry. The analytical steady state variance derived from ψ_∞ is also indicated on the plots.
- The third and sixth showcase the evolution of the SSRE, defined in Section 4.2.2, for the aforementioned methods.
- Finally, the fourth and seventh figures are confidence plots for the FPK-SPH method.

The results corresponding to the *linear* scenario are represented in Figure 28 to Figure 34. In Figure 28, the disk of particles starts rotating and elongating, before reaching a steady state in which it appears as a disk of radius 5, approximately. This observation is useful in the following when comparing the particles arrangement with that of the nonlinear regimes. In Figure 29 and Figure 32, all the methods show a clear agreement and converge towards the expected steady state solution. This is supported by Figure 30 and Figure 33, for which the CEM method performs particularly well, as their error is ever-decreasing during the time horizon. The FPK-SPH and MCS methods are satisfactory too, as their steady state solutions correspond to $\mathcal{E} \approx 1$ [%] for both position and velocity. Regarding the confidence plots of Figure 31 and Figure 34, they indicate that the FPK-SPH is likely close to the exact solution, throughout the whole time horizon.

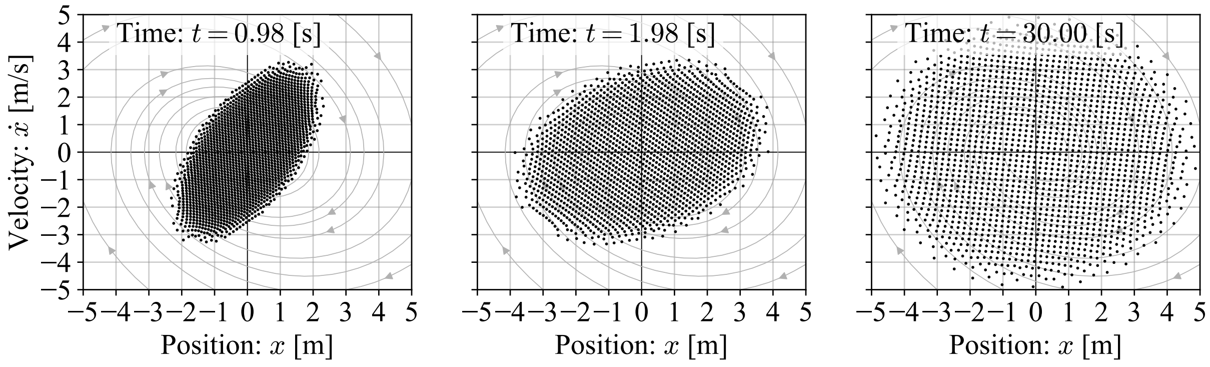


Figure 28: Snapshots of the SPH particles arrangement for the Duffing unimodal linear regime at the time references $t \approx 1$ [s] (left), $t \approx 2$ [s] (middle) and $t \approx 30$ [s] (right). Streamlines of the drift vector field are sketched in gray.

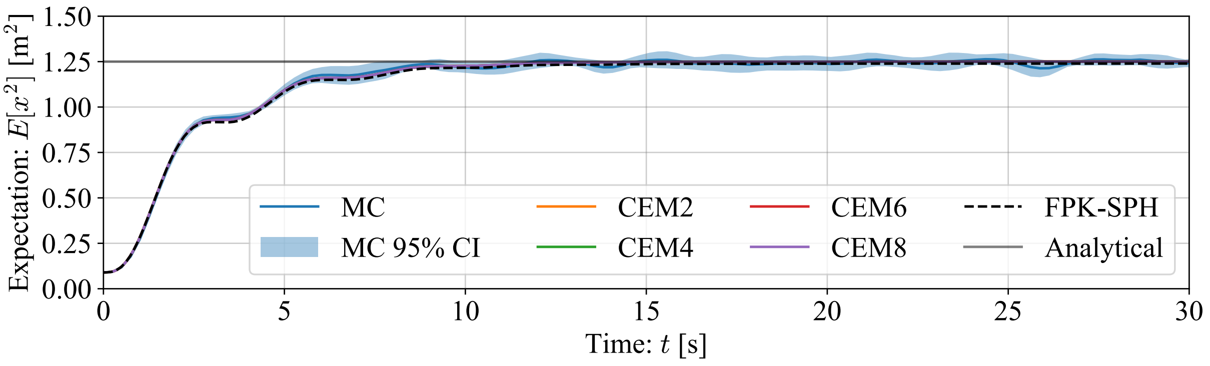


Figure 29: Evolution of the variance of position for the Duffing unimodal linear regime, computed with different solvers. The analytical steady state value is 1.25 [m^2].

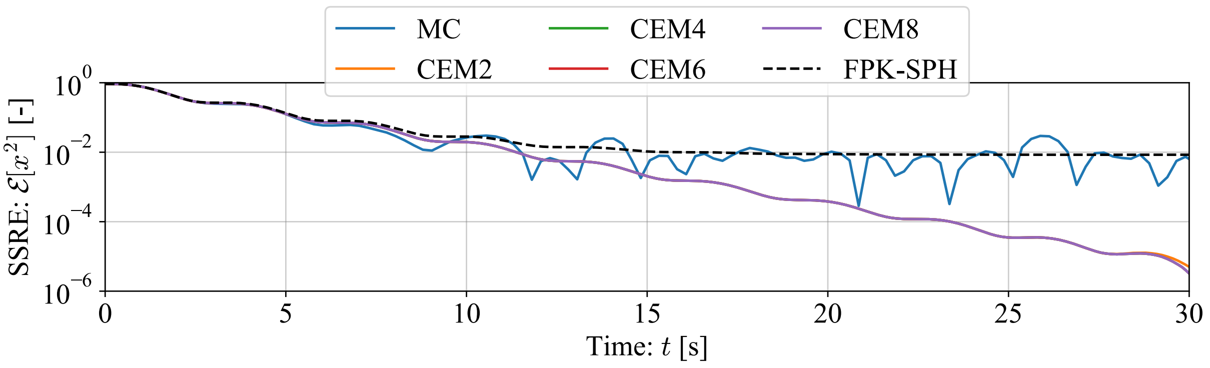


Figure 30: Evolution of the (absolute) SSRE for the variance of position of the Duffing unimodal linear regime, considering several solvers.

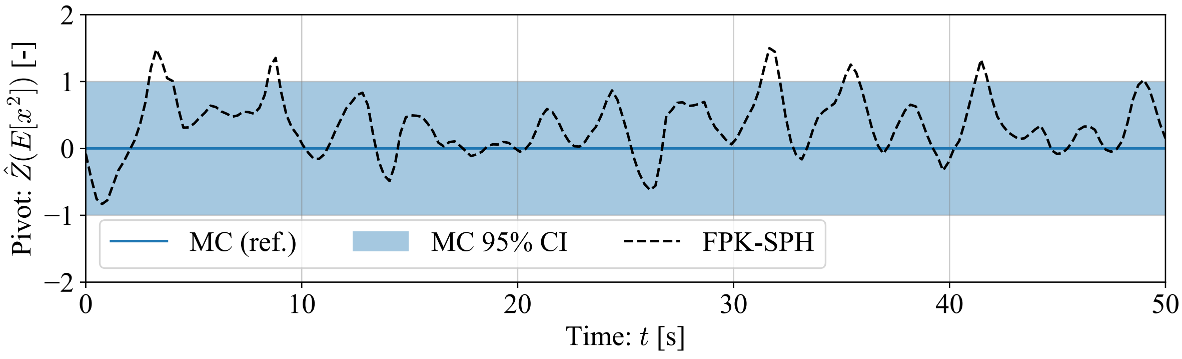


Figure 31: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of position of the Duffing unimodal linear regime.

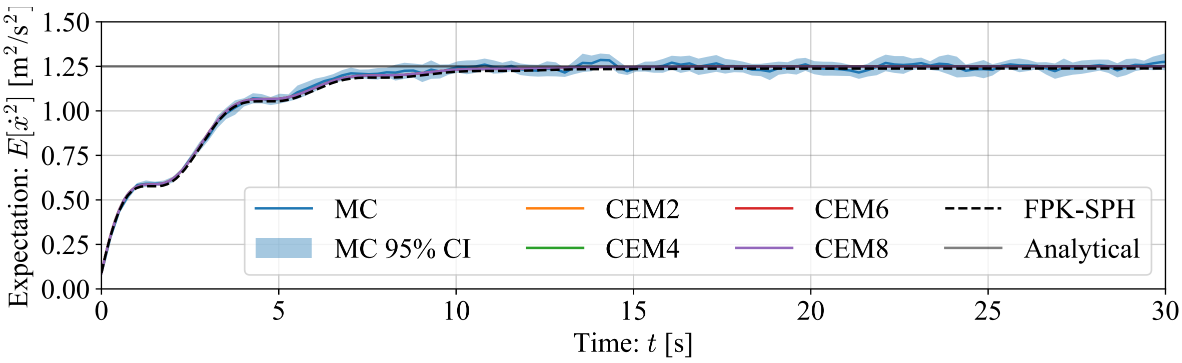


Figure 32: Evolution of the variance of velocity for the Duffing unimodal linear regime, computed with different solvers. The analytical steady state value is 1.25 [$\mathrm{m}^2/\mathrm{s}^2$].

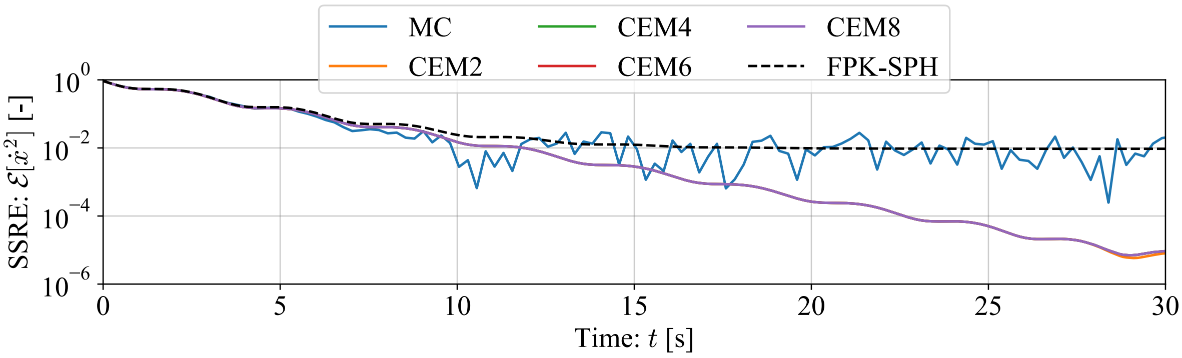


Figure 33: Evolution of the (absolute) SSRE for the variance of velocity of the Duffing unimodal linear regime, considering several solvers.

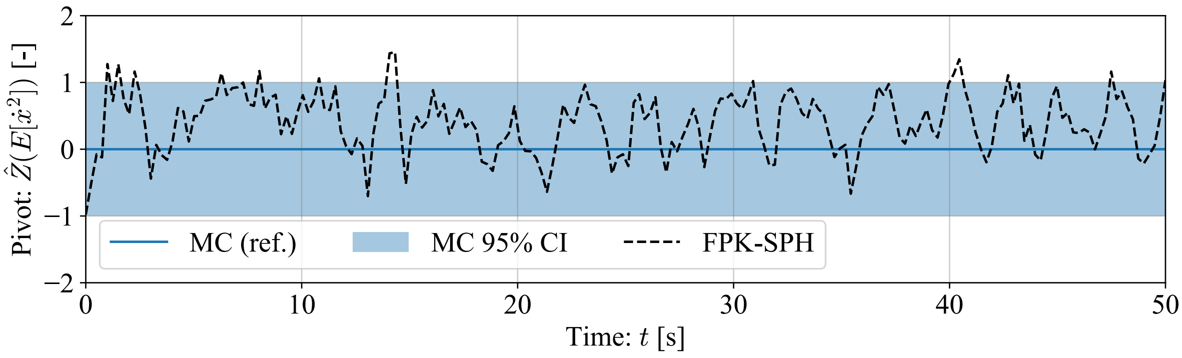


Figure 34: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of velocity of the Duffing unimodal linear regime.

The results corresponding to the *weakly nonlinear* scenario are represented in Figure 35 to Figure 41. As illustrated in Figure 35, the trajectory that the SPH particles follow is approximately the same than for the linear case. However, some dissimilarities can be observed; in particular, the steady state arrangement of particles now corresponds to an ellipse, shrunk in the x-direction. As a consequence, the CEM seems to suffer from this change in the system's dynamics, especially for its position estimation (see Figure 36 and Figure 37). Additionally, the quality of the FPK-SPH results now sits between that of the CEM of order 2 and order 4. Besides, the weakly nonlinear scenario is very similar to the linear case in terms of results and the FPK-SPH results keep the SSRE inferior to 1 [%] for both position and velocity.

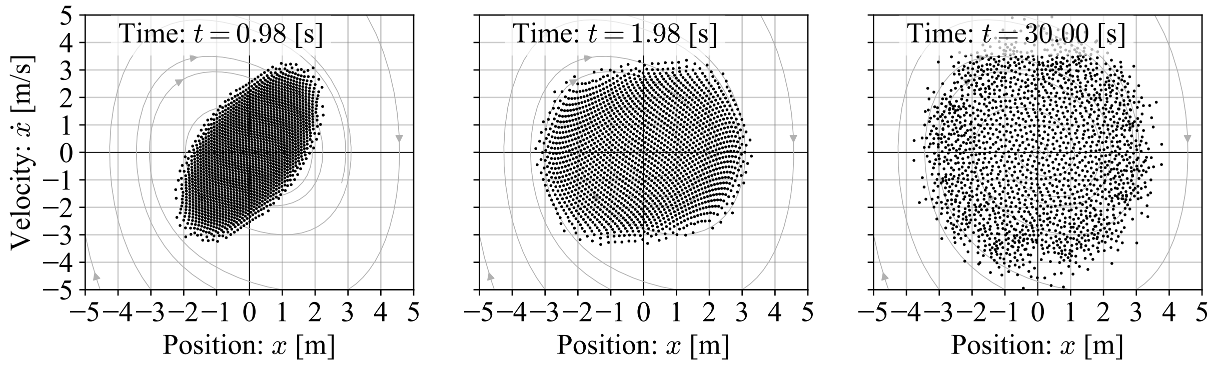


Figure 35: Snapshots of the SPH particles arrangement for the Duffing unimodal weakly nonlinear regime at the time references $t \approx 1$ [s] (left), $t \approx 2$ [s] (middle) and $t \approx 30$ [s] (right). Streamlines of the drift vector field are sketched in gray.

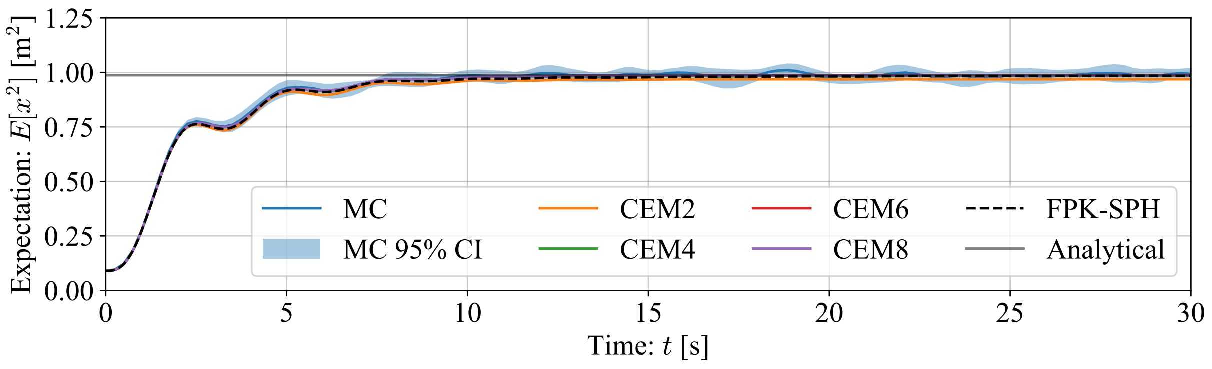


Figure 36: Evolution of the variance of position for the Duffing unimodal weakly nonlinear regime, computed with different solvers. The analytical steady state value is ~ 0.987 [m^2].

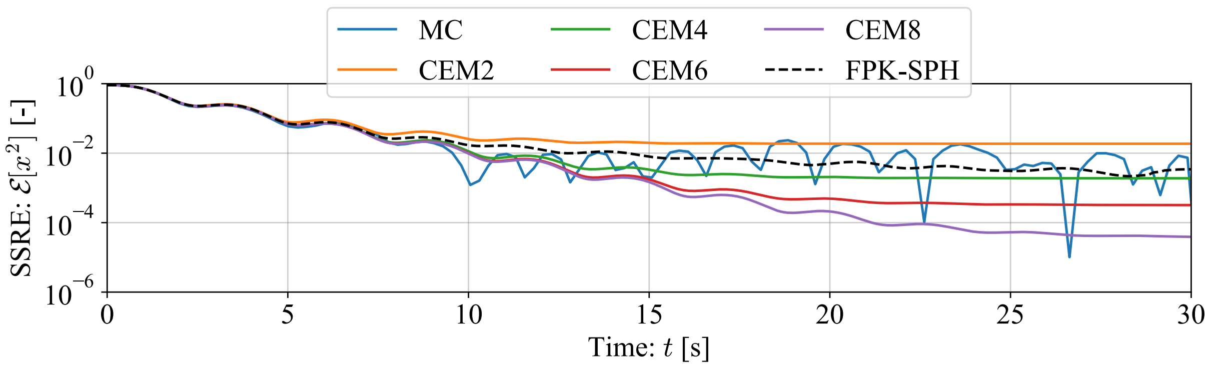


Figure 37: Evolution of the (absolute) SSRE for the variance of position of the Duffing unimodal weakly nonlinear regime, considering several solvers.

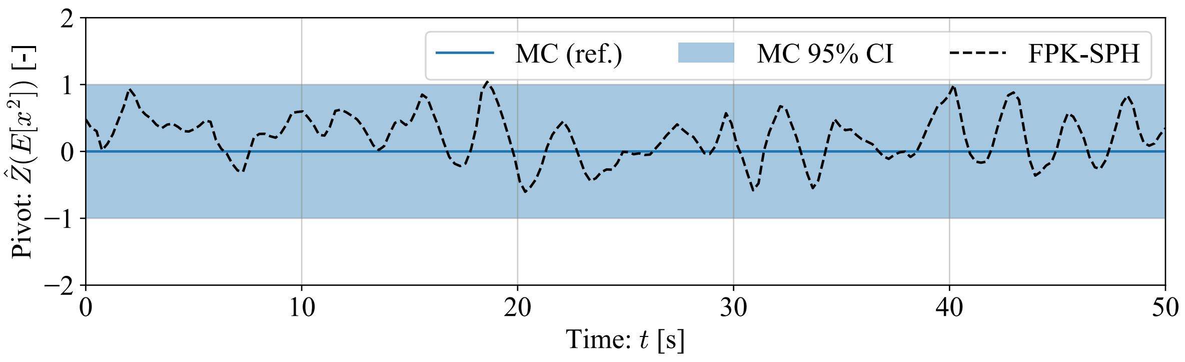


Figure 38: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of position of the Duffing unimodal weakly nonlinear regime.

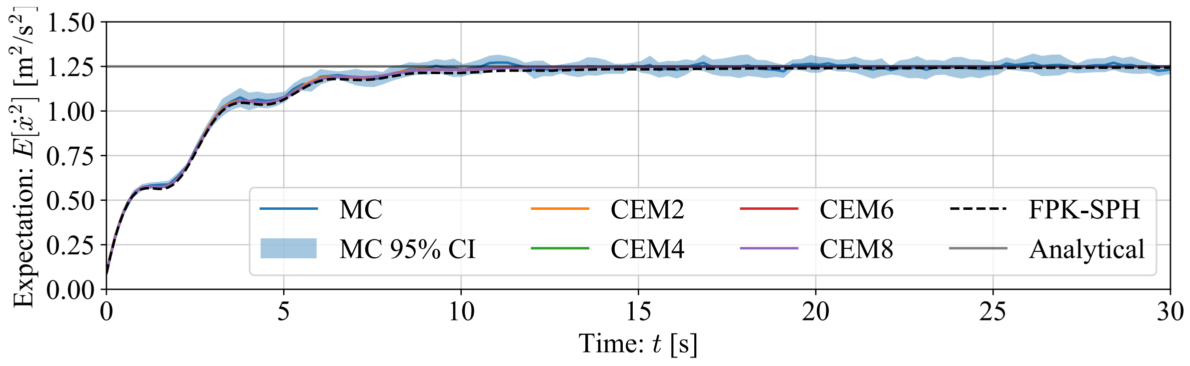


Figure 39: Evolution of the variance of velocity for the Duffing unimodal weakly nonlinear regime, computed with different solvers. The analytical steady state value is 1.25 [m^2/s^2].

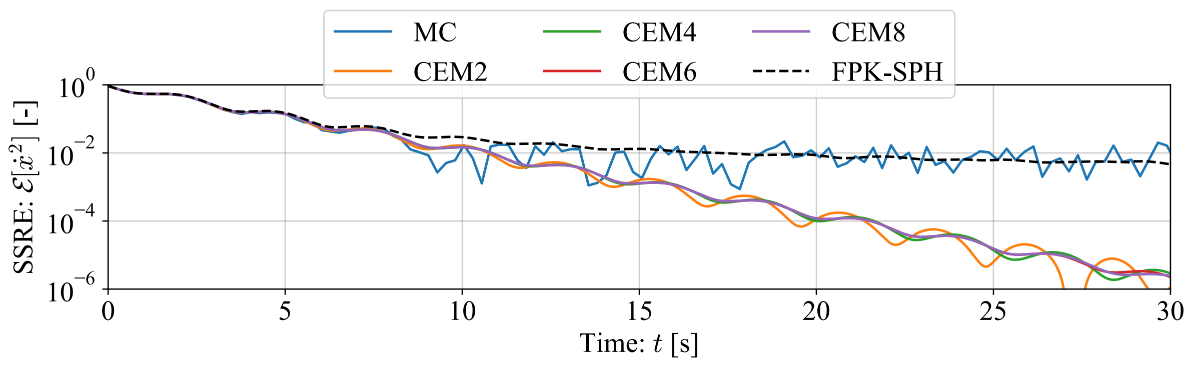


Figure 40: Evolution of the (absolute) SSRE for the variance of velocity of the Duffing unimodal weakly nonlinear regime, considering several solvers.

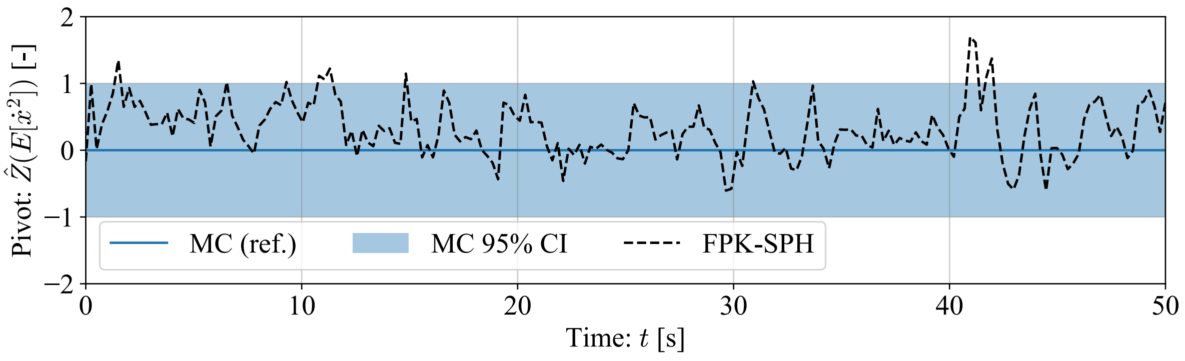


Figure 41: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of velocity of the Duffing unimodal weakly nonlinear regime.

