Hybrid PDE solver for data-driven problems and modern Branching

Hybrid PDE solver for data-driven problems and modern Branching

Francisco Bernal [1,2], Gonçalo dos Reis [3,4] and Greig Smith [3]

[1] *CMAP - Centre de Mathématiques Appliquées, Ecole Polytechnique, Route de Saclay, 91128 Palaiseau Cedex, FR.*
[2] *INESC-ID\IST, TU Lisbon. Rua Alves Redol 9, 1000-029 Lisbon, PT.*
[3] *University of Edinburgh, School of Mathematics, Edinburgh, EH9 3FD, UK.*
[4] *Centro de Matemática e Aplicações (CMA), FCT, UNL, PT.*

(*Received 27th October, 2016*)

The numerical solution of large-scale PDEs–such as those naturally occurring in data-driven applications–unavoidably require powerful parallel computers and tailored parallel algorithms to make the best possible use of them. In fact, considerations about the parallelization and scalability of realistic problems are often critical enough to warrant acknowledgement already in the modelling phase. The purpose of this paper is to spread awareness of the Probabilistic Domain Decomposition (PDD) method, a fresh approach to the parallelization of PDEs with excellent scalability properties. The idea exploits the stochastic representation of the PDE via Monte Carlo sampling in combination with deterministic high-performance PDE solvers. We describe the ingredients of PDD and its range of applicability in the scope of data science. In particular, we highlight recent advances in stochastic representations for nonlinear PDEs using branching diffusions, which have significantly broadened the scope of PDD.

We envision this work as a dictionary giving large-scale PDE practitioners references on the very latest algorithms and techniques of a non-standard, yet highly parallelizable, methodology at the interface of deterministic and probabilistic numerical methods.

Keywords: Probabilistic Domain Decomposition, high-performance parallel computing, scalability, nonlinear PDEs, Marked Branching diffusions, hybrid PDE solvers, Monte-Carlo methods.

2010 AMS subject classifications:
Primary: 65C05, 65C30, Secondary: 65N55, 60H35, 91-XX, 35CXX

1 Introduction

Partial Differential Equations (PDEs) are ubiquitous in modelling. They appear in many applications such as image analysis and processing, inverse problems, shape analysis and optimization, filtering, data assimilation and optimal control[1]. They are used in Math-Biology to model population dynamics with competition[2] or growth of tumours. They are also used to model the complex dynamics of movement of persons in crowds or to model

[1] See J.A. Sethian Berkeley's Homepage
[2] See http://www.mimuw.edu.pl/ pdemb/index.php?siteID=program

(ir)rational decisions of players in games (financial or not). They feature in many complex problems in Mathematical Finance ranging from calibration, pricing and hedging of financial contracts to some problems of optimal liquidation in the high-frequency trading context. Underpinning all these applications is the necessity of solving numerically such equations either in bounded or unbounded domains.

The standard example of a Boundary Value Problem (BVP) is Laplace's equation with Dirichlet Boundary Conditions (BCs):

$$\Delta u(\mathbf{x}) = 0 \text{ if } \mathbf{x} \in \Omega \subset \mathbb{R}^d, \qquad u(\mathbf{x}) = g(\mathbf{x}) \text{ if } \mathbf{x} \in \partial\Omega. \tag{1.1}$$

The large data sets involved in realistic applications of as those cited before nearly always imply that the discretization of a BVP such (1.1) leads to algebraic systems of equations that can only be solved on a parallel computer with a large number (say $P >> 1$) of processors. Not only does parallelization require multiple processors but also parallel algorithms. The classical Schwarz's alternating method was the first and remains the paradigm of such algorithms which we refer to as "Deterministic Domain Decomposition" (DDD) [50]. While state-of-the-art DDD algorithms outperform Schwarz's alternating method in every respect, it nonetheless serves to illustrate the crucial difficulty that they all face. The idea of Schwarz's algorithm is to divide Ω into a set of P (as many as there are processors) overlapping subdomains, and have processor j solve the restriction of the PDE to the subdomain Ω_j–see Fig. 1.1. But because the solution is not known in the first place, an initial guess has to be made at start, in order to give processor j a well-posed (yet incorrect) problem. The BCs along the fictitious interfaces of Ω_j are then updated from the solution of the surrounding subdomains in an iterative way, hopefully leading to convergence. However, since the inter-processor communication involved in this updating procedure is intrinsically sequential, it will eventually set an upper limit to the scalability of the algorithm by virtue of the well known Amdahl's law. For the sake of clarity, let us describe the situation more precisely. A fully scalable algorithm would take half the time (say $T/2$) to run if the number of processors was doubled. But if there is a fraction $\nu < 1$ of the algorithm which is sequential, then the completion time will not drop below $T\nu$, regardless of how many processors are added. For instance, in Schwarz's method, if 5% (i.e. $\nu = .05$) of the execution time of one given processor was lost to waiting for the artificial BCs to be ready, then the execution time could be shortened by at most a factor of about 20–and that with no fewer that 4000 processors. In other words, Schwarz's alternating algorithm; or *any* DDD algorithm for that matter cannot exploit the full capabilities of a parallel computer due to the idle time wasted in (essential) communication. Another feature of Schwarz's alternating algorithm, shared by many other DDD methods, is that the velocity of convergence of the iterations to the solution of the problem decreases with the amount of overlap between subdomains [50]. This is undesirable, because induces a trade off between the number iterations and the number of subdomains, which is compounded in higher dimensions.