The results corresponding to the *fully nonlinear* scenario are represented in Figure 42 to Figure 48. Directly, one can notice that the SPH particles distributions in Figure 42 are relatively different than that of the linear case. Indeed, the initial disk of particles is clearly subject to nonlinear deformations which eventually lead to an even more important shrink of the distribution in the x-direction, while keeping its extent nearly untouched in the $\dot{x}$-direction. As for the weakly nonlinear regime, the CEM results are negatively affected, as it is shown in Figure 43 and Figure 44. This time, the FPK-SPH method gives the best results while still keeping $\mathcal{E} \approx 1$ [%] for both position and velocity. The confidence plots in Figure 45 and Figure 48 indicate that the FPK-SPH solution is still likely close to the exact solution, but the fidelity has decreased in comparison with the linear scenario.

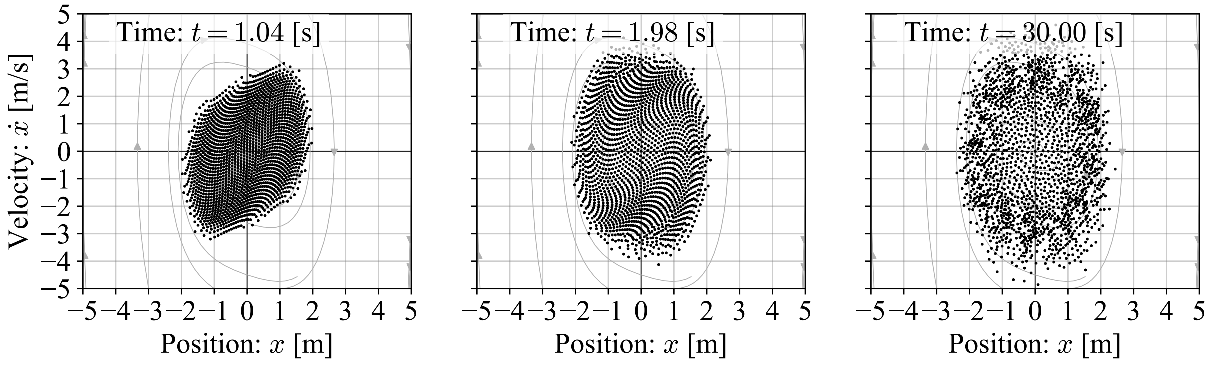


Figure 42: Snapshots of the SPH particles arrangement for the Duffing unimodal fully nonlinear regime at the time references $t \approx 1$ [s] (left), $t \approx 2$ [s] (middle) and $t \approx 30$ [s] (right). Streamlines of the drift vector field are sketched in gray.

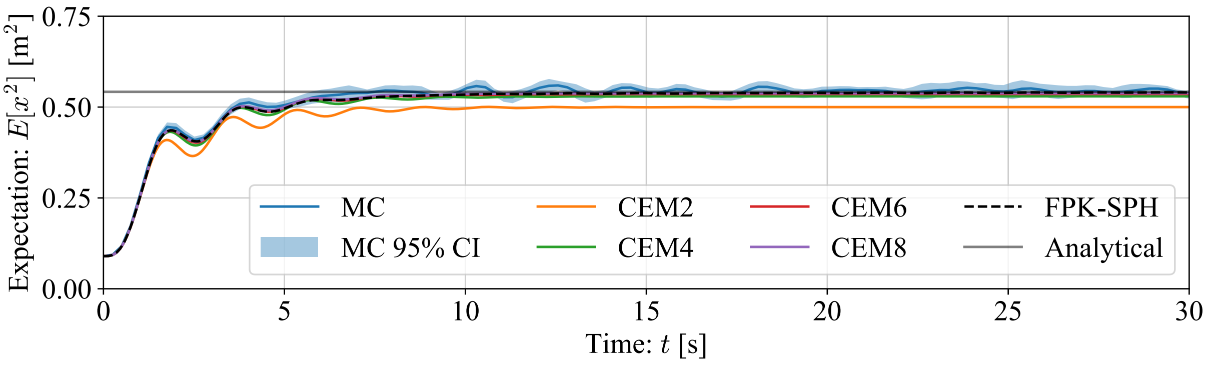


Figure 43: Evolution of the variance of position for the Duffing unimodal fully nonlinear regime, computed with different solvers. The analytical steady state value is ~ 0.542 [m^2].

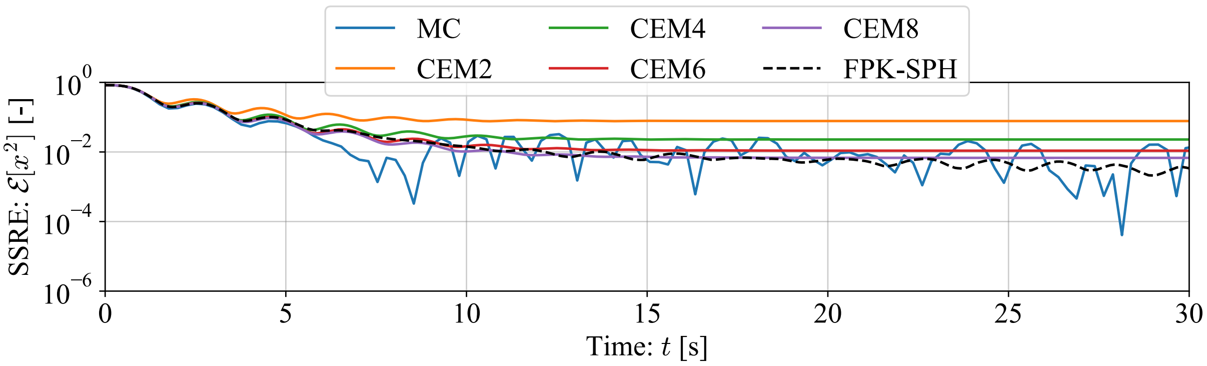


Figure 44: Evolution of the (absolute) SSRE for the variance of position of the Duffing unimodal fully nonlinear regime, considering several solvers.

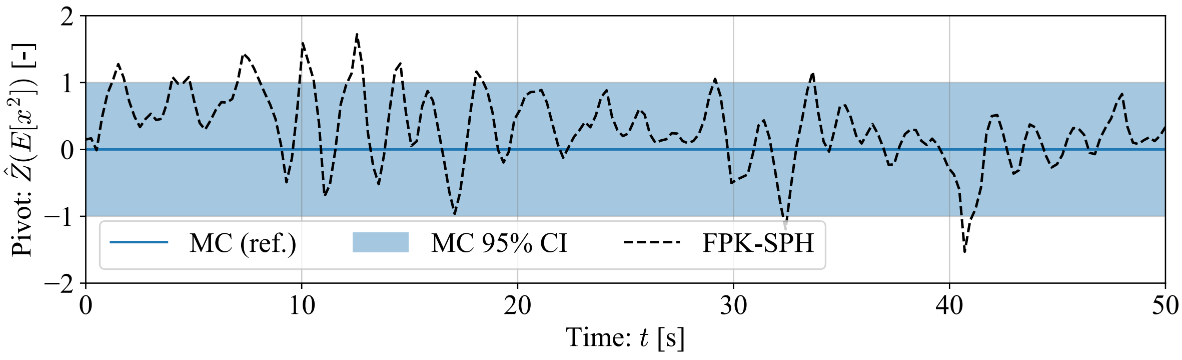


Figure 45: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of position of the Duffing unimodal fully nonlinear regime.

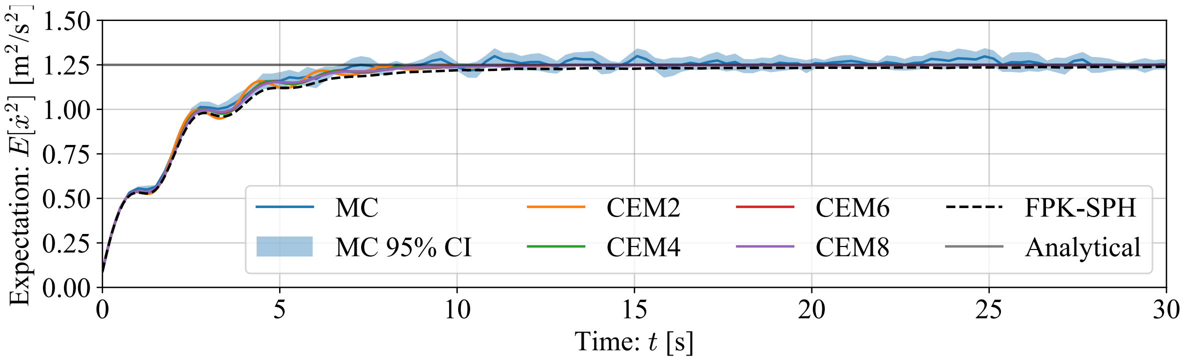


Figure 46: Evolution of the variance of velocity of the Duffing unimodal fully nonlinear regime, considering several solvers. The analytical steady state value is 1.25 [m^2/s^2].

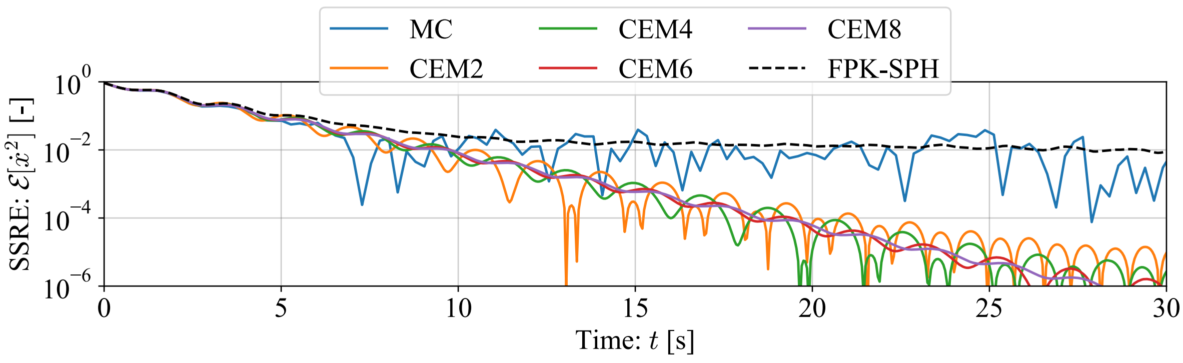


Figure 47: Evolution of the (absolute) SSRE for the variance of velocity of the Duffing unimodal fully nonlinear regime, considering several solvers.

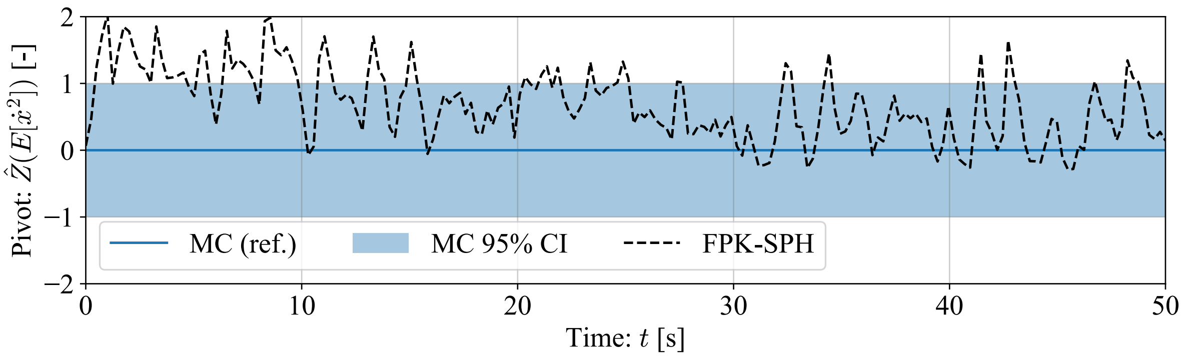


Figure 48: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of velocity of the Duffing unimodal fully nonlinear regime.

4.2.7 Bimodal regime

In this second part, the physical Duffing oscillator model is extended to a model in which k_0 and b are negative, which results in a bimodal steady state distribution. In particular, two scenarios are presented in Table 3 and solved in this section.

Scenario	$\boldsymbol{c}$ **[Ns/m]**	$\boldsymbol{k_0}$ **[N/m]**	$\boldsymbol{b}$ **[-]**
Weakly nonlinear	0.4	−1.0	−0.1
Fully nonlinear	0.4	−1.0	−1.0

Table 3: Parameter values used in each Duffing bimodal scenario.

The steady state PDF is bimodal (i.e., it has two peaks) in the x-direction for both scenarios, as illustrated in Figure 49. As the results show in the following, the transition from a unimodal to a bimodal distribution heavily impacts the CEM; this is discussed in Section 4.2.8. Noticeably, the CEM of order 8 is not used for the bimodal study as it fails to solve the problem in a reasonable amount of time.

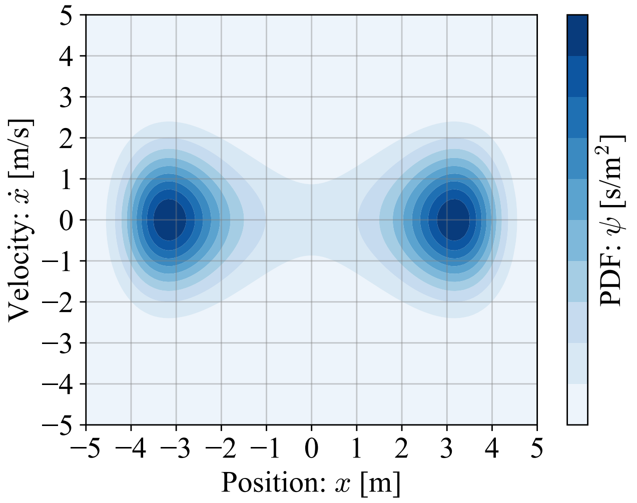


Figure 49: Contours of the steady state PDF of the bimodal Duffing oscillator, here obtained for the weakly nonlinear scenario ($b = -0.1$ [-]).

The results of the *bimodal weakly nonlinear* test case are presented in Figure 50 to Figure 57. As one can tell by looking at Figure 50 and Figure 51, the SPH particles adopt a totally different trajectory than in the unimodal case.

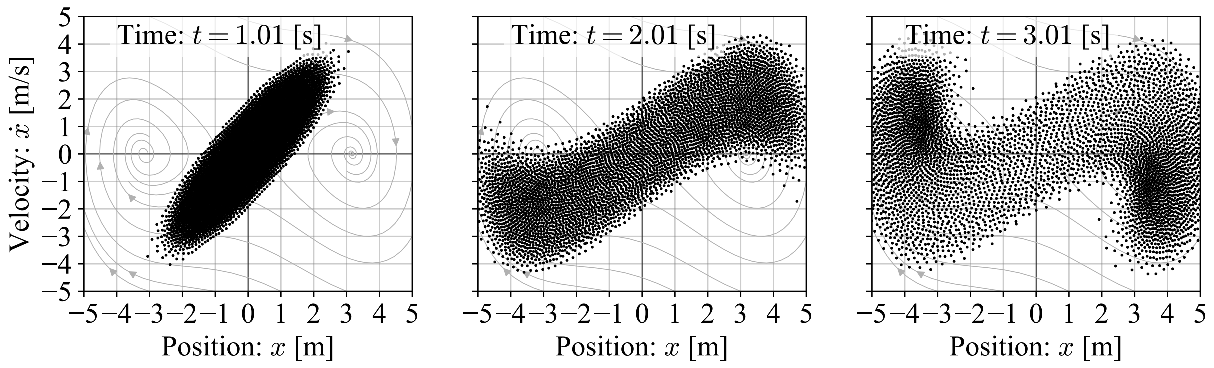


Figure 50: Snapshots of the SPH particles arrangement for the Duffing bimodal weakly nonlinear regime at the time references $t \approx 1$ [s] (left), $t \approx 2$ [s] (middle) and $t \approx 3$ [s] (right). Streamlines of the drift vector field are sketched in gray.

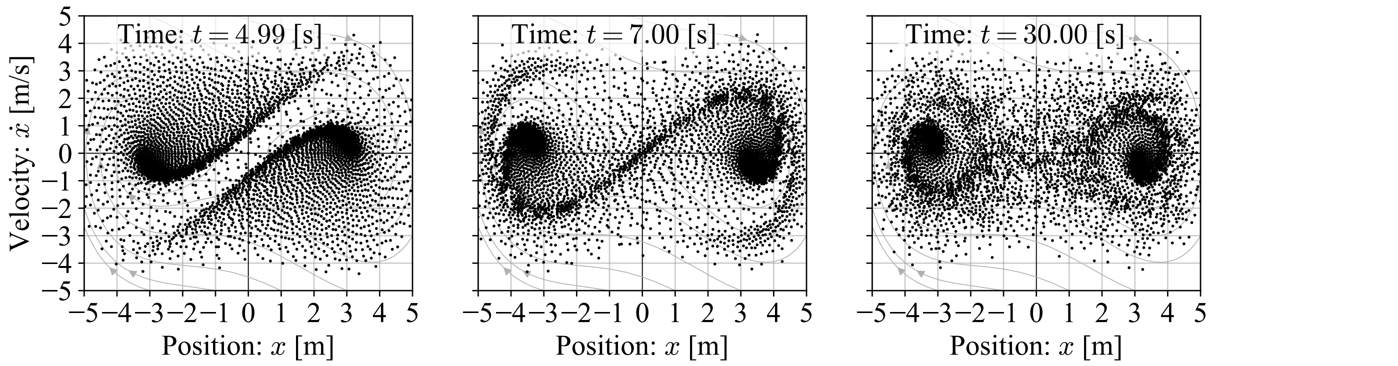


Figure 51: Snapshots of the SPH particles arrangement for the Duffing bimodal weakly nonlinear regime at the time references $t \approx 5$ [s] (left), $t \approx 7$ [s] (middle) and $t \approx 30$ [s] (right). Streamlines of the drift vector field are sketched in gray.

This time, the particles concentrate at and rotate around two distinct points, corresponding to the two modes of ψ_∞. Besides, it is worth noting that these two modes correspond to the two equilibrium points when the Duffing system is subject to no external excitation.

For the position state, Figure 52 shows that the CEM converges to a wrong steady state value for $E[x^2]$. On the other hand, the FPK-SPH method seems to agree with the MCS during the total time horizon (see Figure 54) and converge towards the expected steady state value, with again an SSRE of the order of 1 [%] (see Figure 53). Regarding the variance of velocity, the CEM converges towards the steady state value with a better rate than that of the FPK-SPH and MCS methods, as illustrated by Figure 56. Indeed, the CEM's steady state error seems ever-decreasing within the time horizon while the FPK-SPH and MCS results still seem to saturate at $\mathcal{E}[\dot{x}^2] \approx 1$ [%]. In the transitory phase of the system, Figure 55 shows that the CEM captures a wrong transitory solution, whereas the FPK-SPH and MCS methods agree to a similar evolution of the variance of velocity. The latter is true for the entire time horizon, as indicated by Figure 57.

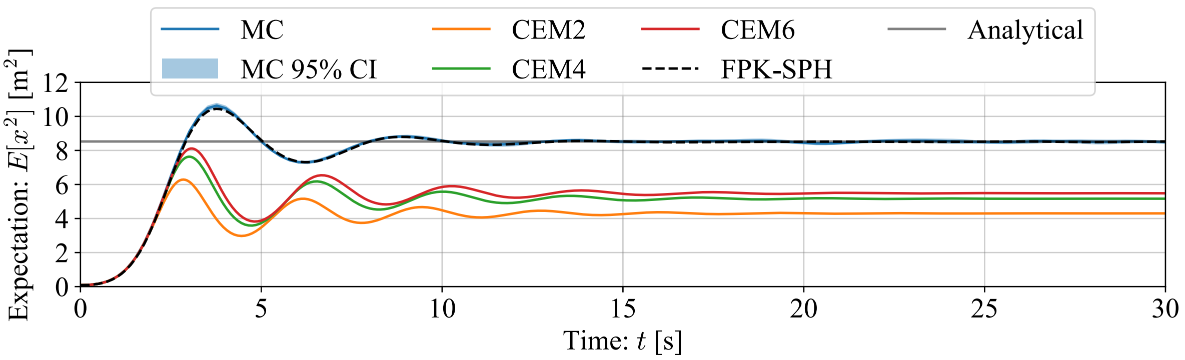


Figure 52: Evolution of the variance of position for the Duffing bimodal weakly nonlinear regime, computed with different solvers. The analytical steady state value is ~ 8.526 [m^2].

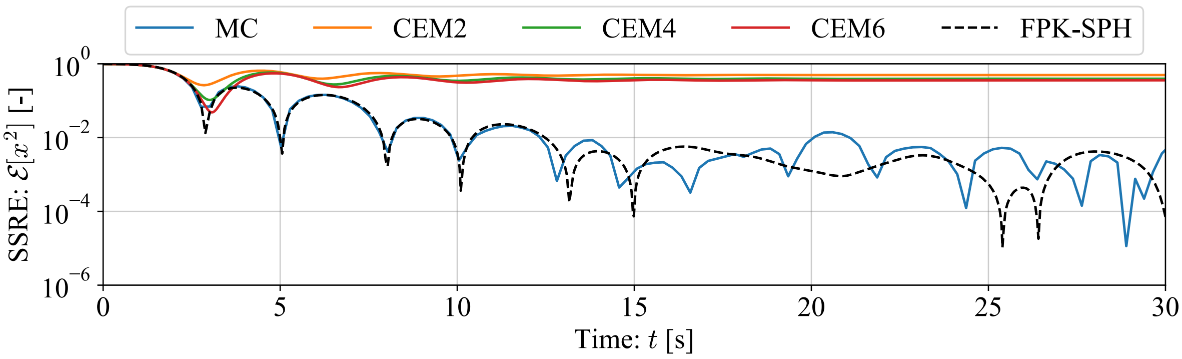


Figure 53: Evolution of the (absolute) SSRE for the variance of position of the Duffing bimodal weakly nonlinear regime, considering several solvers.

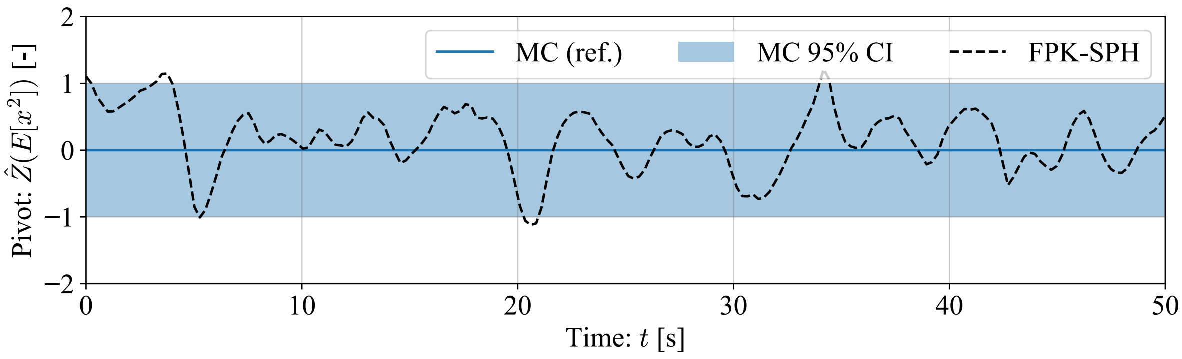


Figure 54: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of position of the Duffing bimodal weakly nonlinear regime.

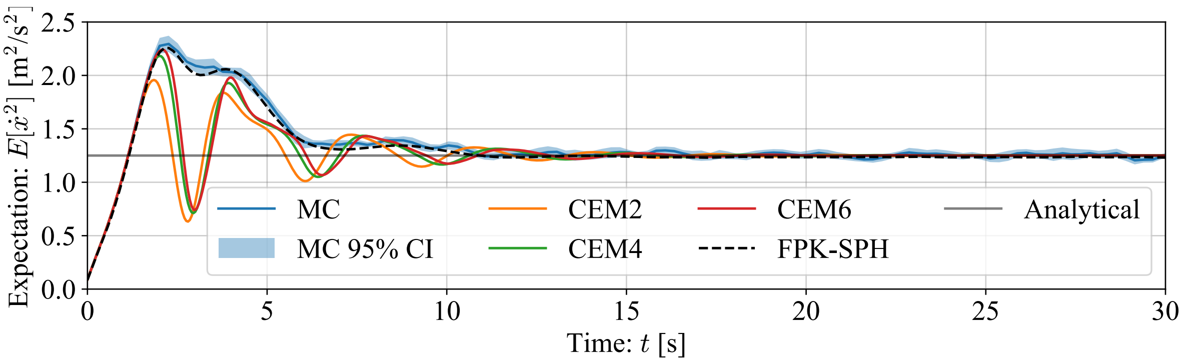


Figure 55: Evolution of the variance of velocity of the Duffing bimodal weakly nonlinear regime, considering several solvers. The analytical steady state value is 1.25 $[\mathrm{m}^2/\mathrm{s}^2]$.