We emphasize this point further by borrowing an example given by David Keyes [3]. The Gordon Bell prizes are annually awarded to numerical schemes which achieve a breakthrough in performance when solving a realistic problem. In 1999, one such problem

[3] See http://www.mcs.anl.gov/research/projects/petsc-fun3d/Talks/bellTalk.ppt

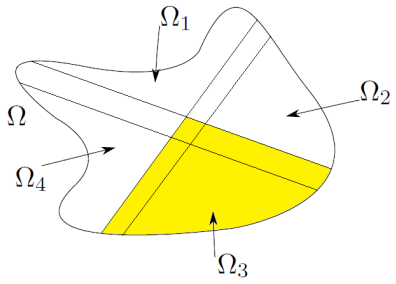

Figure 1.1. Domain decomposition on an arbitrary domain Ω, split into four overlapping subdomains Ω_i as required for Schwarz's alternating method. The subdomain Ω_3 is highlighted.

was simulating the compressible Navier-Stokes equations around a wing. With 128 processors, the winner code took 43 minutes in solving the task. On the other hand, with 3072 processors it took it 2.5 minutes instead of 1.79 as would have been the case with a fully parallelizable algorithm. The remaining 28% of computer time had been lost to interprocessor communication. At this point, adding more processors would have led to a faster loss of scalability.

Certainly, progress in DDD has not halted since 1999. In fact, the modern interpretation of DDD is in terms of preconditioning rather than the geometric picture associated with Schwarz's algorithm. However, it is also a fact that interprocessor communication is unavoidable for DDD methods, and the resulting ultimate cap to scalability is one of most serious issues faced by the mathematical modelling of many applications via PDEs. Nowadays (2016) the largest supercomputers in the world boast millions of processors rather than thousands. Besides scalability, this raises the issue of how to deal with the growing probability of processor failure during the simulation.

A conceptual breakthrough was achieved by Acebrón *et al.* with the PDD method (or rather, the PDD framework) [1], based on a previous, unpublished idea by Renato Spigler. PDD is the only domain decomposition method potentially free of communication, and thus potentially fully scalable. It does so by splitting the simulation into two separate stages, the first of which involves the non-trivial mathematical device of recasting the BVP into Monte Carlo simulations–a process which requires introducing stochastic calculus and stochastic numerics. Moreover, in addition to the kind of errors present in DDD (due to discretization), PDD results are affected by random, statistical errors. From a theoretical point of view, this is inoffensive, because statistical errors can be confined to within arbitrary confidence intervals just by taking more Monte Carlo realizations. From a practical point of view, on the other hand, both kinds of errors must be carefully balanced, which is compounded by the notoriously poor convergence rate of the statistical error (proportional to $1/\sqrt{N}$, where N is the Monte Carlo sample size).

The combination of a nontrivial, hybrid formulation with underdeveloped stochastic theory and numerics, plus the fact that the scalability advantage of PDD only starts to be telling on very large simulations, have somewhat held back PDD. Fortunately, the leap in stochastic calculus, spurred by other applications such as uncertainty quantification

and financial engineering means that the scope for PDD is widening fast. Currently, PDD approaches to linear and many nonlinear BVPs are well understood. Particularly exciting are the recent developments regarding the stochastic representations of nonlinear parabolic BVPs [47],[46],[22],[19] and more recently [34],[36],[35]. The latter encases a novel approach to branching diffusions, called *marked* and *age-marked* Branching diffusions that have expanded the scope of the methodology used to deal with the KPP equation [49],[52],[43], this particular topic is discussed below.

Outline of the paper

The remainder of the paper is organized as follows. Section 2 is an overview of PDD. The emphasis there is on the ideas rather than on technical details of stochastic calculus. In Section 3, we illustrate the the method in the context of linear elliptic PDEs with Dirichlet boundary conditions. In Section 4, we consider the parabolic PDE case for both linear and non-linear PDEs. We discuss some recent advancements in the field of stochastic analysis, which allow for general representation of these PDEs and algorithms for obtaining the Monte Carlo solution. In Section 5, we detail algorithms and show results for PDD for fully nonlinear parabolic PDEs relevant in data science. Section 6 concludes.

Remark. In this paper, we have chosen not to delve into the numerical methods needed for solving the PDD formulation of the PDE under consideration. First, because they are involved enough to distract the reader from the theoretical connection between the PDE and PDD. Second, because–as stressed throughout–those numerical methods are evolving rapidly and interested readers are directed to particular references.

2 An Overview of Probabilistic Domain Decomposition

An alternative to deterministic methods which is specifically designed to circumvent the scalability issue is the PDD method, proposed by Renato Spigler [1],[2]. First, the domain Ω under consideration is divided into non-overlapping subdomains. (Rather than into overlapping ones, such as with Schwarz's alternating algorithm discussed in the Introduction. Note that touching subdomains are fewer than overlapping ones.) PDD consists of two stages. In the first stage of PDD, the solution is calculated only on a set of interfacial nodes along the artificial interfaces, by solving the stochastic representation of the BVP with the Monte Carlo method. The stochastic representation is the crux of PDD, and can be highly non-trivial. But it can also be extremely simple, for instance in (1.1), it is well known (see [38]) that the solution u can be represented as

$$u(\mathbf{x}) = \mathbb{E}_{\mathbf{x},0}\big[\, g(X_{\tau_{\partial\Omega}})\,\big] \tag{2.1}$$

where X is the solution to the simplest Stochastic Differential Equation (SDE)

$$\text{for } t \geq 0 \quad \mathrm{d}X_t = \mathrm{d}W_t, \quad X_0 = \mathbf{x} \in \mathbb{R}^d \quad \Leftrightarrow \quad X_t = \mathbf{x} + W_t \tag{2.2}$$

where $(W_t)_{t\geq 0}$ is a d-dimensional Brownian motion; $X_{\tau_{\partial\Omega}}$ is the point on $\partial\Omega$ where the trajectory of X_t *first* hits the boundary; and lastly, $\mathbb{E}[\cdot]$ is the usual expectation operator

(in (2.1), $\mathbb{E}_{\mathbf{x},0}[\cdot]$ emphasizes that the diffusion $(X_t)_{t\geq 0}$ starts at time $t = 0$ from position $\mathbf{x}$.) See Fig.2.1 for an illustration.

The expected value above is meant to be approximated – introducing some statistical error via a Monte Carlo method, i.e. as the mean over many independent realizations of the SDE (2.2), which can be carried out independently from one another in a fully parallelizable way. A numerical scheme is required to solve the SDE, we refer to it as "stochastic numerics" [39][45].

By using the stochastic representation (2.1), one can compute the equation's solution at each point at the artificial boundaries $\Omega_i \cap \Omega_j$. It is then possible to reconstruct (approximately) the solution on the interfaces (interpolating the values at interfacial nodes with Chebyshev polynomials, say), so that the PDE restricted to each of the subdomains is now well posed. Now, the P Laplace's equations on each of the P subdomains are separate problems, and can be independently solved "deterministically"–the second stage of PDD. Note that both stages in the PDD are embarrassingly parallel by construction. We