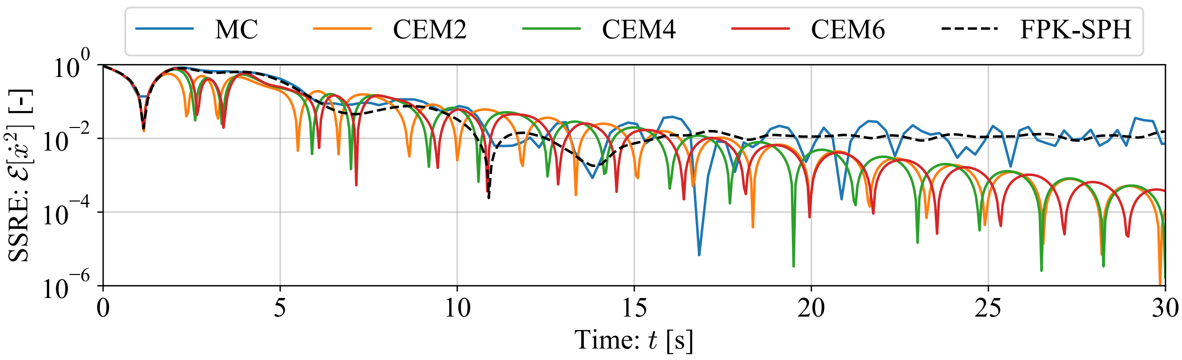


Figure 56: Evolution of the (absolute) SSRE for the variance of velocity of the Duffing bimodal weakly nonlinear regime, considering several solvers.

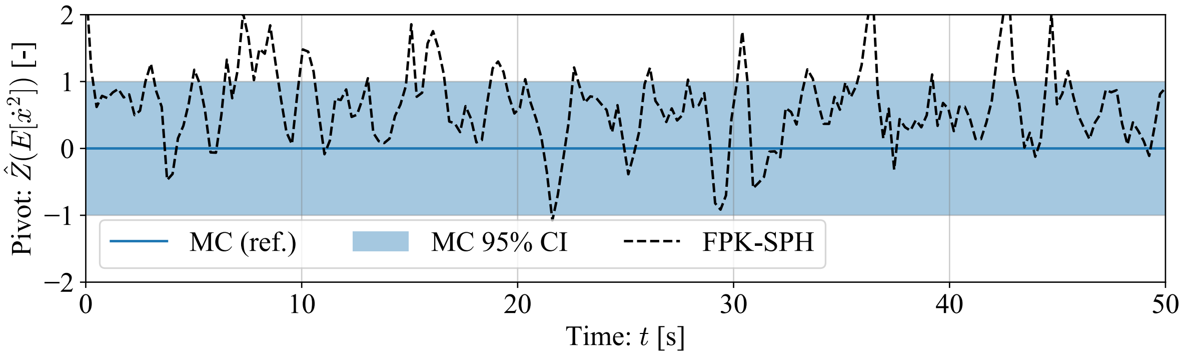


Figure 57: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of velocity of the Duffing bimodal weakly nonlinear regime.

The same analysis is finally performed on the *bimodal fully nonlinear* test case, as presented in Figure 58 to Figure 65. As for the arrangement of particles represented in Figure 58 and Figure 59, it is relatively similar to that of the bimodal weakly nonlinear test case. The SPH swarm of particles wraps around itself and eventually settles into a relatively compact arrangement.

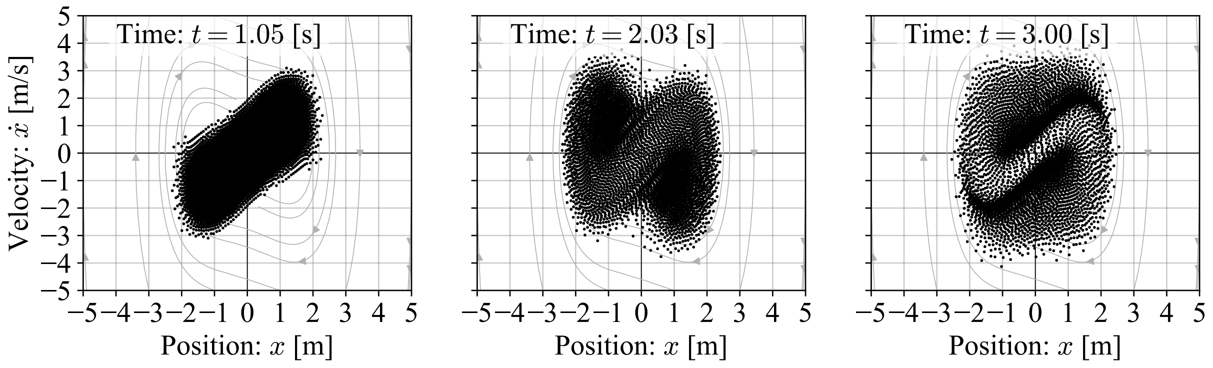


Figure 58: Snapshots of the SPH particles arrangement for the Duffing bimodal fully nonlinear regime at the time references $t \approx 1$ [s] (left), $t \approx 2$ [s] (middle) and $t \approx 3$ [s] (right). Streamlines of the drift vector field are sketched in gray.

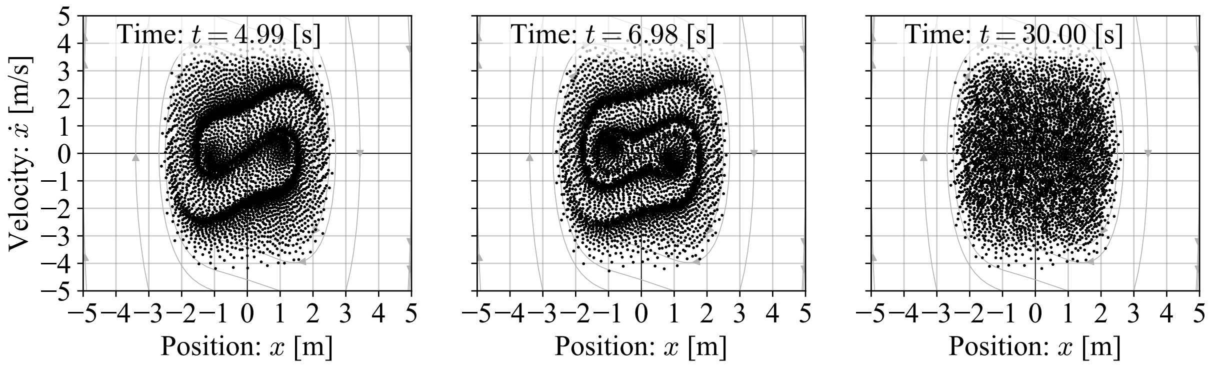


Figure 59: Snapshots of the SPH particles arrangement for the Duffing bimodal fully nonlinear regime at the time references $t \approx 5$ [s] (left), $t \approx 7$ [s] (middle) and $t \approx 30$ [s] (right). Streamlines of the drift vector field are sketched in gray.

For the sake of conciseness, let us point out that the observations that can be made for this scenario are exactly the same as for the previous one.

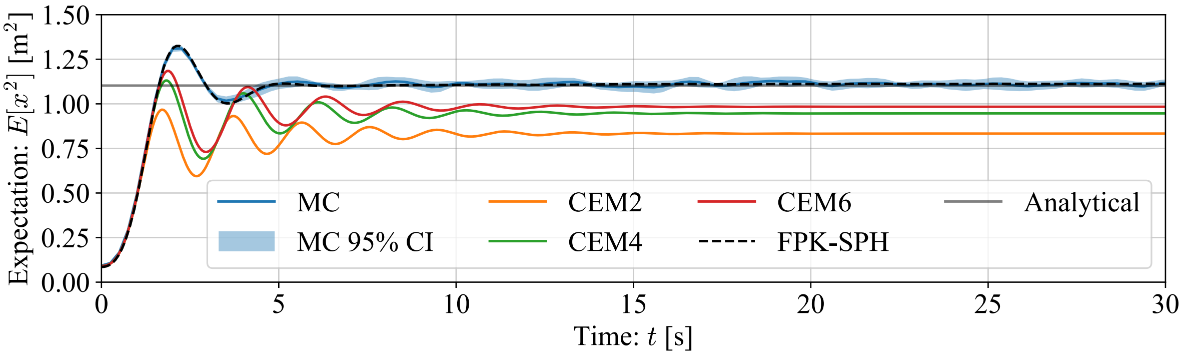


Figure 60: Evolution of the variance of position for the Duffing bimodal weakly nonlinear regime, computed with different solvers. The analytical steady state value is ~ 1.1025 [m^2].

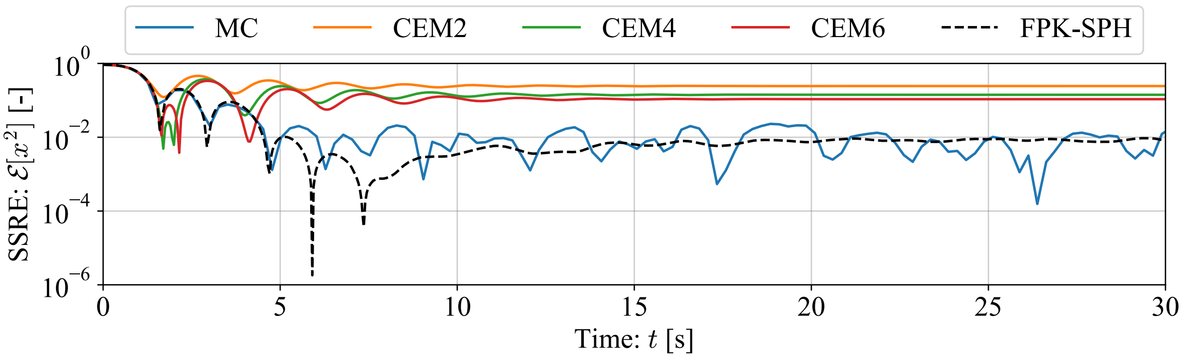


Figure 61: Evolution of the (absolute) SSRE for the variance of position of the Duffing bimodal fully nonlinear regime, considering several solvers.

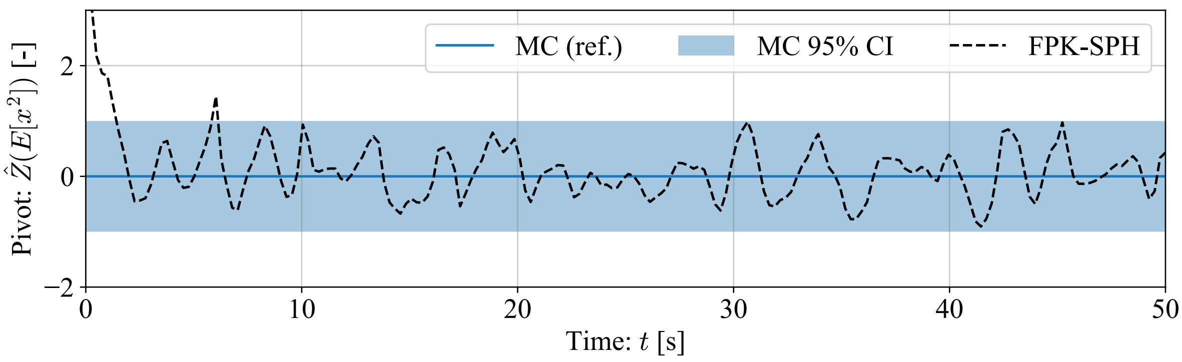


Figure 62: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of position of the Duffing bimodal fully nonlinear regime.

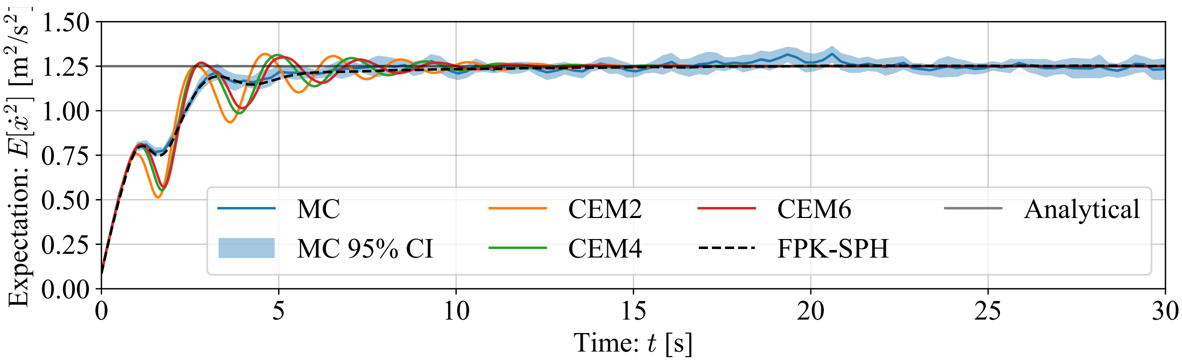


Figure 63: Evolution of the variance of velocity of the Duffing bimodal fully nonlinear regime, considering several solvers. The analytical steady state value is 1.25 [m^2/s^2].

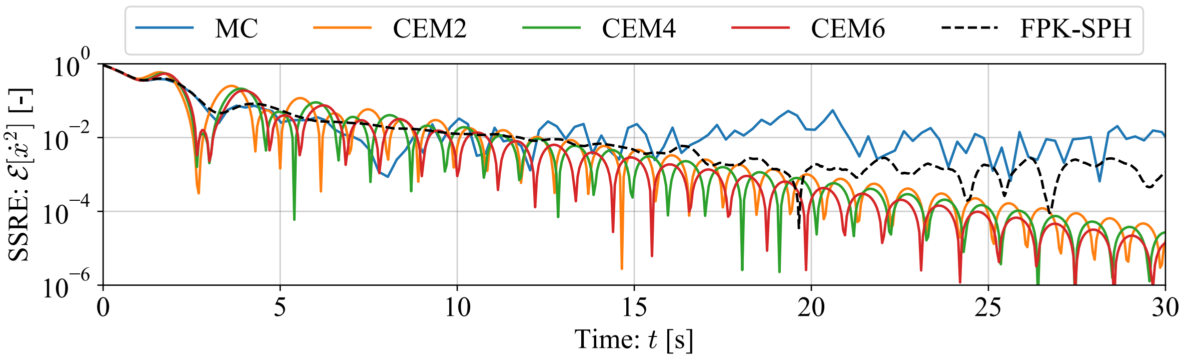


Figure 64: Evolution of the (absolute) SSRE for the variance of velocity of the Duffing bimodal fully nonlinear regime, considering several solvers.

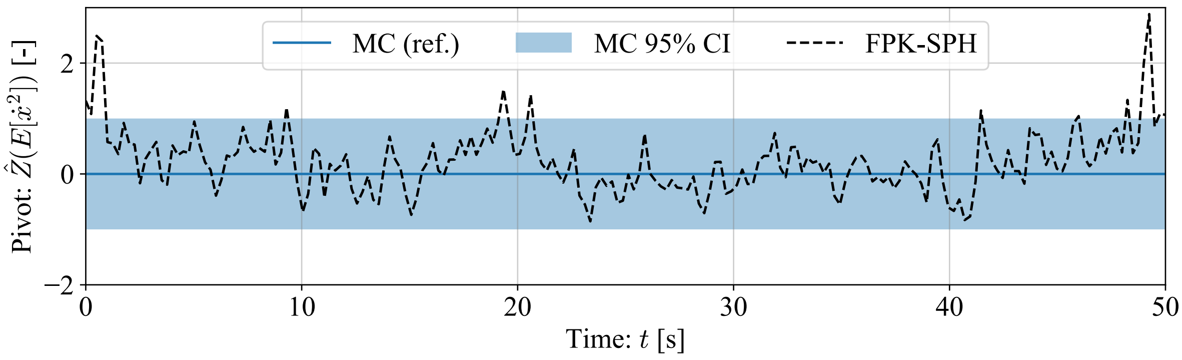


Figure 65: Confidence plot of the FPK-SPH method with respect to its MCS counterpart, for the variance of velocity of the Duffing bimodal fully nonlinear regime.

4.2.8 Discussion and runtime comparison

In this section, we aim at presenting a comprehensive comparison of the performance of the FPK-SPH solver and the CEM solver [Lei+25], both in terms of precision and computation time. In that scope, the computation time of the solvers for the different scenarios of Table 2 and Table 3 are summarized in Table 4. Here, a direct runtime comparison with the MCS method would be unfair since not much effort was invested in its implementation. For information purpose only, let us note that the MCS method is here implemented in Python, using a single CPU core, and runs in about 140 [s] for each Duffing scenario.

		FPK-SPH [s]		**CEM** [s]			
	Scenario	**absolute**	**relative**	**order 2**	**order 4**	**order 6**	**order 8**
Unimodal	Linear	276	47	0.005	0.016	0.063	0.707
	Weakly NL	6577	56	0.006	0.021	0.118	0.870
	Fully NL	> 10000	325	0.01	0.036	0.137	1.801
Bimodal	Weakly NL	> 10000	689	0.011	0.059	0.593	diverged
	Fully NL	> 10000	900	0.012	0.053	0.448	diverged

Table 4: Computation time of the Duffing unimodal test case for several solvers and settings. This includes the FPK-SPH method with a stability coefficient $S = 0.1$ [-] and either an absolute or a relative time stepping. The computation time of the CEM is also given for different orders of accuracy. The results in this table are given for $T = 50$ [s] instead of $T = 30$ [s].

Before comparing the CEM and FPK-SPH performances, let us discuss the use of the absolute and relative conventions for the adaptive time stepping of the FPK-SPH method. As the figures show, the SPH particles revolve around one or two points in state space. Therefore, the outermost particles have a large linear velocity in state space and the absolute adaptive time stepping will be too restrictive regarding the time step. For this reason, the simulation runs in an acceptable amount of time when the relative convention adaptive time stepping is used, for the chosen stability factor $S = 0.1$ [-]. Of course, the absolute adaptive time stepping could work correctly by relaxing the stability coefficient. However, in such case, using the relative adaptive time stepping in the first place makes more sense as there is no need for fitting the stability coefficient to a potentially unphysical value. Overall, it is important to bear in mind that if the relative adaptive time stepping makes more sense for the present application, it is not a generality: the best time stepping convention is problem-dependent.

For the unimodal regime of the Duffing oscillator, the CEM performs very well in terms of precision and time cost. It loses a little accuracy as nonlinearity is introduced in the system, but this is easily remedied by increasing the order of the method: in the worst case (i.e., the fully nonlinear case), the 8^{th} order guarantees an SSRE of the order of 1 [%] for both position and velocity, while keeping the computation time under 2 [s]. The efficiency of the method for the unimodal case is supported by the fact that ψ_∞ is close to a Gaussian distribution and can thus be effectively approximated with a limited number of cumulants.

Then, when the bimodal regime is considered, the CEM is impacted negatively, both in terms of runtime and accuracy. Even though being increased, the runtime remains relatively small with respect to that of the FPK-SPH, with the exception of the 8^{th} order CEM which does not converge in a reasonable amount of time. The accuracy of the method is heavily impacted by the distribution having two modes. In particular, the transient part of the evolution of the variances of position and velocity is incorrect. Additionally, the method cannot converge towards the expected steady state value for the variance of position. This can be explained by the fact that ψ_∞ is marginally bimodal for x. Consequently, the cumulant generating function $K_x(t) = \log(E[\exp(itx)])$ is periodic, which implies that an infinite number of cumulants are necessary for representing ψ_∞ exactly. Using at most 6 cumulants thus offers a poor representation of ψ_∞ and its variance in the x-direction of the state space.

Unlike the CEM, the FPK-SPH method seems to be relatively insensitive to the change of regime for the Duffing model. In fact, its SSRE stays of the order of 1 [%] regardless of the chosen scenario. This is because the FPK-SPH method relies on a local approximation of the PDF itself, instead of global quantities like the cumulants. This comes at a computational cost in terms of time and numerical resources spent for the simulation. Concerning runtime, Table 4 clearly shows that the FPK-SPH method runs much slowly than the CEM. The computational resources are also different. On the one hand, the FPK-SPH solver is a compiled C++ program which relies on a large number of particles ($N = 1789$ for unimodal and $N = 4049$ for bimodal) and benefits from multi-core acceleration. On the other hand, the CEM is a Python script, which is interpreted at runtime and therefore less efficient.

Overall, both methods are promising. The CEM is well-suited for problems in which the PDF can be described with a limited number of cumulants. In such a case, the CEM yields results of great accuracy while keeping the computation time very low. By opposition, the FPK-SPH method comes with a heavier computational burden, but it offers a better fidelity for more complex dynamics.

4.3 Lorenz system

4.3.1 Problem definition

The Lorenz system is a well-known system of three nonlinear ordinary differential equations, originally derived from a simplified model of convection in the atmosphere. It is defined as

$$\begin{cases} \dot{x}(t) = \sigma(y(t) - x(t)) \\ \dot{y}(t) = \rho x(t) - x(t)z(t) - y(t) \\ \dot{z}(t) = x(t)y(t) - \beta z(t). \end{cases} \tag{162}$$

Under certain conditions, the system exhibits peculiar behaviors. In particular, the parameters defining the Lorenz system are set to $\sigma = 10$ [-], $\rho = 28$ [-], and $\beta = 8/3$ [-] for this entire test case. This combination of values causes the system to act as a strange attractor in the state space. The system is chaotic, meaning that small changes in the initial conditions can lead to vastly different trajectories over time. In Figure 66, the trajectory of a single realization is shown.

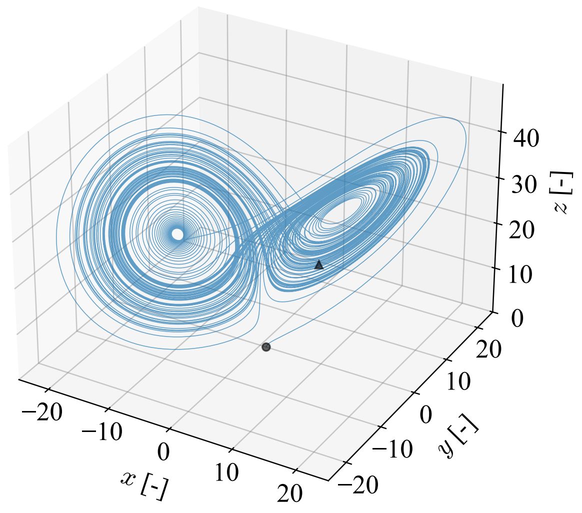


Figure 66: Single trajectory within the Lorenz strange attractor ($\sigma = 10$, $\rho = 28$, $\beta = 8/3$). The initial condition (black dot) is $(x, y, z) = (1, 0, 0)$ and the solution evolves to a final state (black triangle) after $T = 100$ [-].

As such, the system is deterministic albeit chaotic. Adding an independent DGWN to each equation formally leads to

$$\begin{cases} \dot{x}(t) = \sigma(y(t) - x(t)) & + w_x(t) \\ \dot{y}(t) = \rho x(t) - x(t)z(t) - y(t) & + w_y(t) \\ \dot{z}(t) = x(t)y(t) - \beta z(t) & + w_z(t). \end{cases} \tag{163}$$

where w_x, w_y and w_z are independent DGWNs. Writing this formal equation as a more rigorous Itō SDE, one gets

$$\mathrm{d}\underline{X} = \mathrm{d}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} \sigma(y - x) \\ \rho x - xz - y \\ xy - \beta z \end{pmatrix}}_{\underline{f}(\underline{X},t)} \mathrm{d}t + \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\underline{\underline{g}}(\underline{X},t)} \mathrm{d}\underline{W} \tag{164}$$

and can readily identify the drift vector function $\underline{f}(\underline{X}, t)$ and the noise amplification matrix function $\underline{\underline{g}}(\underline{X}, t)$. The trajectory obtained by stochastic integration is almost surely unrelated to the deterministic one and the final state is completely different, as shown in Figure 67.