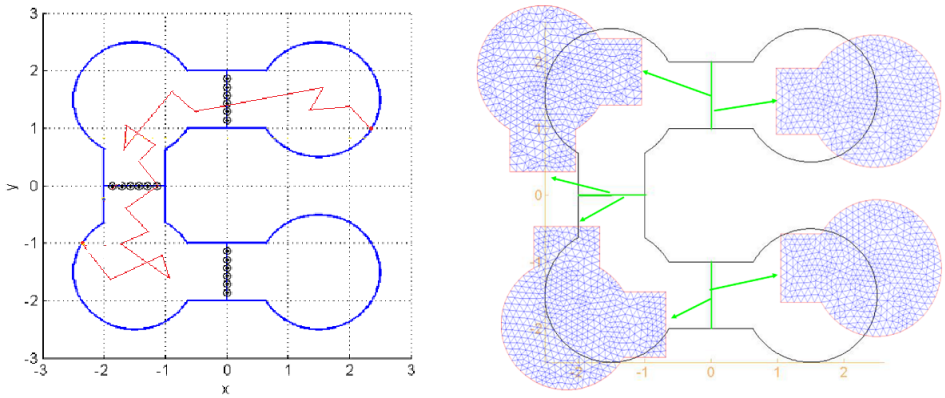

Figure 2.1. An illustration of the two stages of PDD. (Left) First, the solution is computed on each of the interface nodes (black circles) between subdomains by numerically solving an appropriate SDE, which involves many independent realizations from the same interfacial node. Two such realizations are the trajectories shown in red. (Right) After the nodal solutions are known, the artificial interfaces (green segments) are provided with a Dirichlet BC by interpolation, and the BVPs on the various subdomains are decoupled. They can then be solved independently from one another with a deterministic method, such as FEM (meshes depicted).

stress the fact that this has been made possible by the unique capability of Monte Carlo sampling of stochastic representations of PDEs of solving a BVP at an isolated point of the domain. On the other hand, DDD methods are matrix-based and deliver the solution everywhere; but by doing this are bound to move around information.

Another feature of PDD is that it is, by construction, fault-tolerant: the outage of an individual processor does not lead to substantial–let alone fatal–damage to the overall simulation. Again, this stands in stark contrast to matrix-based domain decomposition algorithms.

The ingredients of PDD

The numerical solution of a PDE on a parallel computer with PDD requires four ingredients: an interpolator, a subdomain solver, a stochastic representation of the PDE at hand, and an efficient numerical method for exploiting the latter with the Monte Carlo method. Let us briefly comment on them.

▷ The *interpolator* scheme joins the (approximate) solutions at the interface nodes (computed with Monte Carlo) in order to yield an (approximate) Dirichlet BC on the whole interface. (We remark that, in time-dependent problems, the interface runs along the time component, as well as the spatial ones.) A flexible and accurate interpolation scheme well suited to PDD is RBF interpolation–see [11] and references therein.

▷ The *subdomain solver* is any state-of-the-art numerical scheme to solve a PDE (or a BVP) on a finite domain, without parallelization. It takes the approximation to the solution on the boundary delivered by the interpolator as Dirichlet BCs and produces a solution everywhere within the subdomain. All the subdomain solutions are later "glued together" in order to have a complete solution of the large-scale original problem. Common examples of subdomain solvers are finite differences, finite elements, and meshless methods.

▷ The *stochastic representation of the PDE*, like that in (2.1), is the pointwise characterization of the solution of a PDE (stationary or time-dependent) as the expected value of the functional of a system of stochastic differential equations (SDEs). The paradigm of stochastic representations is the well-known Feynman-Kac formula for linear parabolic PDEs with Dirichlet BCs, which requires solving a single SDE. This representation was extended by Dynkin to linear elliptic BVPs. Neumann, Robin and mixed BCs can be incorporated into the formalism by appending a second SDE for the local time (see [26]). In [17] the authors deal with representations (and Monte Carlo simulation) for nonlinear divergence-form elliptic Poisson-Boltzmann PDE over the whole $\mathbb{R}^3$. Quasilinear parabolic PDEs admit a stochastic representation in terms of forward-backward SDEs (FBSDEs) [47],[46] and many extensions exist ranging from representations for obstacle PDE problems [22] to representations of fully nonlinear elliptic and parabolic PDEs [19].

In general, transport equations do not have a stochastic representation, with two important exceptions. The first exception consists in the one-dimensional telegraph equations, [3] and references therein. More importantly, the second is the Vlasov-Poisson system of equations. For such system of equations it was proposed in [44] a probabilistic representation in the Fourier space.

An important class of nonlinear PDEs for which PDD seems to be tailor made are those that can be represented by Branching processes, as introduced by [52],[49],[43]. This methodology avoids backward regressions, the so-called “Monte Carlo of Monte Carlo simulation” problem, see [34, Section 3.1]. In Section 4 below we discuss these classes of equations in more detail and give an account of recent developments.

▷ *Efficient stochastic numerical methods* to exploit the stochastic representation. With the Monte Carlo approach, the expected value which represents the solution at a point is approximated as the average over many independent samples and this has to be repeated for every interfacial node. Contrary to deterministic methods, in the stochastic

setting the errors are twofold, brought about by both discretization of time and by the substitution of the expected value by a sample mean. (One can say that the estimator of the expected value is biased due to the discretization of the time variable.) In order to attain a target accuracy a, both sources of errors must be balanced and this leads to lengthy simulations: the computational effort is $\mathcal{O}(a^{-4})$ if naive numerics are used (such as the Euler-Maruyama scheme plus boundary test for linear BVPs, see [37]). This cost can be substantially reduced (to around $\mathcal{O}(a^{-2})$ and even better) with sophisticated, novel schemes. The new integrators for bounded stopped diffusions put forward by Gobet [28] and Gobet and Menozzi [32] have doubled the convergence rate of the weak error associated to the interaction with the boundary, see also [10]. For reflected diffusions, Lèpingle's method [28] or Milstein's bounded methods [45] are recommended. Two further promising directions are the incorporation of Giles' Multilevel method [37] and the use of pathwise control variates for variance reduction [11]. Regression methods for FBSDEs are also being improved fast [29]. Good overviews for numerics of FBSDE can be found in [18], [20] and PDE inspired problems dealt by BSDEs can be found in [25], [41], [42]. In any case, the numerics of SDEs (and generalizations thereof) are underdeveloped compared to those of deterministic equations. A positive aspect is that many new ideas being currently developed for SDEs in financial mathematics can be transplanted to PDD. An important practical aspect of PDD is balancing the systematic errors incurred by the various ingredients–see [11] for details.

Ideally, a numerical method tailored to the stochastic representation of the given PDE exists, resulting in huge gains in efficiency. An example is the Walk on Spheres algorithm for the stopped Brownian motion, which is the stochastic representation of Laplace's equation with Dirichlet BCs. This and related numerical schemes should be used whenever possible (see [12] for references).

Some results with PDD

The initial undertaking of the PDD programme was due to J. A. Acebrón and his research group [1]-[5]. Some realistic large-scale simulations carried out in supercomputers at the Barcelona Supercomputer Center and Rome's CASPUR proved the superiority of PDD over ScaLAPACK in terms of both total time and observed scalability when solving a nonlinear equations including KPP [7],[8], [4]. Moreover, in [5], the Vlasov-Poisson equations were tackled, which are of unquestionable interest in plasma physics. Independently, Gobet and Mairé proposed a PDD-like domain decomposition scheme for the Poisson equation in [31]. Bihlo and Haynes have applied PDD to the generation of meshes for finite elements, a PDE-based task very well suited to stochastic representations [15], [14], [16].

3 The Elliptic Case

To get an idea of the probabilistic method for PDEs and show the potential power of PDD, let us start with a linear elliptic PDE with Dirichlet boundary conditions:

$$\mathcal{L}(\mathbf{x})u - c(\mathbf{x})u = f(\mathbf{x}) \text{ if } \mathbf{x} \in \Omega, \qquad u(\mathbf{x}) = g(\mathbf{x}) \text{ if } \mathbf{x} \in \partial\Omega \tag{3.1}$$

where $\partial\Omega$ the boundary of the smooth domain $\Omega \subset \mathbb{R}^d$, and where $\mathcal{L}$ is the differential operator defined by

$$\mathcal{L}(\mathbf{x}) := \frac{1}{2}\sum_{i,j=1}^{d} a_{ij}(\mathbf{x})\frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{d} b_i(\mathbf{x})\frac{\partial}{\partial x_i}. \tag{3.2}$$

We further assume the functions $a_{ij}, b_k, c, f, g : \mathbb{R}^d \to \mathbb{R}$ for $i, j, k \in \{1, \cdots, d\}$ satisfy the necessary assumptions such that (3.1) has a unique solution. Such problems have been well studied and one can find the necessary requirements in texts such as [24], [26].