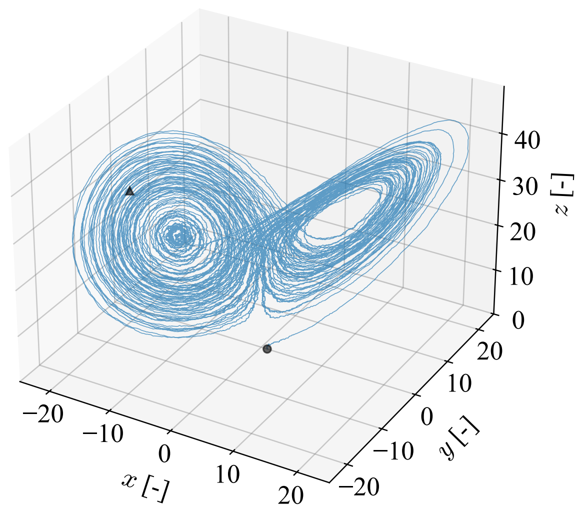


Figure 67: Single realization within the Lorenz strange attractor ($\sigma = 10$, $\rho = 28$, $\beta = 8/3$) under the DGWN $\underline{w}$. The initial condition (black spot) is $(x, y, z) = (1, 0, 0)$ and the solution evolves to a final state (black triangle) after $T = 100$ [-].

One proposes to study the evolution of the PDF of the system thanks to the SPH solver for the FPK equation. The application is chosen for benchmarking the method for several reasons. First, the system is well-studied and exhibits a nonlinear behavior. Second, the fidelity of the method can be assessed in a chaotic regime, where the PDF is expected to be very sensitive to the initial conditions, and to noise for that matter. Finally, the system is three-dimensional, which allows to test whether the method generalizes well to higher dimensions than the two-dimensional problems considered so far.

4.3.2 FPK-SPH statement

Under the FPK-SPH formalism, Equation 164 is expressed by an equivalent FPK equation (see Equation 39), which is then solved by applying the SPH method with the following configuration.

- Initial condition: one considers an initial multivariate normal distribution with $\underline{0}$ mean and an identity covariance, expressed by

$$\psi(\underline{X}, 0) = C \exp\left(-\frac{x^2 + y^2 + z^2}{2}\right), \tag{165}$$

  where C is a normalization factor. A cubic kernel is used.

- Initial arrangement of particles: the $[-4.0, 4.0]^3$ interval of state space is covered with a regular lattice of $28^3 = 21952$ particles. Amongst these particles, only those which lie within 4 times the standard deviation are kept, as the mass carried by the rest is negligible. This results in $N = 10144$ particles distributed in the 3-ball $\Omega = \{\underline{X} : \|\underline{X}\| < 4\}$, as illustrated in Figure 68. Since the initial condition described in the previous point relies on an infinite support, only a truncated version of the initial condition is represented, which implies a normalization of the total probability held by the particles.

- Time integration: for this study, a time horizon of $T = 3$ [s] is simulated, using a fixed time step of $\Delta t = 10^{-4}$ [s] and a forward Euler integration scheme. The time horizon is chosen sufficiently large to capture the steady state configuration of the system.
- Efficiency: running the solver on 64 cores, the total computation time is ~ 49 [min].

Hereunder, Figure 68, Figure 69 and Figure 70 depict the evolution of the SPH particles in the first stages of the system's evolution, until it reaches a steady state as depicted in Figure 71. These figures illustrate the focus property of the SPH method introduced in Section 3.1, stating that the particles inherently focus on the region of interest. In comparison, traditional mesh-based methods would require to mesh the relatively large portion of state space through which the PDF evolves, causing significant overhead.

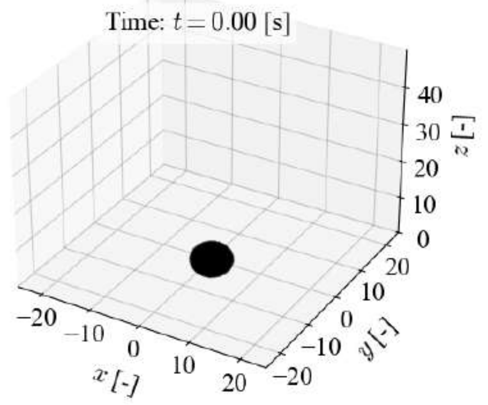


Figure 68: Configuration of the SPH particles at $t = 0$ [s]. The particles are packed in a ball of radius 4 [-] around the origin.

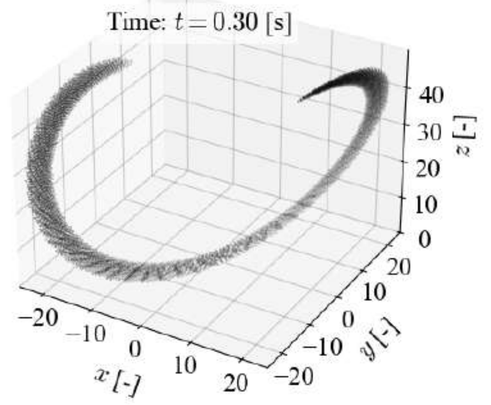


Figure 69: Configuration of the SPH particles at $t = 0.30$ [s]. The particles spread out in helicoidal shape.

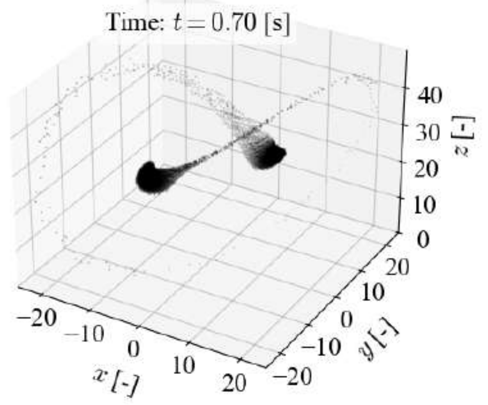


Figure 70: Configuration of the SPH particles at $t = 0.70$ [s]. The two branches agglomerate at two distinct states.

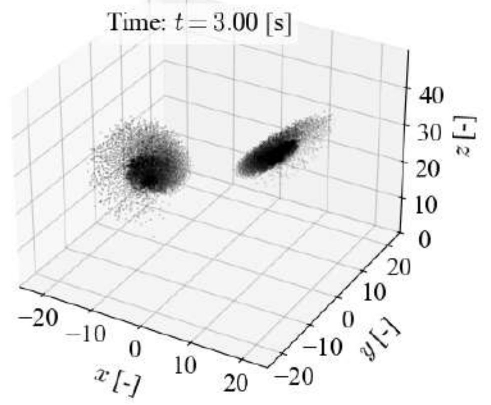


Figure 71: Configuration of the SPH particles at $t = 3$ [s]. The two agglomerates rotate and spread in intersecting planes.

4.3.3 MCS statement

For the MCS method, the results are obtained by direct numerical integration of the SDE given by Equation 164, and by considering the following setup.

- Initial condition: the MCS samples each start from a state $\underline{X}_0 = (x(0), y(0), z(0))$ drawn from

$$x(0), y(0), z(0) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1). \tag{166}$$

- Sampling resolution: a total of $N = 10^4$ realizations are derived from the MCS, which is comparable to the number of SPH particles.
- Time integration: as for the FPK-SPH simulation, a time horizon $T = 3$ [s] is considered, using a fixed time step of $\Delta t = 10^{-4}$ [s]. The integration method is the Euler-Maruyama time integration scheme introduced in Section 2.4.

4.3.4 Results

The results of the FPK-SPH method and the MCS method are presented for the second and sixth raw statistical moments for each component of $\underline{X}$. For each statistical moment, the results are presented under two different forms: a regular plot of the evolution of the moment for both SPH and MCS, and a confidence plot.

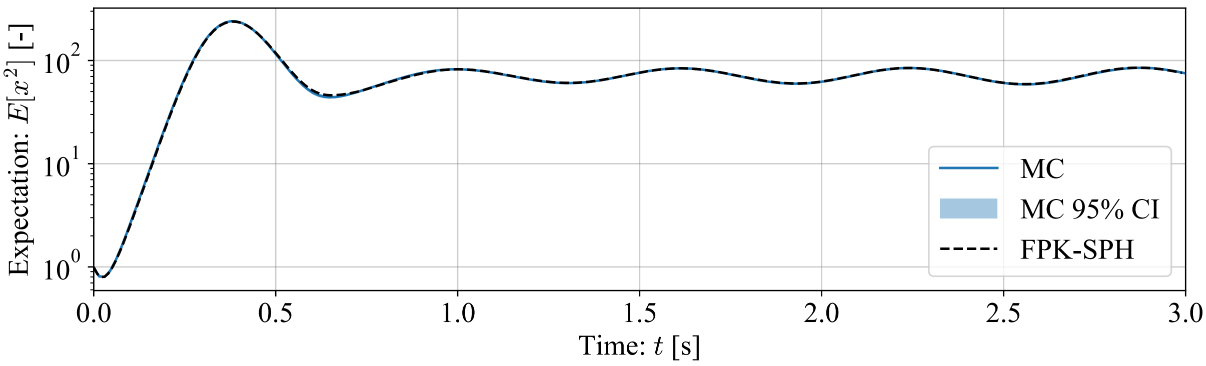


Figure 72: Evolution of the second raw moment of x, for both FPK-SPH and MCS methods.

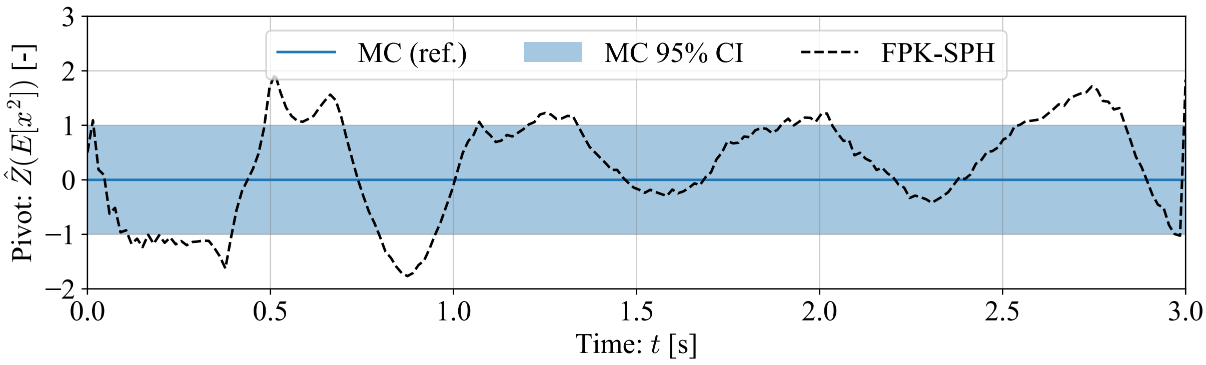


Figure 73: Confidence plot for the evolution of the second raw moment pivot for x, for both FPK-SPH and MCS methods.

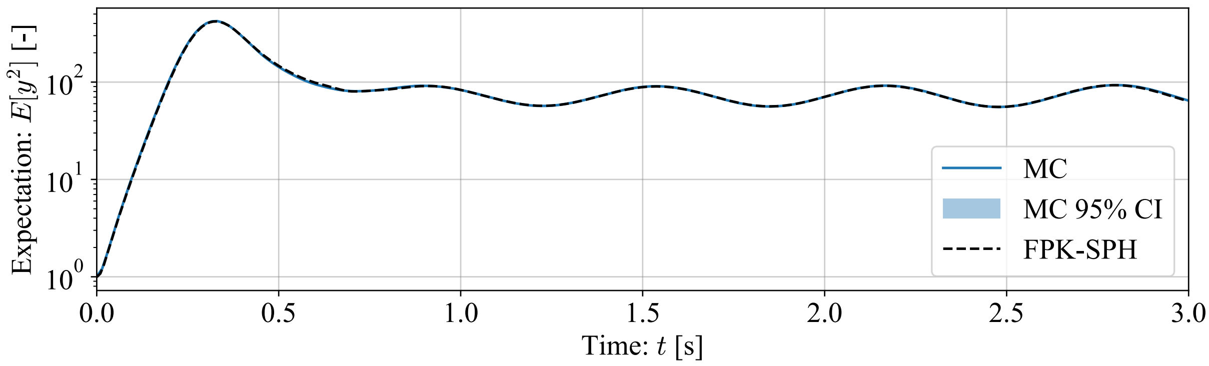


Figure 74: Evolution of the second raw moment of y, for both FPK-SPH and MCS methods.

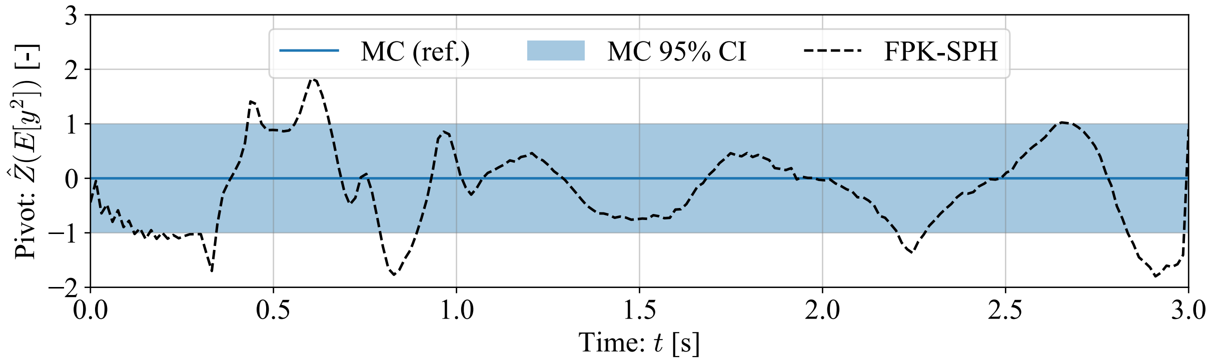
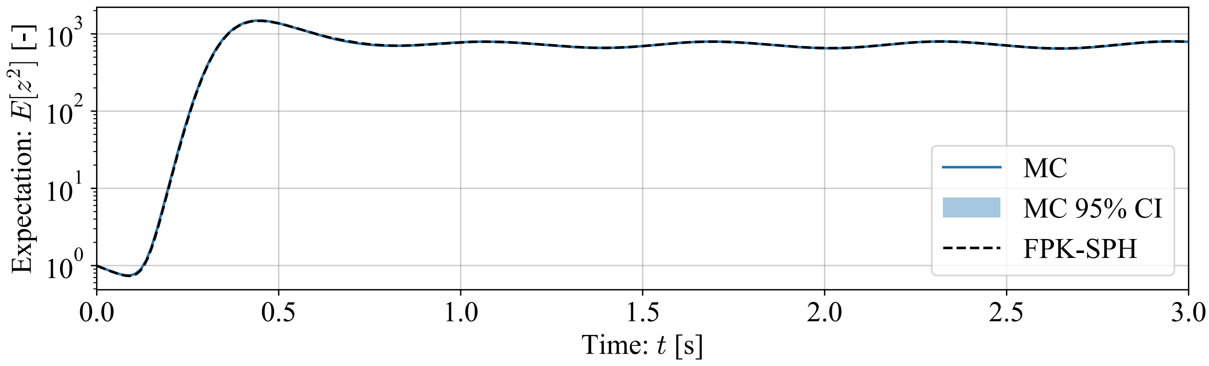


Figure 75: Confidence plot for the evolution of the second raw moment pivot for y, for both FPK-SPH and MCS methods.



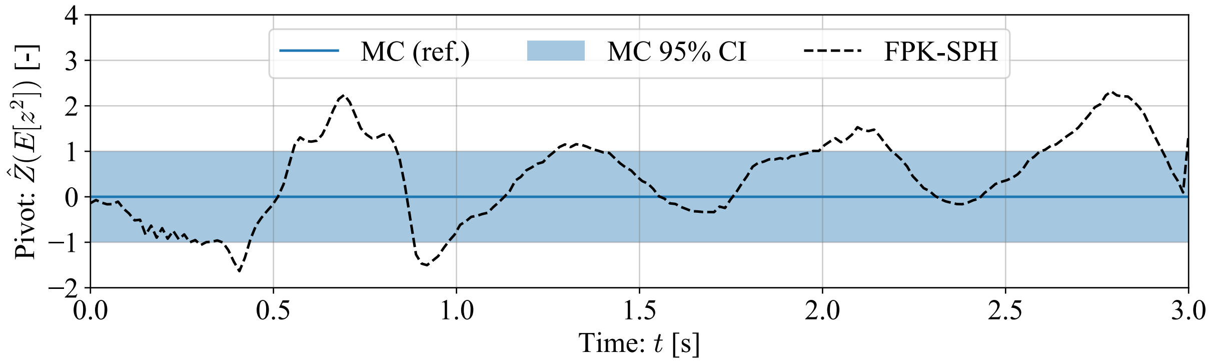


Figure 77: Confidence plot for the evolution of the second raw moment pivot for z, for both FPK-SPH and MCS methods.

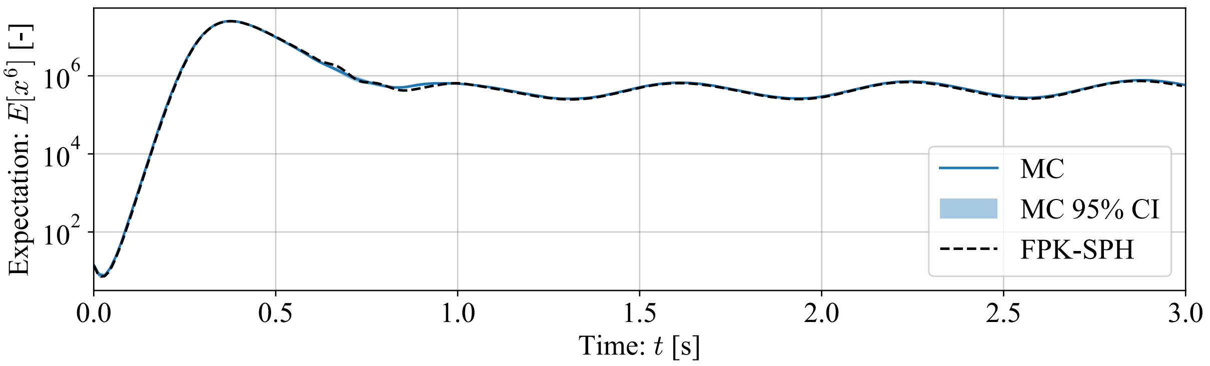


Figure 78: Evolution of the sixth raw moment of x, for both FPK-SPH and MCS methods.

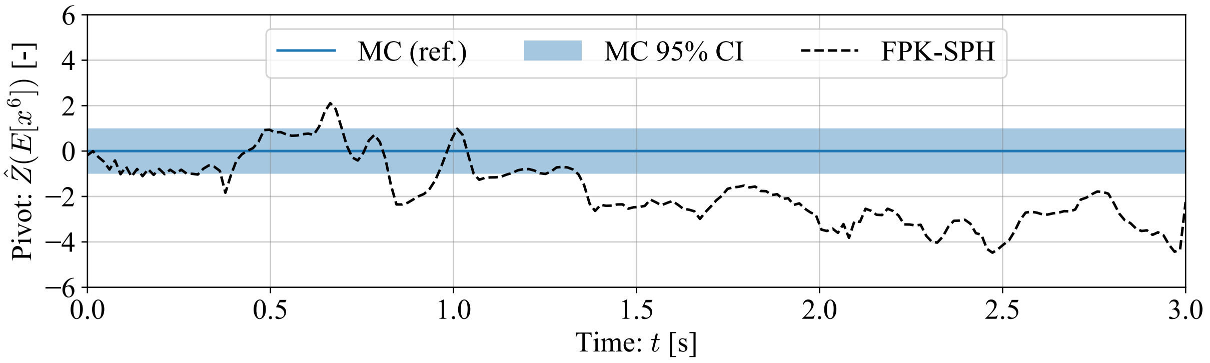


Figure 79: Confidence plot for the evolution of the sixth raw moment pivot for x, for both FPK-SPH and MCS methods.

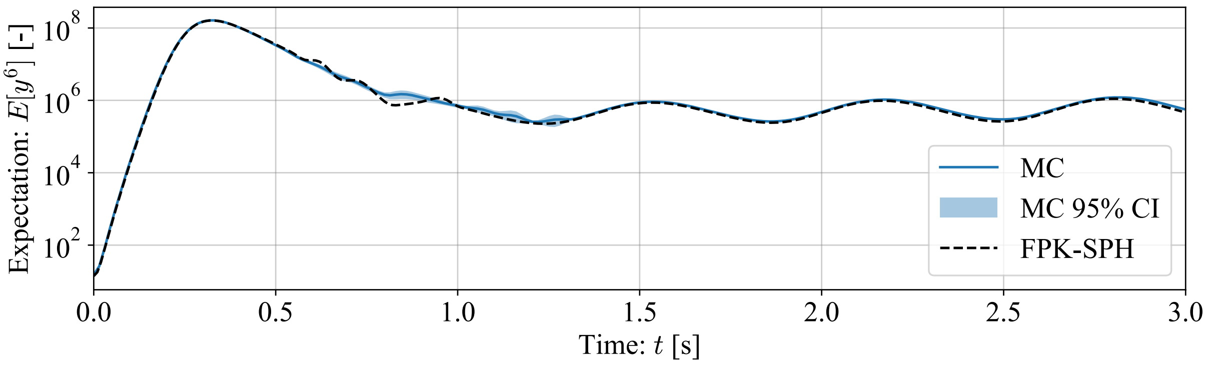


Figure 80: Evolution of the sixth raw moment of y, for both FPK-SPH and MCS methods.

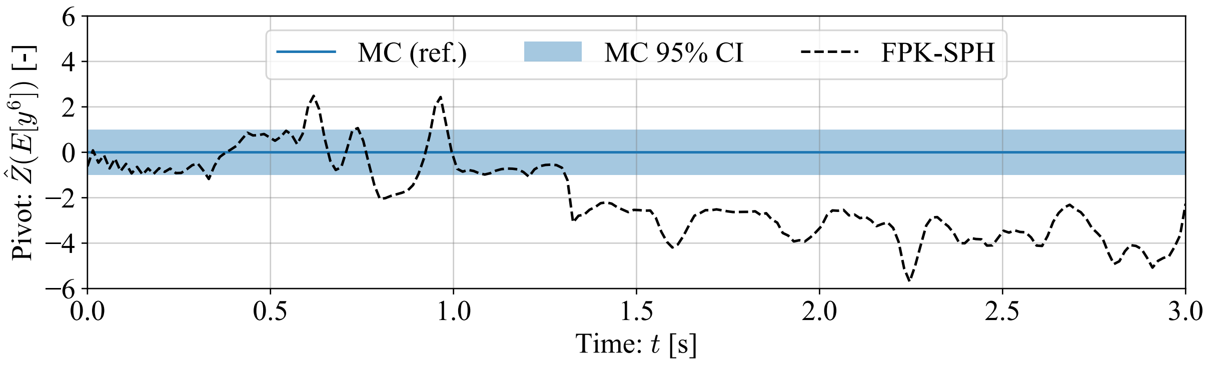


Figure 81: Confidence plot for the evolution of the sixth raw moment pivot for y, for both FPK-SPH and MCS methods.