The probabilistic representation of 3.1 is given by Dynkin's formula [26]:

$$u(\mathbf{x}) = \mathbb{E}_{\mathbf{x},0}\left[g(X_{\tau_{\partial\Omega}})e^{-\int_0^{\tau_{\partial\Omega}} c(X_s)\mathrm{d}s} - \int_0^{\tau_{\partial\Omega}} f(X_t)e^{-\int_0^t c(X_s)\mathrm{d}s}\mathrm{d}t\right], \qquad \mathbf{x} \in \mathbb{R}^d, \tag{3.3}$$

where the "trajectory" $(X_t)_{t\geq 0}$ is now the solution to the SDE

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}W_t, \quad X_0 = \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t \geq 0. \tag{3.4}$$

In (3.4), $(W_t)_{t\geq 0}$ is a d-dimensional Brownian motion (with associated probability space); $b = (b_1, \ldots, b_d)$ is the drift; σ is the diffusion matrix, such that $\sigma\sigma^T = [a_{ij}]$ ($[a_{ij}]$ is the matrix of second-derivative coefficients of $\mathcal{L}(\mathbf{x})$) and $\tau_{\partial\Omega}$ is the first passage time (or hitting time) of the trajectories started at point $\mathbf{x}$ to the domain's boundary $\partial\Omega$.

An illustrative example of the effect of improved numerics

While (3.3) is one of the simplest possible stochastic representations and well known for over half a century, remarkable research into the numerical methods to solve it during the last fifteen years has meant that it can be solved today at a fraction of the cost required when PDD was introduced. To highlight the effect of improved stochastic numerics on the PDD methodology, let us take an example from [11]. There the authors study the BVP

$$\nabla^2 u + \frac{\cos(x+y)}{1.1+\sin(x+y)}\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) - \frac{x^2+y^2}{1.1+\sin(x+y)}u + f(x,y) = 0, \tag{3.5}$$

with $f(x,y)$ such that $u(x,y) = 2\cos\big(2(y-2)x\big) + \sin\big(3(x-2)y\big) + 3.1$ is the exact solution, as well as the Dirichlet boundary condition. The BVP domain can be seen in Figure 2.1.

This problem was solved with the most current version of PDD (called IterPDD in [11]). IterPDD is a numerical suite which includes several methods including: variance reduction techniques with iterative control variates based on nested, increasingly accurate global solutions.

The speedup of a method A over a method B to solve a BVP is defined as

$$S(A,B) = \frac{\text{Time taken by method B}}{\text{Time taken by method A}}. \tag{3.6}$$

Table 3.1 shows the speedup of IterPDD over the previous versions of PDD. Note that IterPDD retains all the advantages of any previous version of the PDD algorithm and hence it is not necessary to compare it against deterministic domain decomposition solvers, since

a_0	theoretical speedup	observed speedup (approx.)
.04	13.93	15
.02	28.34	29
.01	57.42	60
.005	116.00	125
.0025	233.84	250

Table 3.1. Acceleration of the most current version of PDD (IterPDD), with improved stochastic numerics, over the previous PDD algorithm. a_0 is the error tolerance on interfacial nodes. The BVP being solved is (3.5).

$S(IterPDD, DDD) = S(IterPDD, PDD) \times S(PDD, DDD)$. The theoretical speedup in Table 3.1 is the optimal speedup predicted by a sensitivity algorithm presented in [11]. Importantly, the latter is designed in such a way that it relies on fast warm up Monte Carlo sampling, and runs in parallel so as not to defeat the purpose of PDD. The agreement with the experimentally observed speedup is consistently conservative.

The acceleration (speedup) grows larger as the target nodal accuracy a_0 (the largest admissible error of the numerical solution on the interface nodes, obtained via the probabilistic representation) decreases, approximately in an inversely proportional way. In summary, nearly every previously reported result using PDD, which already were faster than deterministic solvers, could be additionally accelerated by between one and two orders of magnitude thanks to improved numerics. Further acceleration improvements are expected to follow in combination with Multilevel formulations [27]. Moreover, this improved version of PDD can be used with *any* linear BVP–not necessarily elliptic with Dirichlet BCs as (3.3).

4 Parabolic Case: the old and new ideas on Branching

One can derive stochastic representations for nonlinear parabolic PDEs and therefore PDD methods are not limited to the elliptic case. Restrictions are necessary, nonetheless, recent results have allowed for fast computation for a large classes of parabolic PDEs. We briefly introduce the linear case and then generalize to nonlinear parabolic differential equations. The techniques are similar for linear and nonlinear but the latter is more involved.

As in the previous section, to each PDE below one associates the differential operator $\mathcal{L}$ and a stochastic process X as was done in between (3.2) and (3.4). To keep notation simple, we assume that b, σ in (3.2) do not depend on time (this is straightforward to generalize).