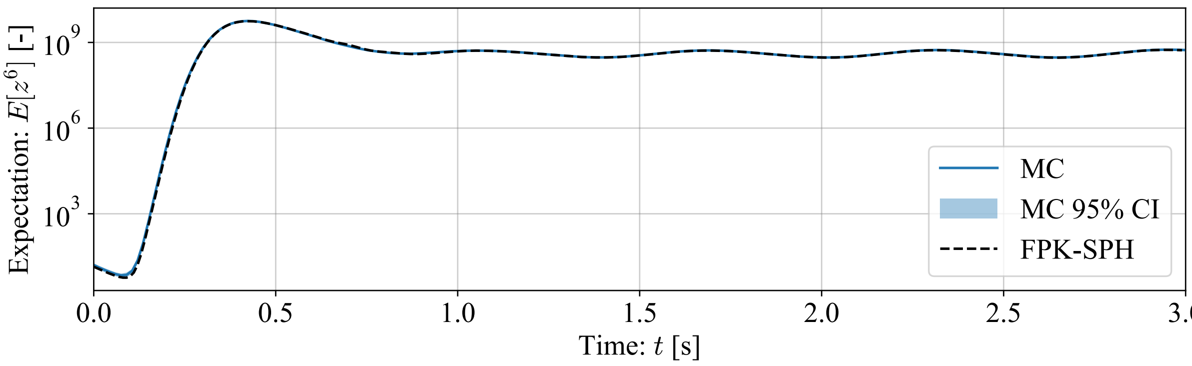


Figure 82: Evolution of the sixth raw moment of z, for both FPK-SPH and MCS methods.

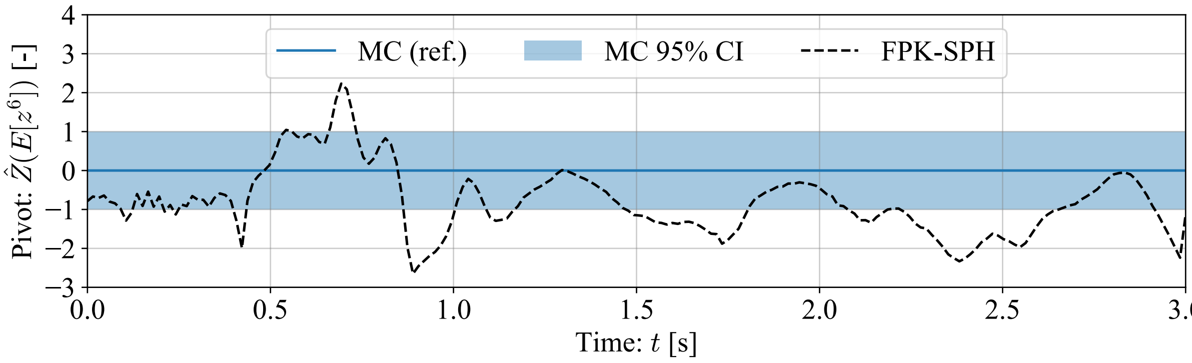


Figure 83: Confidence plot for the evolution of the sixth raw moment pivot for z, for both FPK-SPH and MCS methods.

Overall, the results are very satisfactory. As the figures show, the FPK-SPH method and the MCS method are very close for most raw statistical moments. A first comment can be done on the small discrepancies which appear for the sixth x and y raw statistical moments when $t \in [0.5, 1.0]$ [s]. During this time period, which corresponds to the creation of the two particle agglomerates represented in Figure 70, the FPK-SPH solver experiences some numerical errors. However, the MCS method seems to experience the same instability in its results, as its confidence interval suddenly grows. A second comment is done regarding the long-term errors which appear in Figure 79 and Figure 81 for $t \in [1.0, 3.0]$ [s]. In this time interval, the FPK-SPH and MCS methods seem to have converged to slightly different steady state solutions, which results in an everlasting error between the two solutions. Nevertheless, the error stays largely acceptable, as Figure 78 and Figure 80 show.

4.3.5 Performance improvement

In his work, [Can14] presents the results obtained for a slightly modified test case, with the computational resources available in 2014. Since improving the method with modern computational techniques is one of the main claims of this work, we suggest a raw comparison of the computational performance, as summarized in Table 5. In this scope, the aforementioned input parameters are changed to correspond to those used by [Can14]. Overall, the computational power is increased thanks to modern hardware capabilities and multi-core acceleration implementation. The additional computational power contributes both to making the solver faster ($\sim \times 20$ speedup for this test case) and more versatile (all the inputs are treated at runtime).

Criterion	From [Can14]	Present study
Number of time steps	30000 [-] (same)	30000 [-] (same)
Number of particles	9261 [-] (same)	9261 [-] (same)
Processor	Intel Core i5 at 2.3GHz	AMD Epyc Rome 7542 at 2.9 GHz
Programming language	Fortran 90	C++ using `g++` v13.2
Input	Hard coded	Versatile `json` inputs
Number of threads	1	64 (using `OpenMP`)
Computation time	$\sim$ 30000 [s]	$\sim$ 1400 [s]

Table 5: Comparison of the computational performance of the FPK-SPH solver used in [Can14] and in the present study. Hardware and software information is given for fair comparison.

4.4 Compartmental epidemiological model

4.4.1 Problem definition

Compartmental epidemiological models [Bra08] describe the spread of a disease within a population by dividing individuals into distinct health states, called compartments. For instance, the Susceptible-Infected-Recovered-Deceased (SIRD) model [CNP20] is a simplified model, defining four such compartments, among which the population is distributed. By identifying the fluxes that govern the transition from one compartment to another, one can simulate the impact of the disease on the population's health state. The quality of a compartmental model strongly relies on the choice of compartments and on how the fluxes are modeled. In this study, the focus is directed towards a variation of the SIRD model, in which stochastic factors come into play.

Traditionally, the deterministic SIRD model defines four states $S(t), I(t), R(t)$ and $D(t)$, representing the proportions of the population in the susceptible, infected, recovered and deceased compartments, respectively, at time t. By construction, these states are such that

$$S(t) + I(t) + R(t) + D(t) = 1 \tag{167}$$

at any instant. The evolution of these states, represented in Figure 84, is dictated by

$$\begin{cases} \mathrm{d}_t S = -\beta S I \\ \mathrm{d}_t I = \beta S I - \gamma I - \eta I \\ \mathrm{d}_t R = \gamma I \\ \mathrm{d}_t D = \eta I, \end{cases} \tag{168}$$

where β, γ and η are related to the transmissibility, recovery rate and mortality rate of the disease, respectively. The SIRD model makes the strong assumption that people who have recovered from the disease cannot get infected anymore.

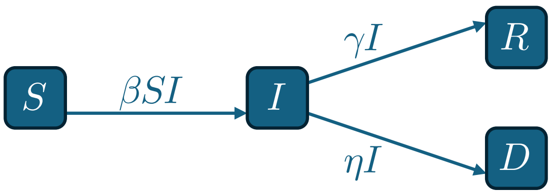


Figure 84: Representation of the SIRD model, the blue boxes represent the different compartments and the arrows indicate inter-compartment fluxes.

We propose to modify this problem to take into account various forms of uncertainties that can appear in practice. Specifically, the initial values of S, I, R and D are assumed to be random variables such that $S(0), I(0)$ and $D(0)$ are drawn from a known probability distribution and $R(0)$ is simply given by $R(0) = 1 - S(0) - I(0) - D(0)$. This probability distribution ought to be well chosen, to avoid negative values of R. Moreover, to capture the unpredictable influences affecting the epidemiological dynamics, β, γ and η are affected by random variations during the entire time horizon, as detailed next. To ensure these parameters remain within a meaningful range, the following decomposition is introduced.

$$x(t) = \overline{x} + \Delta x \Phi(x'(t)), \tag{169}$$

where x can be replaced by β, γ and η, and where $\Phi(x)$ is a continuous bijective mapping from $\mathbb{R}$ to $]-1;1[$ such that, at any time, $x \in\ ]\overline{x} - \Delta x; \overline{x} + \Delta x[$. For this study, the arctangent mapping is chosen, as represented in Figure 85.

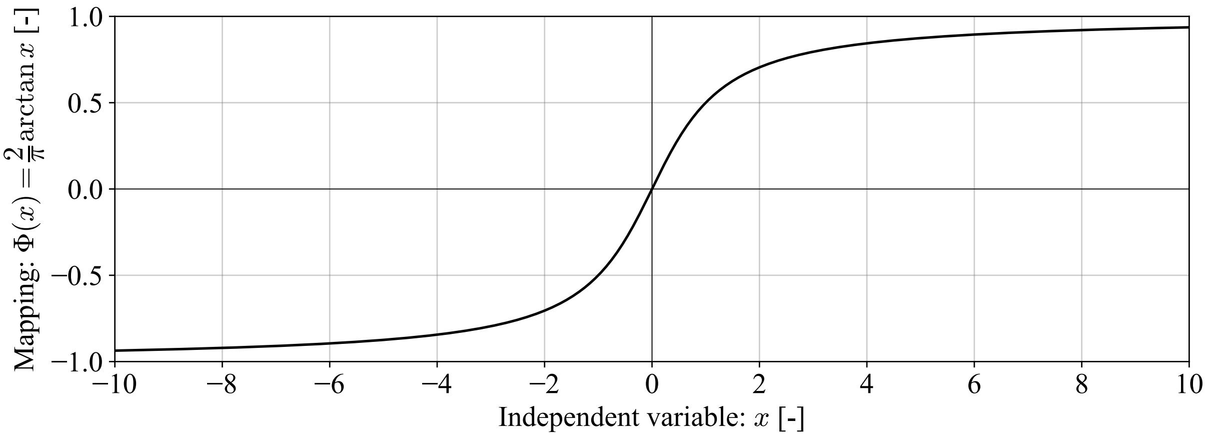


Figure 85: Mapping chosen to ensure that fluctuations in β, γ and η remain bounded.

For simplicity, we model x' as a scaled Wiener process

$$\mathrm{d}x'(t) = g_x \,\mathrm{d}W_x(t), \tag{170}$$

where g_x is assumed to be constant. The initial condition of β', γ' and η' is given by a supposedly known probability distribution. In the end, the SDE describing the evolution of $\underline{X} = (S, I, D, \beta', \gamma', \eta')$ is

$$\mathrm{d}\begin{pmatrix} S \\ I \\ D \\ \beta' \\ \gamma' \\ \eta' \end{pmatrix} = \underbrace{\begin{pmatrix} -\big(\overline{\beta} + \Delta\beta\Phi(\beta')\big)SI \\ \big(\overline{\beta} + \Delta\beta\Phi(\beta')\big)SI - (\overline{\gamma} + \Delta\gamma\Phi(\gamma'))I - (\overline{\eta} + \Delta\eta\Phi(\eta'))I \\ (\overline{\eta} + \Delta\eta\Phi(\eta'))I \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\underline{f}(\underline{X},t)} \mathrm{d}t \tag{171(1)}$$

$$+ \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_\beta & 0 & 0 \\ 0 & 0 & 0 & 0 & g_\gamma & 0 \\ 0 & 0 & 0 & 0 & 0 & g_\eta \end{pmatrix}}_{\underline{\underline{g}}(\underline{X},t)} \mathrm{d}\underline{W}. \tag{171(2)}$$

The relation $S(t) + I(t) + R(t) + D(t) = 1$ is used once again, here for recovering $R(t)$ without having to compute it with the SDE.

The FPK-SPH method is applied to this test case for benchmarking purposes, as it presents both practical and theoretical interest. On the one hand, it demonstrates the method's applicability to real-world problems, such as epidemic modeling. On the other hand, it challenges the solver with a six-dimensional problem, which is to our knowledge an achievement not yet demonstrated in the current state of the art.

The results discussed in Section 4.4.4 are obtained by using the parameter values indicated in Table 6.

		Transmissibility	**Recovery rate**	**Mortality rate**
Average	[day^{-1}]	$\overline{\beta} = 0.3$	$\overline{\gamma} = 0.1$	$\overline{\eta} = 0.01$
Half-span	[day^{-1}]	$\Delta\beta = 0.25$	$\Delta\gamma = 0.05$	$\Delta\eta = 0.005$
Variability	[day^{-1}]	$g_\beta = 0.1$	$g_\gamma = 0.1$	$g_\eta = 0.1$

Table 6: Parameter values used for the SIRD test case.

4.4.2 FPK-SPH statement

Under the FPK-SPH formalism, the problem at hand is defined as follows.

- Initial condition: one assumes that the initial probability distribution is a uniform distribution over the region occupied by the SPH particles. In other words, the particle masses are all set to the same value. A Gaussian kernel is used.
- Initial arrangement of particles: the particles are arranged in a Cartesian interval of $\mathbb{R}^6$, denoted Ω and characterized by $S(0) \in [0.88; 0.92]$, $I(0) \in [0.03; 0.07]$, $D(0) \in [0.0025; 0.0125]$ and $(\beta'(0), \gamma'(0), \eta'(0)) \in [-0.005; 0.005]^3$. The interval is covered by a Cartesian lattice of SPH particles, with a spacing $s = 0.005$, resulting in $N = 6561$ particles. While the number of particles seems relatively large, the per-dimension resolution remains low (between 3 and 9 particles per axis), highlighting the curse of dimensionality, where even coarse discretizations become computationally intensive as dimensionality increases. To illustrate this, consider Figure 86, in which the initial arrangement of particles is represented as a projection in the (S, I, D) space. Despite using $N = 6561$ particles, the resolution in the 3D (S, I, D) space alone ***remains very coarse***.

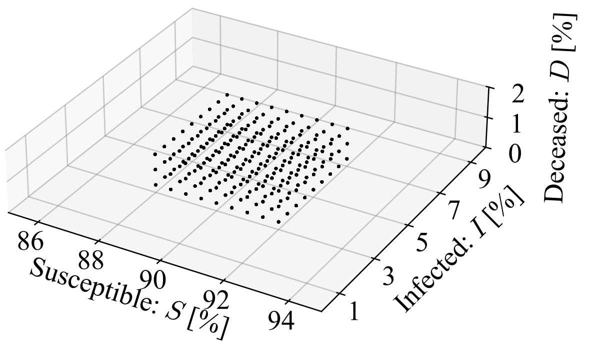


Figure 86: Configuration of the SPH particles at $t = 0$ [day], projected in the (S, I, D) space.

- Time integration: a time horizon of $T = 50$ [day] is simulated, which is sufficient to capture the epidemiological dynamics and, in particular, the infection peak. An adaptive time step (relative convention) is used with a stability coefficient $S = 0.8$ [-] and a maximum time step $\Delta t_{\max} = 1$ [day]. The midpoint method is used for numerical time integration.

- Efficiency: running the solver on 64 cores results in a total runtime of ~ 2.5 [h].

The evolution of the SPH particles in the (S, I, D) state space is shown at different time references in Figure 87 to Figure 90.

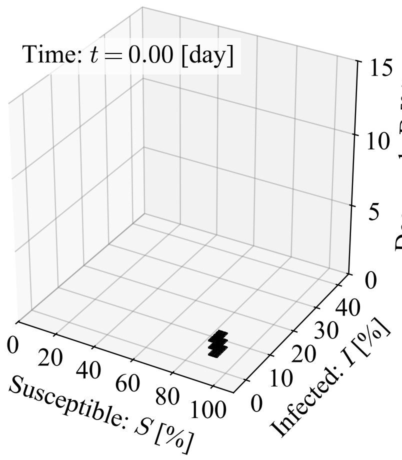


Figure 87: Configuration of the SPH particles at $t = 0$ [day]. The particles are regularly packed in a hypercube.

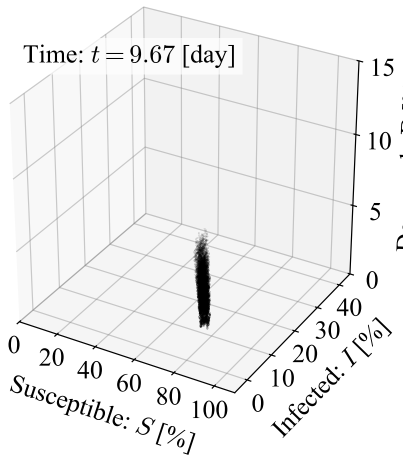


Figure 88: Configuration of the SPH particles at $t \approx 10$ [day]. The distribution of SPH particles stretches as susceptible people get infected.

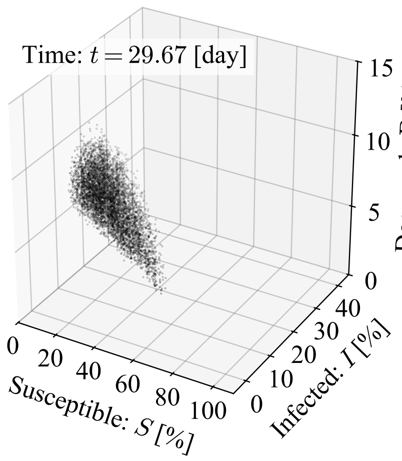


Figure 89: Configuration of the SPH particles at $t \approx 30$ [day]. The cloud of particles spreads as uncertainty grows. The susceptible population has decreased in favor of the infected, recovered and deceased states.

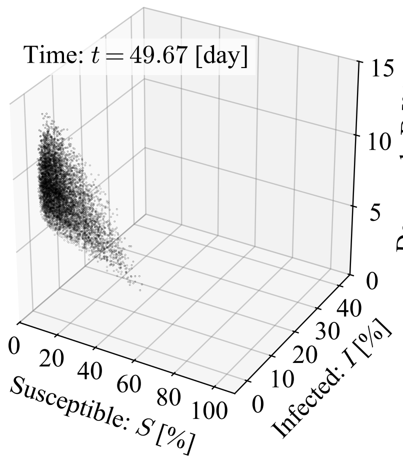


Figure 90: Configuration of the SPH particles at $t \approx 50$ [day]. The population slowly converges to the vicinity of $S = 0$, leaving the main source of uncertainty on I, R and D.

It is important to keep in mind that these visualizations are *projections* from a subspace of $\mathbb{R}^6$ to a subspace of $\mathbb{R}^3$. As a result, the figures may misleadingly suggest a high resolution in the region of interest. In reality, the underlying resolution in $\mathbb{R}^6$ remains low. This phenomenon is depicted in Figure 91: a sparse coverage in a high-dimensional space may erroneously appear dense when viewed in a projection.

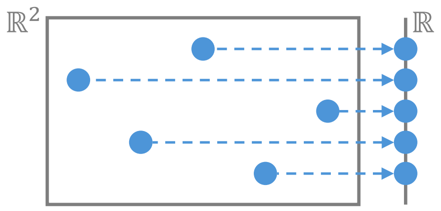


Figure 91: A sparse scattering of particles in $\mathbb{R}^2$ is projected to a dense coverage of $\mathbb{R}$.

4.4.3 MCS statement

Regarding the MCS resolution, the problem is defined as follows.

- Initial condition: as for the FPK-SPH formulation of the problem, one assumes a uniform probability distribution. The support of this distribution corresponds to the region Ω of state space covered by SPH particles, extended by half the inter-particle spacing s in each dimension to account for particle extent. The components of the initial state vector $\underline{X}(0)$ are independently and identically distributed as follows:

$$S(0) \overset{\text{i.i.d.}}{\sim} \text{Unif}\left(0.88 - \frac{s}{2}, 0.92 + \frac{s}{2}\right) = \text{Unif}(0.8775, 0.9225), \tag{172}$$

$$I(0) \overset{\text{i.i.d.}}{\sim} \text{Unif}\left(0.03 - \frac{s}{2}, 0.07 + \frac{s}{2}\right) = \text{Unif}(0.0275, 0.0725), \tag{173}$$

$$D(0) \overset{\text{i.i.d.}}{\sim} \text{Unif}\left(0.0025 - \frac{s}{2}, 0.0075 + \frac{s}{2}\right) = \text{Unif}(0.0, 0.015), \tag{174}$$

$$\beta'(0), \gamma'(0), \eta'(0) \overset{\text{i.i.d.}}{\sim} \text{Unif}\left(-0.005 - \frac{s}{2}, 0.005 + \frac{s}{2}\right) = \text{Unif}(-0.0075, 0.0075). \tag{175}$$

- Sampling resolution: a total of $N = 10^5$ MCS samples are drawn, which suffices to obtain convergence.
- Time integration: as for the FPK-SPH resolution, a time horizon of $T = 50$ [day] is considered, using a fixed time step $\Delta t = 0.01$ [day]. The SDE describing the evolution of the system is numerically integrated using the Euler-Maruyama scheme (see Section 2.4).

4.4.4 Results

In the following figures, the results of the FPK-SPH and MCS methods are presented. Compared to the previous test cases, the results yielded by the FPK-SPH method are less accurate, presumably because the method is limited to a low resolution, as previously discussed. Consequently, and given that the error induced by the converged MCS method is negligible compared to that of the FPK-SPH method, the confidence plots are replaced by more traditional absolute error plots. These plots illustrate the time evolution of the absolute error associated with the estimation of the expected value

$$|\Delta E[\varphi(\underline{X})]| = |E^{\text{SPH}}[\varphi(\underline{X})] - E^{\text{MC}}[\varphi(\underline{X})]|, \tag{176}$$

where $\varphi(\underline{X})$ is some scalar function of $\underline{X}$.

The first order statistical moments $E[\cdot]$ and corresponding absolute errors $|\Delta E[\cdot]|$ are depicted in Figure 92 to Figure 97, for S, I and D. Then, the standard deviations $\sigma[\cdot]$, calculated as

$$\sigma[\varphi(\underline{X})] = \sqrt{E[\varphi(\underline{X})^2] - E^2[\varphi(\underline{X})]}, \tag{177}$$

are given for S, I and D in Figure 98 to Figure 100.

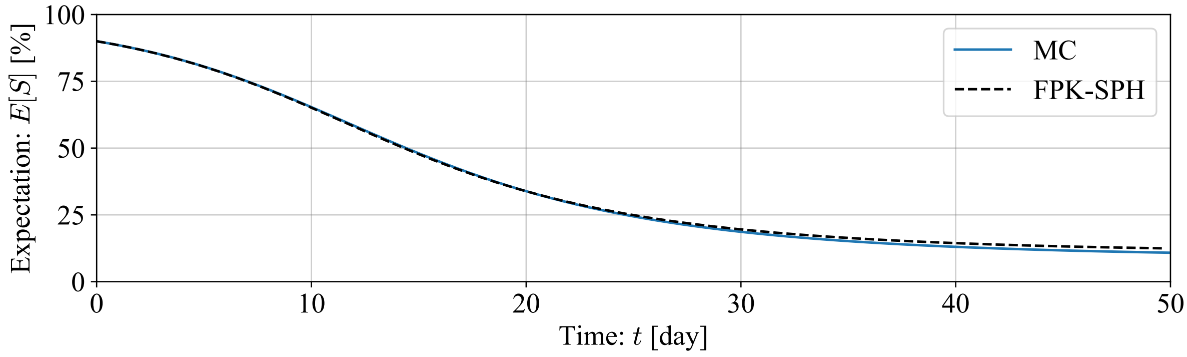


Figure 92: Evolution of the expected susceptible population percentage, for both FPK-SPH and MCS methods.

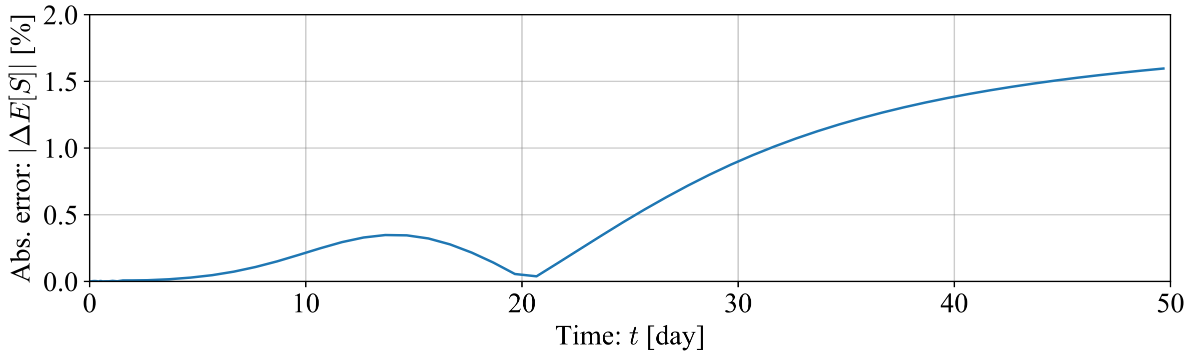


Figure 93: Evolution of the absolute error between the FPK-SPH and MCS methods, for the expected susceptible population percentage.

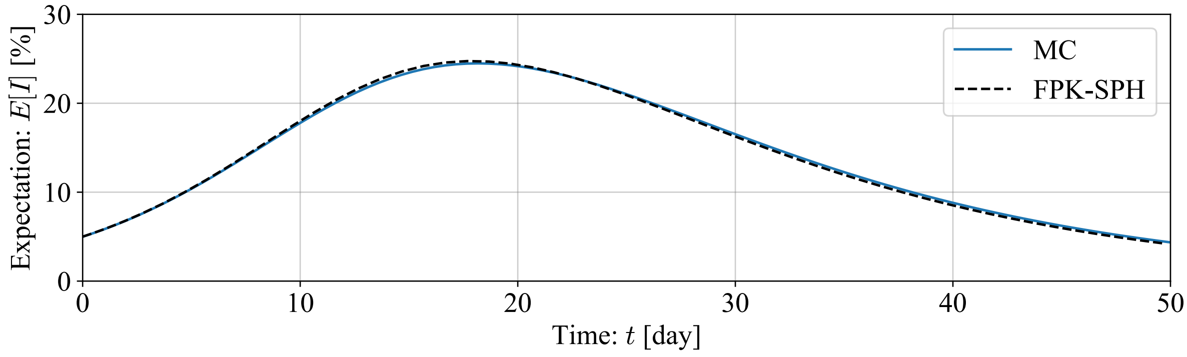


Figure 94: Evolution of the expected infected population percentage, for both FPK-SPH and MCS methods.

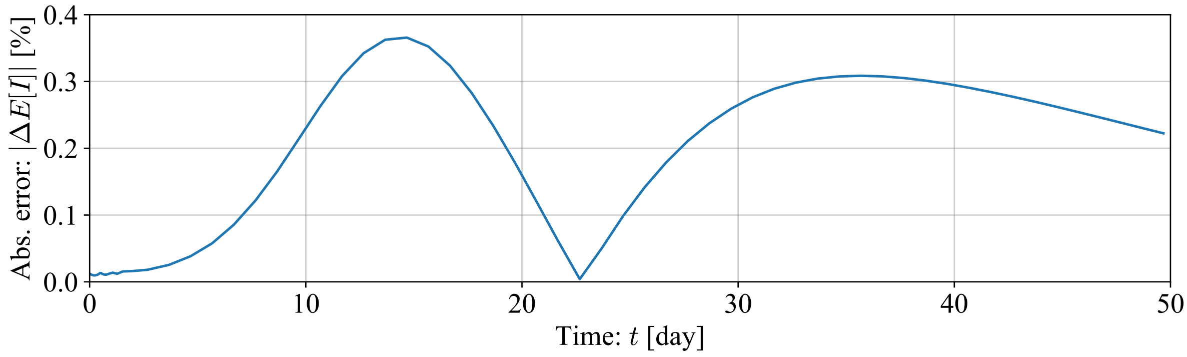


Figure 95: Evolution of the absolute error between the FPK-SPH and MCS methods, for the expected infected population percentage.

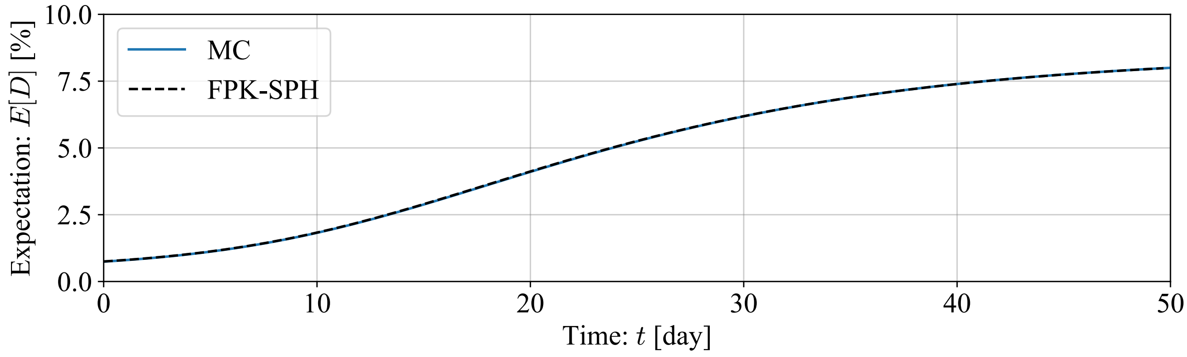


Figure 96: Evolution of the expected deceased population percentage, for both FPK-SPH and MCS methods.

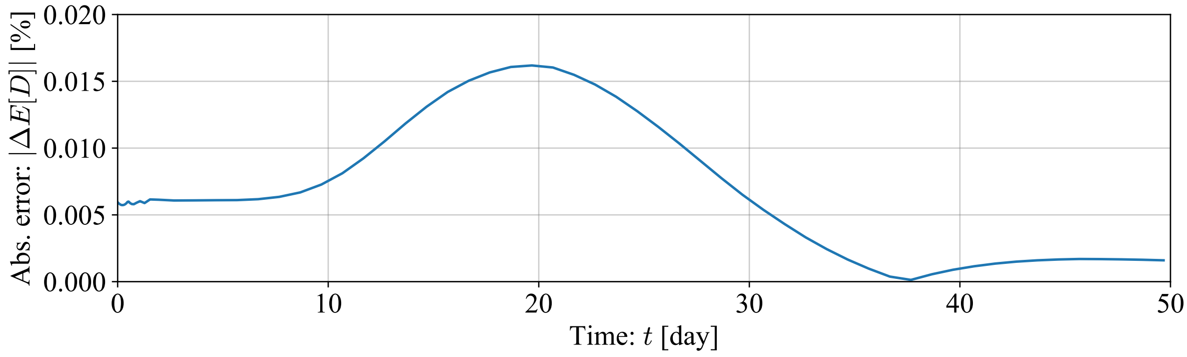


Figure 97: Evolution of the absolute error between the FPK-SPH and MCS methods, for the expected deceased population percentage.

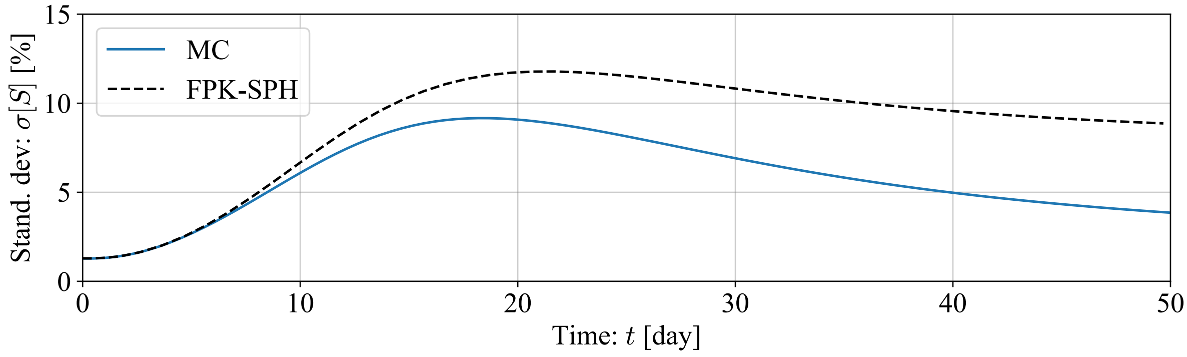


Figure 98: Evolution of the standard deviation of the susceptible population percentage, for both FPK-SPH and MCS methods.

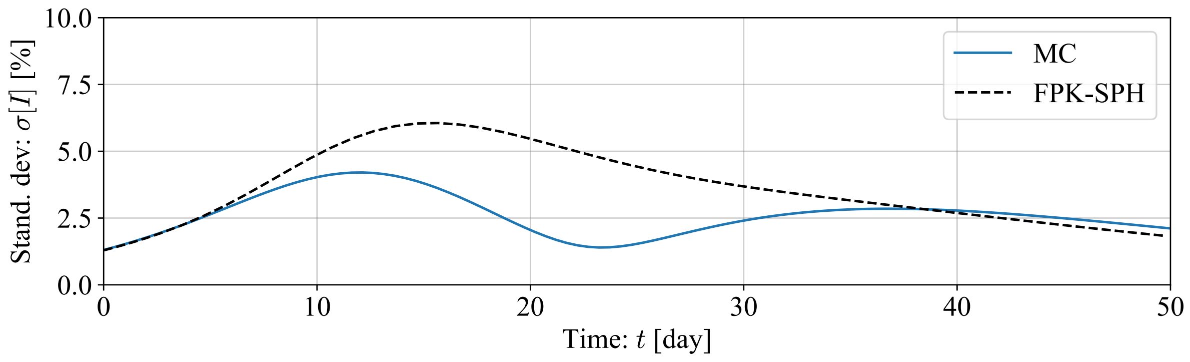


Figure 99: Evolution of the standard deviation of the infected population percentage, for both FPK-SPH and MCS methods.

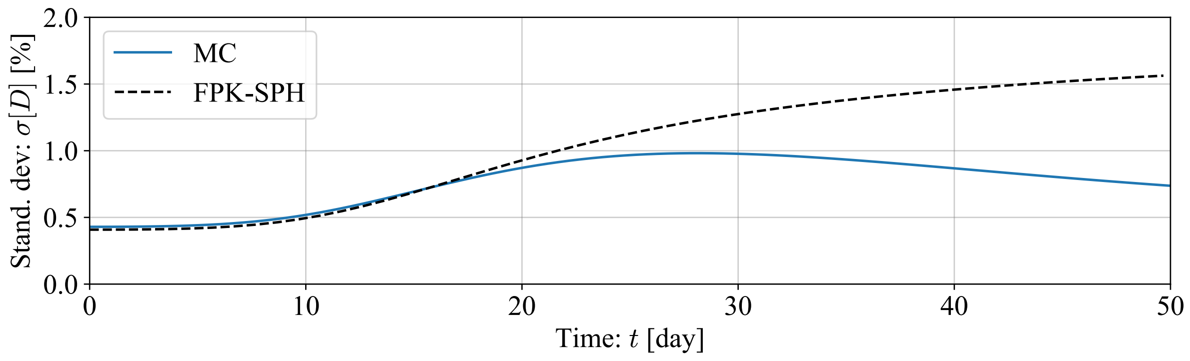


Figure 100: Evolution of the standard deviation of the deceased population percentage, for both FPK-SPH and MCS methods.

Overall, the FPK-SPH method demonstrates satisfactory accuracy for the first statistical moments, inducing an error of at most 2 [%]. As one can tell, the results for the standard deviations are less convincing. While using a finer arrangement of SPH particles would likely yield better results, such improvements are currently limited by available computational power and memory. As illustrated in Figure 101, each particle connects to more neighbors as dimensionality increases. This higher connectivity heavily degrades the efficiency of the neighbor search algorithm, whose computational and memory costs grow with dimensionality, according to a power law. For high-dimensional problems, this issue should be addressed by implementing a more efficient neighbor search algorithm on a manifold.

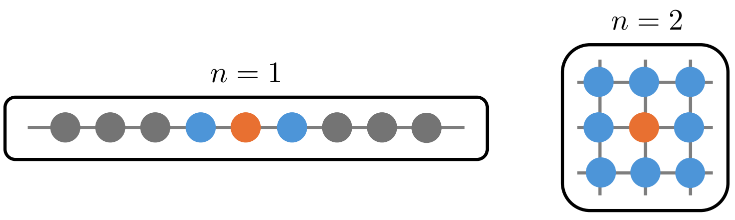


Figure 101: Illustration of the increase in number of neighbors (blue) of a reference particle (orange) as the number of dimensions increases.

To illustrate the practical relevance of the probability of exceedance introduced in Equation 129, consider the following situtation. Let $I_{\max}$ represent the maximum proportion of infected individuals that the healthcare system can accommodate. In a deterministic setting, one might ask: “Will the healthcare system be overwhelmed?”. In a stochastic context, however, this becomes: “What is the probability that the healthcare system will be overwhelmed?”. This question is directly answered by evaluating the probability of exceedance $P(I > I_{\max})$, which can be computed for varying thresholds $I_{\max}$. The results obtained with the FPK-SPH and MCS methods are shown in exceedance plots in Figure 102 for qualitative comparison. For a more quantitative evaluation of the error caused by the FPK-SPH solver, let us consider Figure 103, in which the absolute error is plotted.

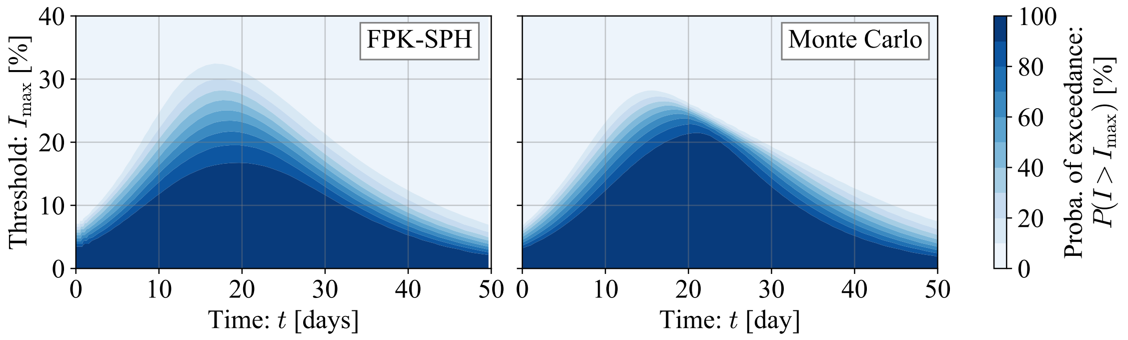


Figure 102: Comparison of the exceedance plots obtained with the FPK-SPH (left) and MCS (right) methods.

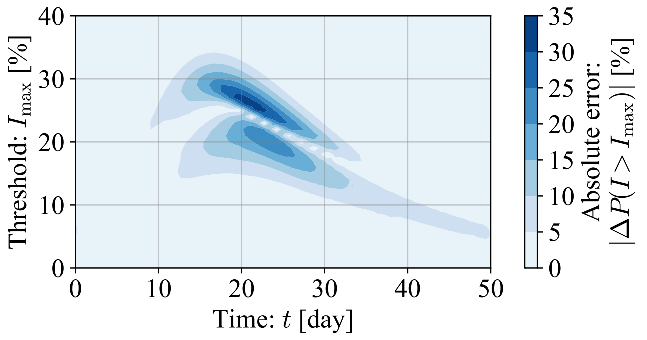


Figure 103: Absolute error between the exceedance plots generated by the FPK-SPH and MCS methods.

The SIRD test case highlights the solver's flexibility in handling high-dimensional problems, demonstrating its ability to operate effectively in six dimensions, which is a notable achievement. However, the curse of dimensionality severely limits the achievable resolution, and the current implementation incurs substantial computational and memory demands. As a result, simulations can only deliver coarse approximations. Ultimately, the SIRD test case pushes the solver to its limits, and could be the starting point for a more thorough study into algorithmic improvements aimed at reducing computational and memory costs in high-dimensional settings.

5 Numerical aspects of the solver

5.1 Introduction

The main goal of this section is to provide all the information a user could find useful to run the solver. To this end, the solver is first presented in terms of inputs and outputs for who wishes to use the solver as a black box. Then, an overview of the general algorithm is presented, as well as some implementation choices. Finally, this part closes upon some metrics describing the sequential and parallel performances of the solver. Let us simply remind that the entire project is hosted in a GitLab repository[3], which provides additional information for hands-on usage of the solver, as well as additional documentation, examples and illustrations.

5.2 Input file

The simulation parameters are fed into the solver via a JSON file, which is then parsed by the `json` library [Loh24]. Therefore, the input parameters only need to be known by the program at runtime, thus offering high flexibility, at a computational cost. In this section, the different parts of an input file are described, and the parameters are explained.

5.2.1 Outputting

The `"output"` field defines the general parameters for the results export.

```
"output": {
  "filename"    :  ... ,
  "csvExport"   :  ... ,
  "refreshRate" :  ...
}
```

In this `json` object, `"filename"` is the string upon which all the output files are named. More precisely, the output files are composed of the string, followed by the time step, and a file

[3]Available at https://gitlab.uliege.be/structural-and-stochastic-dynamics/fpk-sph-solver.

extension (which is `.csv` for raw data). The `"csvExport"` field is a boolean value that indicates whether the raw data — that is, for each time step, the state, velocity, smoothing length, density and density gradient of each particle — should be exported at all. If set to `false`, the raw data output is not performed, which can have a significant impact on the performance of the solver. Finally, `"refreshRate"` is the number of time steps which are performed between two successive outputs. This is useful to reduce the number of output files, and thus the amount of data to be stored. Choosing a low value for this parameter can significantly increase the performance of the solver as well.

5.2.2 Time management

Defining the time-related parameters is crucial for the simulation. This is done within the `"time"` field, the structure of which is the following.

```
"time": {
  "interval"    :  ... ,
  "stepping"    : [...],
  "integration" :  ...
}
```

The first parameter, `"interval"`, is simply the total duration of the simulation. The second parameter, `"stepping"`, is relatively flexible as it proposes two different use cases:

- Fixed time stepping: the time step is fixed to a constant value `dt` throughout the simulation. In this case, the parameter should be set to `[false, dt]`.
- Adaptive time stepping: the time step is adapted to the relative velocity and proximity of the particles, as explained in Section 3.6.3. For a given stability parameter `S` and a maximum allowed time step `dt_max`, the parameter should be set to `[true, S, dt_max]`.

The last parameter, `"integration"`, is the integration method used to integrate numerically through time. The available methods are the explicit forward Euler method (see Section 3.6.1) and the midpoint method (see Section 3.6.2), which are specified with the two values `"euler"` and `"midpoint"`, respectively.

5.2.3 Particles generation

The `"particles"` field defines the parameters which characterize the initial arrangement of particles. The arrangement is a hyper-rectangular grid, which is defined by the following parameters. A more exotic shape can be extracted from the particle grid, as explained in Section 5.2.4.

```
"particles": {
  "origin"      : [...],
  "size"        : [...],
  "spacing"     :  ...
}
```

The first two parameters, `"origin"` and `"size"`, are vectors containing the coordinates of the origin and the size of the particle grid, respectively. The third parameter, `"spacing"`, is the targeted distance s between two adjacent particles in the grid, along each principal direction[4]. If the spacing and the size are not compatible, the spacing is lowered to the closest value that is compatible with the size.

[4] Albeit the spacing is assumed to be the same in each direction, the user is free to operate a simple change of coordinates, namely a scaling of these, and adjust the drift and diffusion expressions accordingly. This way, the spacing is constant in the stretched coordinates but not in the original coordinates.

5.2.4 Kernel

All the aspects of the simulation which are connected to the kernel, or to the continuous approximation for that matter, are indicated in the `"kernel"` field.

```
"kernel": {
  "type"                : ... ,
  "coupling"            : ... ,
  "maxNeighbors"        : ... ,
  "adaptiveSmoothing"   : ... ,
  "iterativeSolver"     : ... ,
  "iterationsMax"       : ... ,
  "discardNotConverged" : ... ,
  "initialCondition"    : ...
}
```

The `"type"` parameter refers to the nature of the kernel to use for the simulation. The choice should be made among the kernels presented in Section 3.3.2, namely `"gaussian"`, `"cubic"` or `"lucy"`. The `"coupling"` provides a value for the coupling factor η introduced in Section 3.7.2; its value is typically between 1.2 and 1.5 [Mon05].

Next, `"maxNeighbors"` is the maximum number of neighbors that a particle is allowed to have. This restriction emerges from the fact that the memory required for the neighbor search algorithm is allocated statically. By doing so, the memory is allocated only once, and the performance of the solver is improved, especially when dealing with multiple cores and shared memory.

The `"adaptiveSmoothing"` parameter is a boolean value that indicates whether the adaptive smoothing length algorithm should be performed as in Section 3.7.2. The three next parameters are only meaningful in the case `"adaptiveSmoothing" = true`. First, the `"iterativeSolver"` parameter indicates which of the fixed point method (see Section 3.7.4.1) or the Newton-Raphson method (see Section 3.7.4.2) should be used as iterative solver for the adaptive smoothing length computation. The `"iterationsMax"` simply indicates the maximum number of iterations that should be performed before meeting the tolerance criterion for the chosen iterative solver. The solver iterates as long as not *all* of the particles have met the tolerance. If the solver has not converged after this number of iterations, the `"discardNotConverged"` parameter comes into play. If set to `true`, the particles that have not converged are discarded from the simulation and their probability mass is redistributed among the remaining particles. If set to `false`, the particles that have not converged are kept in the simulation anyway.

Finally, the `"initialCondition"` parameter is a string that defines the initial probability density function $\psi(\underline{X}, 0)$, which is evaluated by the `ExprTk` expression parsing library [Par99]. This parameter enables the implementation of complex initial distributions, including those with compact support. An important feature is that particles with zero initial probability density are automatically removed from the simulation, making it possible to construct non-rectangular particle arrangements. For example, a circular domain of unit radius can be generated by defining the initial condition as

`"initialCondition" = "if(x0^2 + x1^2 < 1, 1, 0)".`

Setting `"initialCondition" = ""` will result in a uniform distribution of the probability mass among the particles. In any case, the initial distribution is normalized such that the total probability mass carried by the particles equals unity.

5.2.5 Domain

The `"domain"` mainly defines the different parameters related to the neighbor search algorithm, explained in Section 3.8.

```
"domain": {
  "nDimensions"         :  ... ,
  "origin"              : [...],
  "size"                : [...],
  "type"                :  ... ,
  "cellScale"           :  ... ,
  "maxParticlesPerCell" :  ...
}
```

First, `"nDimensions` is a safe check that the number of dimensions of the problem is coherent within the whole input file. If at some point the size of a coordinate vector, even in another `json` field, is different from `"nDimensions"`, an error is raised. The `"origin` and `"size"` parameters are vectors containing the coordinates of the origin and the size of the cell grid, respectively.

The `"type"` parameter is either `"bounded"`, `"periodic"` or `"infinite"`, in reference to the homonymous domain types presented in Section 3.9. The `"cellScale"` parameter is the scale factor α introduced in Section 3.8. If the deduced cell size does not fit exactly in the cell grid, the size of the grid is increased to adapt to the cell size. In most cases, $\alpha = 1$ but it can be meaningful to change this value for increased computational performance or reduced memory usage. For instance, if the particles are initially densely packed with respect to the configuration they later adopt, it may be useful to increase the cell scale. Exploring an adaptive cell scale is a good perspective for future work, but care should be taken to ensure that the overhead caused by repeated memory allocation does not outweigh the performance gain.

Lastly, the `"maxParticlesPerCell"` parameter is the maximum number of particles that a cell is allowed to contain at any time step when applying the cell mapping algorithm (see Section 3.8). This limitation stems from the fact that the memory required for the cell mapping is allocated statically. By doing so, the memory is allocated only once, and the performance of the solver is improved, especially when dealing with multiple cores and shared memory.

5.2.6 Statistics

Yet another useful feature of the solver is the ability to compute statistics on the approximated probability density function. This computation is performed only at export time steps (see Section 5.2.1). The `"statistics"` field defines the parameters related to this feature.

```
"statistics": {
  "moments"     : [...],
  "exceedance"  : [...]
}
```

The `"moments"` parameter is a vector of vectors containing the orders of the moments to compute. For instance, `"moments" = [[1, 0], [2, 3]]` will compute two cross-moments, namely $m(1,0) = E[X_0]$ and $m(2,3) = E[X_0^2 X_1^3]$, following the notations introduced in Section 3.10.

The `"exceedance"` parameter is a vector of strings containing the expressions of the thresholds to use for the exceedance computation. These expressions are parsed by the `ExprTk` library [Par99], which can deal with nonlinear functions and complex logical statements. As a simple example,

```
"exceedance" = ["x0 + t*x1 <= 0.0", "x0 > 0.5 and x1^2 < 1.0"]
```

will compute the exceedance of the approximated probability density function for the two criteria defined in the vector.

The computed statistics are stored in files built upon the `"filename"` parameter of the `"output"` field (see Section 5.2.1). Yet the file extensions differ: the moments are stored in `.stats` files, while the exceedance is stored in `.proba` files. These files have a simple structure, described in Section 5.3.2 and Section 5.3.3 respectively.

5.2.7 Probing

A rudimentary probing system is implemented in the solver. It can be used to visualize the evolution of different quantities for real-time monitoring purposes. Additionally, the system proposes to automatically create a video from the monitored data. Concretely, the solver calls a `Python` script, `monitor.py`, through a pipeline. Then, the solver generates `.monitor` temporary files, which are continuously read and deleted by the `monitor.py` script. If needed, the script converts the monitored data into a video file thanks to `ffmpeg`. The probing system is defined in the `"probing"` field.

```
"probing": {
  "data"    : [...],
  "save"    :  ... ,
  "fps"     :  ... ,
  "origin"  : [...],
  "size"    : [...],
  "spacing" :  ...
}
```

The `"data"` parameter is a vector of strings containing the identifiers of the quantities to monitor. These identifiers can be found in the raw data `.csv` exports if needed. A monitoring window is automatically opened for each quantity to monitor. The `"save"` parameter is a boolean value that indicates whether the monitored data should be saved as a video file, and the `"fps"` parameter indicates the number of frames per second for the video in question.

The three last parameters, `"origin"`, `"size"` and `"spacing"`, are similar to the ones defined in the `"particles"` field of Section 5.2.3. The only difference is that the probing particles do not move through time, and are only used to sample the quantity of interest. The probing particles are generated at the beginning of the simulation, and their initial probability density is set to zero.

5.2.8 Dynamics

The expressions of the drift and diffusion terms are defined in the `"dynamics"` field. They characterize how particles behave individually and interact with each other, respectively.

```
"dynamics": {
  "drift"     : [...],
  "diffusion" : [...]
}
```

The `"drift"` parameter is a vector of strings containing the expression of the drift term $\underline{f}(\underline{X}, t)$. The diffusion parameter, `"diffusion"`, is defined in the same way, but in matrix form (i.e., a vector of vectors). In both cases, the expressions are parsed by the `ExprTk` library [Par99], which is used to transform the strings into callable functions. In addition to parsing general nonlinear expressions, the library is also used for computing the divergence of the diffusion term in Equation 98. For the record, if an entry of the drift vector or diffusion matrix is a simple constant value, it is advised to directly indicate the value instead of a string, in order to avoid the overhead caused by the parsing and the repeated function evaluations.

Caution

The $\underline{\underline{D}}$ matrix is expected, not the $\underline{\underline{g}}$ matrix.

5.3 Output files

The FPK-SPH solver offers several mechanisms for extracting useful data, as explained in the previous section. Mainly, the data are exported to several types of files, which are named with the output file name (defined in Section 5.2.1) to which the time step is appended. For instance, `mytest0001000` refers to the time step 1000 of a simulation for which the output file name is set to `mytest`. Finally, a specific file extension is appended, depending on the type of content stored within the file. In the following, the content associated to each output extension is given.

5.3.1 Default output (`.csv`)

The `.csv` files contain the most general form of information regarding the simulation. Any information provided by the other types of output files can be derived from the `.csv` files through post-processing, but this can be computationally intensive. The `.csv` files are organized as follows.

```
step <time step k>
time <time t>
nDimensions <number n of dimensions>
nParticles <number N of SPH particles>
volume <volume V of the particles (same for all)>
kernel <kernel name>
density,smoothingLength,densityGrad0,...,densityGradn,flow0,...,flown,state0,...,staten
   .          .              .                .          .         .     .          .
   .          .              .                .          .         .     .          .
   .          .              .                .          .         .     .          .
```

where the text between angle brackets is replaced by the content in question. The first six lines are referred to as *metadata* and don't provide any information regarding the state of the SPH particles. For the sake of clarity, let us simply say that the kernel name is of the types allowed in Section 5.2.4. The second part of the `.csv` file contains all the useful particle-related information. It is organized as a matrix of shape $(N \times (2 + 3n))$ in such a way that each row i corresponds to the particle i and contains

- the particle's (constant) density $\psi_i = \Psi_i / V_i$,
- its smoothing length h_i,
- the n components of the density gradient $\nabla\psi(\underline{X}_i, t)$ evaluated at the particle's state,
- the n components of the velocity $\underline{v}(\underline{X}_i, t)$ of the particle,
- the n components of the state $\underline{X}_i$ of the particle.

5.3.2 Statistical moments (`.stats`)

The raw statistical moments can be computed by the solver itself, without requiring any potentially time-consuming post-processing. The moments are computed according to Equation 126 in which the a_i exponents are passed through the `"moments"` list of Section 5.2.6. Assuming that one wants to compute the moments corresponding to the exponents `[0, 0, 1]`, `[1, 0, 2]` and `[2, 4, 1]`, the `.stats` output files have the following layout.

```
step <time step k>
time <time t>
nDimensions <number n of dimensions>
[0,0,1] <moment E[X_2]>
[1,0,2] <moment E[X_0 * X_2^2]>
[2,4,1] <moment E[X_0^2 * X_1^4 * X_2]>
```

5.3.3 Probability of exceedance (`.proba`)

For each expression passed into the `"exceedance"` variable, the corresponding probability of exceedance is computed by the solver at runtime, thanks to Equation 129. The results are stored in `.proba` files at each export time step. Considering that three conditions `c1`, `c2` and `c3` are inputted, the content of a `.proba` file is given hereunder.

```
step <time step k>
time <time t>
<c1> <probability of exceedance of c1>
<c2> <probability of exceedance of c2>
<c3> <probability of exceedance of c3>
```

5.4 General flowchart

In this section, a global overview of the implementation of the solver is presented. The flowchart in Figure 104 is used to illustrate each part of the algorithm.

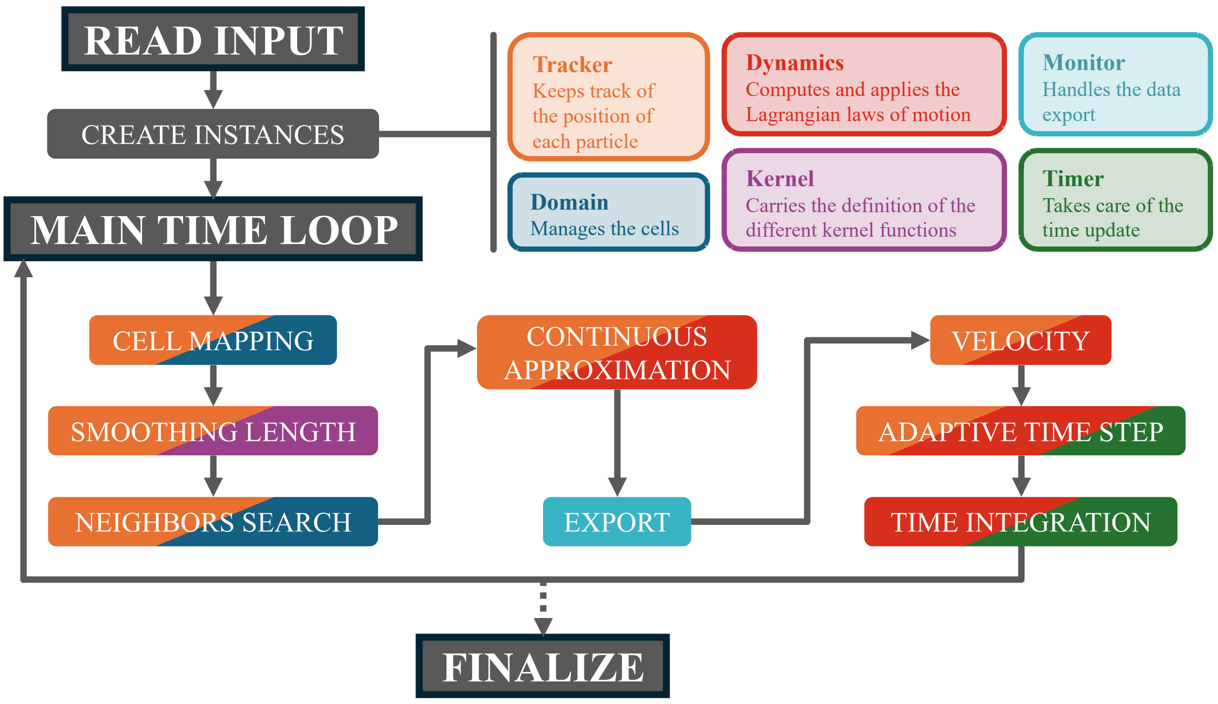


Figure 104: Flowchart of the solver, with each step color-coded by the primary C++ objects used. The C++ objects are indicated on the top right of the figure.

5.4.1 Initialization

The first step of the solver consists in instantiating the objects which drive the main update loop. This instantiation includes the definition of attributes and methods, some of which are only known at run time through the input `.json` file (see Section 5.2). Having variables and, more importantly, functions which are not known at compile time heavily impacts the performance of the solver, but offers a greater flexibility for the end-user. In the following, the initialization step is described for each primary C++ object. It is worth noting that the implementation privileges structures of arrays over arrays of structures, in order to favor the coalescence of the memory accessed by the different execution threads. Additionally, these arrays are initialized once and for all during the initialization phase and act as buffers, which removes the overhead caused by repeated dynamic memory allocation.

The `Tracker` object is the most versatile. At the initialization step, it creates the SPH particles by uniformly covering the hypercube defined in Section 5.2.3. The same thing is done for the probe particles, following what is explained in Section 5.2.7. Depending on the input `.json` file, the iterative solver used for the smoothing length (see Section 5.2.4) is set, as well as the time integration method (see Section 3.6.1 and Section 3.6.2). Similarly, the exceedance functions passed to the input file (see Section 5.2.6) are parsed by the `ExprTk` library [Par99] to be interpreted numerically.

The `Dynamics` object then initializes the probability mass Ψ_i associated to each particle i according to Section 3.4 and discards the massless particles. The drift vector field $\underline{f}(\underline{X}, t)$ and diffusion tensor field $\underline{\underline{D}}(\underline{X}, t)$ are also encoded. Interpreting the expression of the initial condition, the drift and the diffusion relies once again on the `ExprTk` parsing library [Par99].

Depending on the domain type, as defined in Section 3.9, the `Domain` object is instantiated from a specialized subclass, which gives an appropriate definition of the cell indexing and distance computation.

In a similar fashion, the `Kernel` object derives from specialized classes which each define a kernel and its derivative for the given number of dimensions, as explained in Section 3.3.

The `Monitor` object simply launches an external Python script through a communication pipeline if and only if the monitoring feature is enabled (see Section 5.2.7). This script is responsible for displaying the real-time evolution of the solution via the `pyplot.matplotlib` package.

At last, the `Timer` object sets the time step to be either fixed or adaptive, based on the definition given in Section 3.6.3.

Before entering the loop, the `OpenMP` threads are spawned [Ope24], thus avoiding the overhead caused by repeatedly spawning threads. Once the threads are spawned, one must keep in mind that, by default, all the instructions of the loop are read by all the threads at the same time. Let us stress that the parallelization is not applied to the outer time loop itself but rather over the inner loops nested within it. In the following, loops benefitting from multi-core parallelism are indicated with the **p-** prefix (e.g., **p-for**).

5.4.2 Main time loop

At its core, the SPH method relies on the identification of the neighbors of each particle. The first step for doing so consists in assigning each particle to the cell it belongs to, as explained in Section 3.8.

```
CELL MAPPING
p-empty the cells
p-for each particle $i$
    compute the cell coordinates of particle $i$
    assign particle $i$ to the cell it belongs to
```

Algorithm 2: Detailed version of the cell mapping algorithm.

The computation of cell coordinates involves several mathematical operations that depend on the user-defined domain type, as detailed in Section 3.9. This task is assigned to an object instantiated from one of three classes (`Bounded`, `Periodic` or `Infinite`), depending on the type of domain. In an early version of the solver, a single function was responsible for the task. This came with a major drawback as a conditional statement regarding the type of domain was evaluated at each iteration of the solver, for each particle. The current design addresses this inefficiency by distributing the different scenarios among dedicated objects. As a result, the domain type is evaluated only once at initialization, and the appropriate object is instantiated, eliminating the need for repetitive conditional checks during runtime. This object-oriented mechanism is applied in multiple parts of the code; it is particularly important given that all the conditional statements are evaluated at runtime and cannot be optimized by the compiler. Another option could have been to hard-code the problem inputs, but this option is disregarded in favor of a more flexible framework that allows the user to define problem settings dynamically.

When the cell occupied by a particle is identified, the particle index i is stored in a vector buffer of size `nCells` $\times$ `maxParticlesInCell`, where `nCells` is computed based on the geometrical definition of the domain (see Section 5.2.5).

Next, the smoothing length of the particles is updated, if needed. Similarly to the object-oriented approach which suggests to split the different scenarios into different C++ objects, the smoothing length update function is defined once and for all at the beginning of the solver, depending on the user inputs.

```
SMOOTHING LENGTH UPDATE
p-until convergence is met for all particles
or the maximum number of iterations is reached
    call NEIGHBOR PAIRING (see Algorithm 4)
    call CONTINUOUS APPROXIMATION (see Algorithm 5)
    apply one iteration of iterative solver (see Section 3.7.4.1 and Section 3.7.4.2)
if the discard mode is enabled (see Section 5.2.4)
    discard the particles which have not converged
    p-normalize the total probability mass: $\Psi_i^{[k]} \leftarrow \Psi_i^{[k-1]} / \sum_{j=1}^{N} \Psi_j^{[k-1]}$
```

Algorithm 3: Smoothing length update algorithm.

As explained earlier, convergence is expected to be met for *all* SPH particles when their respective smoothing lengths cause a relative error smaller than 10^{-3} [-]. To benefit from multi-core parallelization while checking the validity of a criterion across all the particles, an `OpenMP reduction` clause [Ope24] is employed. Depending on the user's choices, particles which have not converged after a maximum number of iterations are discarded. This passes by the identification of such particles and their removal from all the data structures in which they appear, as well as a reindexing of the remaining particles. Since deleting particles results in a loss of total probability mass, the weights of the remaining particles must be rescaled to restore total probability unity. By experience, most applications seem to work best for a small number of iterations and no particle discarding. Besides, note that the neighbor pairing algorithm is called at each iteration; this is not the case of the cell mapping algorithm since the particles do not move from one iteration to the next.

Regardless of whether the adaptive smoothing length algorithm is applied, the neighbor pairing is called at least once in the main time loop.

```
NEIGHBOR PAIRING
p-reset the set of neighbors of each particle to void
p-for each particle $i$ not being a probing particle
    compute the number of layers $\lambda_i^{[k]}$
        compute the target number of layers: $\overline{\lambda}_i^{[k]} \leftarrow \left\lceil h_i^{[k]} / \left(\alpha h_i^{[0]}\right) \right\rceil$ (see Equation 123)
        limit the number of layers: $\lambda_i^{[k]} \leftarrow \min\left(\overline{\lambda}_i^{[k]}, \lambda_{\max}\right)$ (see Section 3.9.4)
    for each neighbor cell $c$ in the $\lambda_i^{[k]}$ layers
        for each particle $j$ in cell $c$
            if particle $j$ is a probing particle
                add particle $i$ to the list of neighbors of particle $j$
            else, i.e., particle $j$ is a regular particle
                if $\|\underline{X}_j^{[k]} - \underline{X}_i^{[k]}\| < \kappa h_i^{[k]}$
                    add particle $j$ to the list of neighbors of particle $i$
                    if $\|\underline{X}_j^{[k]} - \underline{X}_i^{[k]}\| \geq \kappa h_j^{[k]}$
                        add particle $i$ to the list of neighbors of particle $j$
```

Algorithm 4: Detailed version of the neighbor pairing algorithm.

At first glance, this algorithm may seem overly intricate for the seemingly simple task it performs: identifying pairs of particles that lie within each other's support. However, this algorithmic design is optimized to avoid data races and contention. Specifically, the present structure enabled the use of `OpenMP atomic capture` clauses, which are more efficient than the original `OpenMP critical` clauses [Ope24].

At this stage, each particle "knows" the set of particles with which it will interact and the PDF can be sampled at SPH particle locations using the SPH continuous approximation, as it is done in Algorithm 5. The same is done for the continuous approximation of the gradient of the PDF in Algorithm 6.

```
DENSITY CONTINUOUS APPROXIMATION
p-reset the sampled probability density to zero
p-for each particle $i$
    for each neighbor particle $j$
        compute their relative distance
        compute the contribution of $j$ to the value sampled at $i$: $\tilde{\psi}_{ij}^{[k]} \leftarrow \Psi_i^{[k]} W_{ij}^{[k]}$
        increment the sampled probability density by this value: $\tilde{\psi}_i^{[k]} \leftarrow \tilde{\psi}_i^{[k]} + \tilde{\psi}_{ij}^{[k]}$
# In the end, $\left\langle \psi\left(\underline{X}_i^{[k]}, t^{[k]}\right)\right\rangle \equiv \tilde{\psi}_i^{[k]} = \sum_{j \in \mathcal{N}_i^{[k]}} \Psi_j^{[k]} W_{ij}^{[k]}$, as in Equation 111
```

Algorithm 5: Continuous approximation algorithm for the PDF.

```
DENSITY GRADIENT CONTINUOUS APPROXIMATION
p-reset the sampled probability density derivatives to zero
p-for each particle $i$
    for each neighbor particle $j$
        compute their relative distance
        compute the contribution of $j$ to the value sampled at $i$: $\nabla\tilde{\psi}_{ij}^{[k]} \leftarrow \Psi_i^{[k]} \nabla W_{ij}^{[k]}$
        increment the sampled gradient by this value: $\nabla\tilde{\psi}_i^{[k]} \leftarrow \nabla\tilde{\psi}_i^{[k]} + \nabla\tilde{\psi}_{ij}^{[k]}$
# In the end, $\nabla\left\langle \psi\left(\underline{X}_i^{[k]}, t^{[k]}\right)\right\rangle \equiv \nabla\tilde{\psi}_i^{[k]} = \sum_{j \in \mathcal{N}_i^{[k]}} \Psi_j^{[k]} \nabla W_{ij}^{[k]}$, as in Equation 111
```

Algorithm 6: Continuous approximation algorithm for the gradient of the PDF.

For the last two snippets, the kernel functions are evaluated thanks to C++ objects which are instantiated once and for all. This is similar to what is done for the type of domain.

Up to this point, all the information needed to update the state of the SPH particles is known. However, one wants to export the data relative to the current time step before moving to the next. For this reason, data export operations are carried out at this point of the time loop.

```
EXPORT
if the time step $k$ corresponds to an export time step
    if .csv exports are enabled
        write to a .csv file (see Section 5.3.1)
    if real-time monitoring is enabled
        write to a .monitor file the probed probability densities (see Section 5.2.7)
    if .stats exports are enabled
        for each statistical moment $m^{[k]}$
            compute $m^{[k]}$ with Equation 126
        write the results to a .stats file (see Section 5.3.2)
    if .proba exports are enabled
        for each exceedance set $\Omega^{[k]}$
            compute the exceedance value with Equation 129
        write the results to a .proba file (see Section 5.3.3)
```

Algorithm 7: Export operations.

To update the state of the SPH particles, the velocity field that the SPH particles will follow in state space is determined. More precisely, the velocity field is sampled at the SPH particle states, following Equation 98.

```
VELOCITY FIELD COMPUTATION
for each particle $i$
    compute $\underline{f}_i^{[k]} \equiv \underline{f}\left(\underline{X}_i^{[k]}, t^{[k]}\right)$ with the parser
    compute $\underline{\underline{g}}_i^{[k]} \equiv \underline{\underline{g}}\left(\underline{X}_i^{[k]}, t^{[k]}\right)$ with the parser
    compute $\overline{\nabla \cdot \underline{\underline{g}}_i^{[k]}} \equiv \nabla \cdot \underline{\underline{g}}\left(\underline{X}_i^{[k]}, t^{[k]}\right)$ with the parser
    compute $\underline{v}_i^{[k]} \leftarrow \underline{f}_i^{[k]} - \overline{\nabla \cdot \underline{\underline{g}}_i^{[k]}} - \underline{\underline{g}}_i^{[k]} \nabla \tilde{\psi}_i^{[k]} / \tilde{\psi}_i^{[k]}$, using Equation 98
```

Algorithm 8: Velocity field computation algorithm.

As shown in Section 5.5, this part of the solver does not demand significant computational effort. Consequently, it is deliberately left un-parallelized, which helps reduce memory usage.

The state of the SPH particles is updated by numerical time integration, using a finite time step Δt determined by the algorithm hereunder. As a reminder, two conventions, namely the absolute and relative conventions, have been studied for determining the optimal integration time step. The following piece of algorithm uses the relative convention as it is the one kept in the final implementation.

```
ADAPTIVE TIME STEP COMPUTATION
if adaptive time stepping is enabled
    set the current target time step $\overline{\Delta t}^{[k]} \leftarrow +\infty$
    p-for each particle $i$
        for each particle $j \neq i$
            compute the distance $r_{ij}$ between $i$ and $j$
            compute the norm of the relative velocity $\|\underline{v}_{ij}\|$ between $i$ and $j$
            update the current target time step $\overline{\Delta t}^{[k]} \leftarrow \min\left(\overline{\Delta t}^{[k]}, r_{ij}/\|\underline{v}_{ij}\|\right)$
    limit to the maximum time step: $\Delta t^{[k]} \leftarrow \min\left(\overline{\Delta t}^{[k]}, \Delta t_{\max}\right)$
```

Algorithm 9: Adaptive time step computation algorithm.

Depending on user-defined choices, the adaptive time stepping feature can be enabled and disabled at will. To avoid unnecessary evaluations of conditional statements, the adaptive time stepping function is determined during the initialization phase of the solver. Regarding multi-core parallelization, it is worth noting that an `OpenMP reduction` clause is used to keep the smallest, most restrictive time step encountered in all the threads.

At last, the state of the SPH particles is finally updated through numerical time integration.

```
TIME INTEGRATION
if the Euler integration method (see Section 3.6.1) is chosen
    p-for each particle $i$
        update the state: $\underline{X}_i^{[k+1]} \leftarrow \underline{X}_i^{[k]} + \Delta t^{[k]} \underline{v}_i^{[k]}$
else if the midpoint integration method (see Section 3.6.2) is chosen
    p-for each particle $i$
        apply a half Euler step $\underline{X}_i^{[k+1/2]} \leftarrow \underline{X}_i^{[k]} + 0.5\Delta t^{[k]} \underline{v}_i^{[k]}$
    compute the quantities corresponding to this half time step:
        call CELL MAPPING
        call SMOOTHING LENGTH UPDATE
        call NEIGHBOR PAIRING
        call DENSITY CONTINUOUS APPROXIMATION
        call DENSITY GRADIENT CONTINUOUS APPROXIMATION
        call VELOCITY FIELD COMPUTATION
    p-for each particle $i$
        update the state: $\underline{X}_i^{[k+1]} \leftarrow \underline{X}_i^{[k]} + \Delta t^{[k]} \underline{v}_i^{[k+1/2]}$
increment the simulated time $t^{[k+1]} \leftarrow t^{[k]} + \Delta t^{[k]}$ and time step $k \leftarrow k + 1$
```

Algorithm 10: Time integration algorithm.

Here again, two behaviors are expected depending on the choice of numerical integration method. Therefore, the numerical time integration method is defined only once at the beginning of the solver's execution.

5.5 Profiling

To identify the main bottlenecks of the solver's sequential implementation, the GNU `gprof` [GNU25] is employed. It has proven valuable in guiding development efforts by highlighting the most computationally expensive parts of the code. In Figure 105, the profiling tool is run on the last version of the solver to point out the parts of the code that consume the most computational power and that should be targeted in future work.

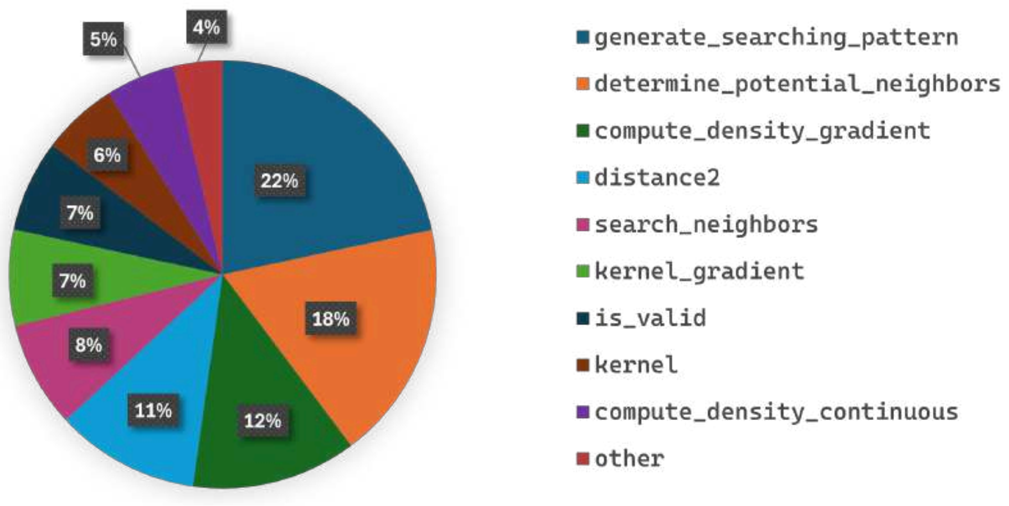


Figure 105: `gprof` profiling ran on the latest version of the code, using the same test case as for the scaling analysis (see Appendix B).

Let us discuss the impact of the functions causing the main overhead. As expected, the *total* neighbor identification process is the most resource-consuming, representing from ~ 55 [%] to ~ 65 [%] of the overall workload. This process includes

- `generate_searching_pattern`, charged of identifying the indices of the cells belonging to the λ layers of a given cell, in n dimensions,
- `is_valid`, used to filter out the redundant neighbor cells,
- `determine_potential_neighbors`, taking care of identifying all the particles belonging to the valid cells,
- a fraction of the calls to `distance2`, to compute the distance between pairs of particles,
- `search_neighbors`, orchestrating these routines.

Aside from that, the sampling of ψ and $\nabla\psi$ at the SPH particle positions make up for the remaining computational load. The former includes

- `kernel`, which computes the value of the user-defined kernel, and
- `compute_density`, which applies Equation 111,

while the latter includes

- `kernel_gradient`, which computes the kernel gradients thanks to Equation 90, and
- `compute_density_gradient`, which applies Equation 112.

Besides, they are both responsible for a fraction of the calls to `distance2`.

5.6 Multi-core parallelization

The entire numerical solver is parallelized using `OpenMP` [Ope24], a shared memory parallelization library. The parallelization is done at the level of the particles, which are distributed among the available cores. To assess quantitatively the numerical quality of the multi-core parallelization, several approaches exist. In the present case, one proposes to study the *strong scaling* of the numerical implementation, that is how the solver adapts to an increase of cores for a fixed problem size. More precisely, the unimodal linear Duffing model, introduced in Section 4.2, is considered on a time horizon reduced to $T = 3$ [s], in order for the simulations to run faster, as the steady state regime is no longer a point of interest. The integration time step is also kept constant with $\Delta t = 5 \times 10^{-3}$ [s]. To guarantee that the scaling analysis focuses on the computational performance of the algorithm, the data exports are disabled. The complete input file is presented in Appendix B for reference.

Let us consider that the solver is made of a fraction $p \in [0, 1]$ of perfectly parallelized code. In such a case, one can express the overall runtime as

$$T_c(p) = (1-p)T_1 + p\frac{T_1}{c}, \tag{178}$$

where c is the number of cores and T_1 is the serial runtime. Consequently, by definition of speedup, one gets *Amdahl's* law [PH11]:

$$S_c^{\mathrm{A}}(p) \equiv \frac{T_1}{T_c(p)} = \frac{1}{1-p+\dfrac{p}{c}}. \tag{179}$$

By fitting Amdahl's law, one can thus estimate the fraction of code that is effectively parallelized. This is done in Figure 106 and results in a very good parallelization factor $p \approx 97.6$ [%].

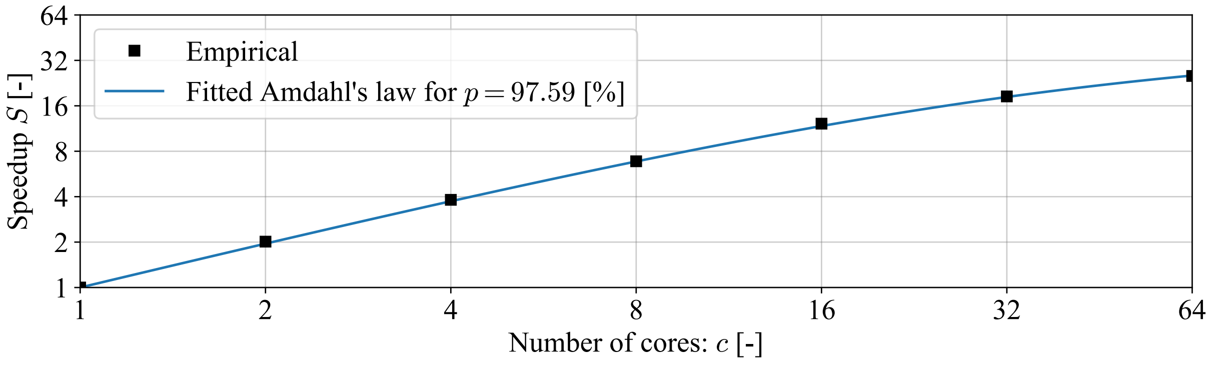


Figure 106: Scalability analysis performed for strong scaling. Amdahl's law is fitted on the solver's average speedup over 5 runs.

Whereas strong scaling evaluates how computational performance improves with an increasing number of processing cores for a fixed problem size, *weak scaling* examines how efficiently a solver maintains performance as both the problem size and the number of computational resources grow proportionally. In such analyses, Gustafson’s law is often used to characterize expected speedup. However, weak scaling is not considered in the present work due to the inherent ambiguity in defining problem size within the context of the SPH method. Specifically, increasing the number of particles does not result in a linear or straightforward increase in computational complexity, as the density and nature of inter-particle interactions evolve in a nonlinear fashion. As such, establishing a meaningful metric for scaling becomes nontrivial and would require a more nuanced formulation, which falls outside the scope of this study.

6 Conclusion and perspectives

This work builds upon previous exploratory efforts to apply the SPH method for solving the FPK equation, extending both the theoretical foundations and the practical implementation of the approach.

A central objective of the thesis was to make the FPK-SPH framework accessible and educational. To that end, the report takes care to progressively introduce the reader to the stochastic context: beginning with deterministic systems, moving to SDEs, and ultimately connecting to their associated FPK equations.

A similar progression is followed for the SPH method, starting from its theoretical formulation and leading up to its numerical implementation, with an emphasis on both standard algorithmic techniques and the practical features developed in this work. In parallel, the method is shown to possess inherent properties that make it particularly suitable for solving the FPK equation, as it naturally ensures positivity, vanishing behavior at infinity, and conservation of total probability. Another inherent property of the method is the ability of the SPH particles to automatically focus on the zones of interest, where the probability density is highest, while traditional mesh-based techniques would require to mesh an entire region of state space. Additionally, as illustrated by [DCB22], the FPK-SPH method still offers considerable potential for improvement, as it remains relatively uncharted in the current literature.

The outcome is a flexible, open-source solver that implements the combined FPK-SPH approach. The solver leverages modern numerical techniques, including an efficient multi-core parallelization. A detailed overview of the algorithm is presented, together with a practical guide to support its utilization. These components aim to assist and promote further research within the field. The implementation has been benchmarked on both linear and nonlinear academic test cases. To the best of our knowledge, this is the first solver capable of handling an arbitrary number of dimensions and it demonstrated to run effectively on a six-dimensional problem. However, limited computational resources constrain the current study to a coarse approximation of the solution. The main algorithmic limitation arises from the high connectivity of particles in high-dimensional state spaces, and must be addressed by a more efficient neighbor search on a manifold. The current implementation would also benefit significantly from memory optimization and refactoring using modern C++ practices. Future work may also explore GPU acceleration, which has already proven very effective for traditional SPH simulations.

Compared to the traditional MCS method, which is often default for solving stochastic problems, the SPH approach offers several key advantages. Most significantly, the SPH method *solves* for continuous probability distributions, whereas the MCS method relies on discrete realizations upon which to *infer*. Converting these realizations into a continuous distribution typically requires introducing additional assumptions, such as choosing an appropriate binning strategy or presupposing the underlying distribution's shape.

Another important advantage of the SPH method, which warrants further investigation, concerns its performance when the evaluation of the drift and diffusion functions is computationally expensive. In such cases, the efficiency of the MCS method can deteriorate significantly due to its reliance on a large number of samples. By contrast, the SPH method typically requires fewer particles and is primarily limited by the efficiency of the neighbor search algorithm, making it a potentially more suitable approach for problems involving expensive model evaluations.

Nonetheless, the MCS method remains remarkably practical for general-purpose engineering problems due to its simpler implementation, relatively low computational cost, and natural integration into massively parallel computing environments. A thorough comparison between SPH, MCS, and more classical approaches remains an open and valuable direction for future research.

If you have made it up to here, congratulations!
I genuinely hope you enjoyed reading this work and I thank you for your consideration.

A [Appendix] Conventions

1.1 Notations

Symbol	Description
$\mathrm{d}_t = \frac{\mathrm{d}}{\mathrm{d}t}$	Total derivative with respect to t
$\partial_t = \frac{\partial}{\partial t}$	Partial derivative with respect to t
$\partial_i = \frac{\partial}{\partial X_i}$	Partial derivative with respect to X_i
$\partial_{ij\dots} = \partial_i \partial_j \cdots$	Combined partial derivatives
$(\cdot)'$	First derivative of a univariate function
$A^{(n)}$	n^{th} distributional derivative of A (univariate)
A	A scalar
$\underline{A}$	A vector
$\underline{\underline{A}}$	A matrix
$A(\cdot)$	A scalar-valued function
$A[\cdot]$	A scalar-valued functional
$[\cdot]$	Units ([-] = dimensionless)
$A^{[k]}$	Value of A at iteration k
$A^{[k,l]}$	Value of A at iteration k and substep l
$[\underline{A}]_i = A_i$	Component i of vector $\underline{A}$
$[\underline{\underline{A}}]_{ij}$	Entry (i, j) of the matrix $\underline{\underline{A}}$
$\lVert\underline{A}\rVert$	The L_2 norm of $\underline{A}$: $\sqrt{\sum_{i=1}^{n} \lvert A_i \rvert^2}$
$\langle A \rangle$	The continuous approximation of A
$\{1, \dots, n\}$	Shorthand for $\mathbb{Z} \cap [1; n]$
$\{A_i\}_{i=1}^{n}$	Shorthand for $\{A_i : i \in \{1, \dots, n\}\}$
$\sum_{i,j,\dots=1}^{n}$	Shorthand for $\sum_{i=1}^{n} \sum_{j=1}^{n} \cdots$
$\sim A'$	Approximately A'
$A \sim A'$	A equals A' up to a multiplicative constant
$A \approx A'$	A is approximately equal to A'
$\overset{\text{i.i.d.}}{\sim}$	Independently and identically distributed
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ
$\operatorname{sgn}(\cdot)$	The sign function
$\operatorname{erf}(\cdot)$	The error function
$\mathcal{O}(\cdot)$	At most of the order of
$\delta(\cdot)$	Dirac delta distribution
δ_{ij}	Kronecker delta
n	The number of dimensions
$\underline{\underline{I}}_n$	The identity matrix in $\mathbb{R}^{n \times n}$
$\underline{X}$	The state vector
$\underline{x}$	The position vector
t	The time
T	The time horizon
V	The (abstract) volume
$\underline{f}(\underline{X}, t)$	The drift vector
$\underline{\gamma}(\underline{X}, t)$	The vector noise source term
$\underline{\underline{g}}(\underline{X}, t)$	The noise amplification matrix
$\underline{\underline{D}}(\underline{X}, t)$	The diffusion matrix
$\underline{w}(t)$	The Delta-correlated Gaussian White Noise (DGWN) vector
$\underline{W}(t)$	Vector of independent standard Wiener processes
$\underline{v}(\underline{X}, t)$	The velocity field
$\psi(\underline{X}, t)$	The Probability Density Function (PDF)
$\psi(\underline{X}, t \mid \underline{X}', t')$	The conditional PDF of $\underline{X}$ at t, given $\underline{X}'$ at t'
Ψ	The probability mass
$P(\cdot)$	The probability function
$E[\cdot]$	The expectation functional
$m(a_1, \dots, a_n)$	The raw statistical moment
$\overline{m}(a_1, \dots, a_n)$	The central statistical moment
N	The number of SPH particles
$\mathcal{N}_i$	The set of neighbor SPH particles of particle i
C	A normalization factor
W	The kernel function
ξ	The dimensionless kernel variable
s	The initial spacing between particles
η	The coupling factor
h	The smoothing length
κ	The (support) scale factor
α	The cell scale factor
S	The CFL-like stability coefficient
ε	The residual
λ	The number of layers
$\nu(\underline{X}, t)$	The threshold function
$\Omega(t)$	A portion of state space
$\mathcal{I}(t)$	The interval of confidence
$Z(m)$	The statistical pivot
$\hat{Z}(m)$	The normalized statistical pivot
$\varphi(\underline{X})$	A scalar function of $\underline{X}$
$\mathcal{E}[\cdot]$	The absolute Steady State Relative Error (SSRE)

1.2 Environments

Core Formula

This environment is used to highlight important mathematical expressions.

Core Idea

This environment is used to highlight important concepts.

Caution

This environment is used to highlight pitfalls and misconceptions.

1.3 Acronyms

The acronyms appearing in the main text are dynamically linked to this table to facilitate readability.

CEM :	Cumulant Equation Method	**ODE :**	Ordinary Differential Equation
CFL :	Courant-Friedrichs-Lewy	**PDE :**	Partial Differential Equation
DGWN :	Delta-correlated Gaussian White Noise	**PDF :**	Probability Density Function
FDM :	Finite Difference Method	**RHS :**	Right Hand Side
FEM :	Finite Element Method	**SDE :**	Stochastic Differential Equation
FPK :	Fokker-Planck-Kolmogorov	**SIRD :**	Susceptible-Infected-Recovered-Deceased
LHS :	Left Hand Side	**SPH :**	Smoothed Particle Hydrodynamics
MCS :	Monte-Carlo Simulation	**SSRE :**	Steady State Relative Error
MEM :	Moment Equation Method		

B [Appendix] JSON input for profiling and scaling analysis

```
(
  output: (
    filename: "duffing-linear",
    csvExport: false,
    refreshRate: 100000,
  ),
  time: (
    interval: 5.0,
    stepping: (false, 0.005),
    integration: "euler",
  ),
  particles: (
    origin: (-1.5, -1.5),
    size: (3.0, 3.0),
    spacing: 0.05,
  ),
  kernel: (
    type: "gaussian",
    coupling: 1.2,
    maxNeighbors: 200,
    adaptiveSmoothing: true,
    iterativeSolver: "FP",
    iterationsMax: 1,
    discardNotConverged: false,
    initialCondition: "if(x0^2+x1^2 < (4*0.3)^2, exp(-0.5*(x0^2+x1^2)/(0.3^2)), 0.0)",
  ),
  domain: (
    nDimensions: 2,
    origin: (0.0, 0.0),
    size: (2.0, 2.0),
    type: "infinite",
    cellScale: 1.0,
    maxParticlesPerCell: 400,
  ),
  statistics: (moments: (), exceedance: ()),
  probing: (
    data: (),
    save: false,
    fps: 0,
    origin: (0.0, 0.0),
    size: (0.0, 0.0),
    spacing: 0.0,
  ),
  dynamics: (
    drift: ("x1", "-2*0.2*1.0*x1 - 1.0*x0 - 0.0*1.0*x0*x0*x0"),
    diffusion: ((0.0, 0.0), (0.0, 0.5)),
  ),
)
```

C [Appendix] Policy on the use of artificial intelligence

The University of Liège promotes transparent usage of artificial intelligence, such as generative models. In accordance with these guidelines, we briefly inform the reader of the use of such tools in the preparation of this work.

C.1 ChatGPT

The *ChatGPT-4o* model offered by OpenAI was employed to assist with various tasks, including:

- clarifying modern C++ concepts,
- addressing questions related to the development environment (e.g., Git, CMake, and Bash commands),
- improving the quality of English in the report,
- helping reformulate ideas into clearer and more structured forms.

C.2 GitHub Copilot

The GitHub Copilot assistant, integrated within the Visual Studio Code environment, was employed to enhance productivity during code development. Specifically, its use was limited to code completion, leveraging the *Gemini 2.0 Flash* model. Additionally, assistance in debugging was sought using *Gemini 2.0 Flash*, *Claude 3.5 Sonnet*, and *Claude 3.7 Sonnet (Thinking)*. These tools were not used to autonomously generate complete blocks of code.

C.3 SciSpace

The latest version of *SciSpace*, an AI-powered tool designed for academic research, was also used to address more theoretical questions, notably by providing relevant academic references to support its answers.

Bibliography

[ML85] J. Monaghan and J. Lattanzio, “A refined particle method for astrophysical problems,” vol. 149, no. 1, pp. 135–143, Aug. 1985.

[Luc77] L. B. Lucy, “A numerical approach to the testing of the fission hypothesis,” *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, vol. 82, pp. 1013–1024, 1977.

[Can14] T. Canor, “New perspectives on probabilistic methods for nonlinear transient dynamics in civil engineering,” 2014.

[Gof13] L. Goffin, “Development of a didactic SPH model,” 2013.

[BBD24] T. Bertrand, C. Borbouse, and F. Distrée, “Smoothed-particle hydrodynamics for microfluidics — MATH0471 - Multiphysics integrated computational project,” University of Liège, 2024.

[BAB02] R. C. Bishop, H. Atmanspacher, and R. Bishop, “Deterministic and indeterministic descriptions,” *Between chance and choice: interdisciplinary perspectives on determinism*, pp. 5–31, 2002.

[LaP14] P. S. LaPlace, *Essai philosophique sur les probabilités.* Paris: Courcier, 1814. Accessed: May 11, 2025. [Online]. Available: http://eudml.org/doc/203193

[Ear86] J. Earman, *A Primer on Determinism.* D. Reidel, 1986.

[Ris89] H. Risken, *The Fokker-Planck Equation: Methods of Solution and Applications Second Edition.* Springer, 1989.

[Hid03] T. Hida, “White Noise Analysis: Part I. Theory in Progress,” *Taiwanese journal of mathematics*, vol. 7, no. 4, pp. 541–556, 2003.

[Dur19] R. Durrett, *Probability: Theory and Examples*, 5th ed. in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

[Le 16] J.-F. Le Gall, *Brownian Motion, Martingales, and Stochastic Calculus*, 1st ed. 2016. in Graduate Texts in Mathematics, 274. Cham: Springer International Publishing, 2016.

[Shr04] S. E. Shreve, *Stochastic calculus for finance 2, Continuous-time models.* New York: Springer, 2004.

[RC04] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods.* in Springer Texts in Statistics. New York, NY: Springer New York, 2004. doi: 10.1007/978-1-4757-4145-2.

[KP92] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, 1st ed., vol. 1. in Stochastic Modelling and Applied Probability, vol. 1. Springer Berlin, Heidelberg, 1992, p. XXXVI, 636. doi: 10.1007/978-3-662-12616-5.

[Pre90] A. Preumont, *Vibrations aléatoires et analyse spectrale.* EPFL Press, 1990.

[GM77] R. A. Gingold and J. J. Monaghan, “Smoothed particle hydrodynamics: theory and application to non-spherical stars,” *Monthly notices of the royal astronomical society*, vol. 181, no. 3, pp. 375–389, 1977.

[CB15] J.-S. Chen and T. Belytschko, “Meshless and Meshfree Methods,” 2015, pp. 886–894. doi: 10.1007/978-3-540-70529-1_531.

[Par62] E. Parzen, “On Estimation of a Probability Density Function and Mode,” *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[LLL03] M. Liu, G. Liu, and K. Lam, “Constructing smoothing functions in smoothed particle hydrodynamics with applications,” *Journal of Computational and Applied Mathematics*, vol. 155, no. 2, pp. 263–284, 2003, doi: 10.1016/S0377-0427(02)00869-5.

[Pri12] D. J. Price, “Smoothed particle hydrodynamics and magnetohydrodynamics,” *Journal of Computational Physics*, vol. 231, no. 3, pp. 759–794, 2012, doi: 10.1016/j.jcp.2010.12.011.

[Kos+19] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, “Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids,” *Eurographics 2019 - Tutorials*, 2019, doi: 10.2312/EGT.20191035.

[CD13] T. Canor and V. Denoël, “Transient Fokker–Planck–Kolmogorov equation solved with smoothed particle hydrodynamics method,” *International journal for numerical methods in engineering*, vol. 94, no. 6, pp. 535–553, 2013.

[Mon05] J. Monaghan, “Smoothed Particle Hydrodynamics,” *Reports on Progress in Physics*, vol. 68, p. 1703, 2005, doi: 10.1088/0034-4885/68/8/R01.

[Bak99] J. A. Baker, “Integration of Radial Functions,” *Mathematics Magazine*, vol. 72, no. 5, pp. 392–395, 1999.

[CF86] P. Combis and J. Fronteau, “A Purely Lagrangian Method for the Numerical Integration of Fokker-Planck Equations,” *Europhysics Letters*, vol. 2, no. 3, p. 227, Aug. 1986, doi: 10.1209/0295-5075/2/3/011.

[HK89] L. Hernquist and N. Katz, “TREESPH-A unification of SPH with the hierarchical tree method,” *Astrophysical Journal Supplement Series (ISSN 0067-0049)*, vol. 70, pp. 419–446, 1989.

[Fis25] S. Fisher Ronald Aylmer, “Applications of "Student's" Distribution,” *Metron*, vol. 5, pp. 90–104, 1925.

[GF14] M. Grigoriu and R. Field, “A method for analysis of linear dynamic systems driven by stationary non-Gaussian noise with applications to turbulence-induced random vibration,” *Applied Mathematical Modelling*, vol. 38, no. 1, pp. 336–354, 2014, doi: https://doi.org/10.1016/j.apm.2013.05.055.

[Lei+25] S. Lei, W. Cui, L. Patruno, S. de Miranda, L. Zhao, and Y. Ge, “State augmentation method for multimode nonstationary vibrations of long-span bridges under extreme winds,” *Engineering Structures*, vol. 332, p. 120021, 2025, doi: 10.1016/j.engstruct.2025.120021.

[Den05] V. Denoël, “Application des méthodes d'analyse stochastique à l'étude des effets du vent sur les structures du génie civil,” 2005.

[KT81] S. Karlin and H. E. Taylor, *A second course in stochastic processes*. Elsevier, 1981.

[WSB96] S. Wojtkiewicz, B. Spencer Jr, and L. Bergman, “On the cumulant-neglect closure method in stochastic dynamics,” *International journal of non-linear mechanics*, vol. 31, no. 5, pp. 657–684, 1996.

[Bra08] F. Brauer, “Compartmental models in epidemiology,” *Mathematical epidemiology*, pp. 19–79, 2008.

[CNP20] G. C. Calafiore, C. Novara, and C. Possieri, “A time-varying SIRD model for the COVID-19 contagion in Italy,” *Annual reviews in control*, vol. 50, pp. 361–372, 2020.

[Loh24] N. Lohmann, “JSON for Modern C++.” Accessed: Feb. 07, 2024. [Online]. Available: https://json.nlohmann.me/

[Par99] A. Partow, “ExprTk - C++ Mathematical Expression Toolkit Library.” Accessed: Aug. 31, 2024. [Online]. Available: https://www.partow.net/programming/exprtk/index.html

[Ope24] OpenMP Architecture Review Board, “OpenMP 6.0 API Syntax Reference Guide.” Accessed: May 20, 2025. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-RefGuide-6.0-OMP60SC24-web.pdf

[GNU25] “GNU gprof.” Accessed: May 28, 2025. [Online]. Available: https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_mono/gprof.html

[PH11] D. Patterson and J. Hennessy, *Computer Architecture: A Quantitative Approach*, 5th ed. in The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann Publishers, 2011, pp. 46–61.

[DCB22] M. Duffy, S.-J. Chung, and L. Bergman, “A general Bayesian nonlinear estimation method using resampled Smooth Particle Hydrodynamics solutions of the underlying Fokker–Planck Equation,” *International journal of non-linear mechanics.*, vol. 146, 2022.