4.1 Linear Problem

This case is fully discussed in [7]. Following on from the above argument, let us consider a parabolic PDE defined on $\Omega \times [0,T] \subset \mathbb{R}^d \times [0,\infty)$ with u satisfying,

$$\begin{cases} \dfrac{\partial u}{\partial t} - \mathcal{L}(\mathbf{x})u + c(\mathbf{x},t)u = 0\,, & \text{if } \mathbf{x} \in \Omega \\ u(\mathbf{x},0) = \psi(\mathbf{x}) & \\ u(\mathbf{x},\cdot) = g(\mathbf{x},\cdot) & \text{if } \mathbf{x} \in \partial\Omega \end{cases}, \tag{4.1}$$

where $\mathcal{L}$ is the differential operator in (3.2) (which can be made to depend on time) and ψ, g are regular enough for a unique solution to exist.

The stochastic representation for $(\mathbf{x},t) \in \Omega \times (0,T]$ can then be shown to be,

$$\begin{aligned} u(\mathbf{x},t) =& \mathbb{E}_{\mathbf{x},0}\left[\psi(X_t)e^{-\int_0^t c(X_s,t-s)ds}\mathbb{1}_{\{\tau_{\partial\Omega}>t\}}\right] \\ &+ \mathbb{E}_{\mathbf{x},0}\left[g(X_{\tau_{\partial\Omega}}, t-\tau_{\partial\Omega})e^{-\int_0^{\tau_{\partial\Omega}} c(X_s,t-s)ds}\mathbb{1}_{\{\tau_{\partial\Omega}\le t\}}\right], \end{aligned}$$

where X and $\tau_{\partial\Omega}$ have the same definition as before, $\mathbb{1}$ is the indicator function and $\mathbb{E}_{\mathbf{x},0}[\cdot]$ denotes the expectation conditional on starting the stochastic process X at $\mathbf{x}$ at time 0. We see that the indicator essentially is describing which boundary we hit. Where as in the elliptic case, we only had a spatial boundary, here it is possible to hit the time boundary before the spatial boundary and such an event is described by $\mathbb{1}_{\{\tau_{\partial\Omega}>t\}}$.

▷ *Branching diffusions and particles*. In the case of c constant, an efficient strategy (which is related to what we use in the nonlinear case) to simulate this process was proposed by [48], namely, to recognize that the term e^{-ct} is related to the density function of a Exponentially distributed random variable and the idea is to use "particles" which survive for an exponentially distributed amount of time. This exponential random variable is independent from the Brownian motion and the intensity is defined from the PDE being considered. The solution of the PDE is then the time t value of the expected fraction of surviving particles starting at some point $\mathbf{x}$. Note, the method described in [48] allows for c to be space dependent but requires converting the problem into Fourier space and assumes conditions on the coefficients. We will ignore such a technique due to the increase in generality we obtain in the following sections an concentrate purely on the case of c constant here.

To illustrate this method we consider (4.1), but with constant $c > 0$. Assume we want to numerically solve the problem at point $(\mathbf{x},t)$ using the stochastic representation and denote by h the step size in our SDE numerical scheme.

4.2 Branching Diffusions and the KPP equation

Probabilistic representations are not limited to linear parabolic PDEs and there are many cases where we can use similar techniques to solve more general PDEs. The first contributions to this are due to [49], [52], [43], to solve the KPP equation (in one spatial dimension),

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - u(u-1) = 0, \qquad u(x,0) = \psi(x) \qquad x \in \mathbb{R}\,,\ t > 0\,,$$

Algorithm 4.1 Solve a linear Parabolic PDE of the form (4.1) at a fixed point $(\mathbf{x}, t)$

for N Monte Carlo simulations **do**
  Set $X_0 = \mathbf{x}$, Event$= 0$ and Simulate $t_c \sim \text{Exp}(c)$.
  while Event$= 0$ **do**
    Simulate X over time step h
    if X hits spatial boundary **then**
      Evaluate $u^{(s)} := g(X_{\tau_{\partial\Omega}}, t - \tau_{\partial\Omega})$ and set Event$= 1$.
    else if we reach time t_c **then**
      Solution is zero i.e. $u^{(s)} := 0$ and set Event$= 1$.
    else if we reach time t **then**
      Evaluate $u^{(s)} := \psi(X_t)$ and set Event$= 1$.
    end if
  end while
end for
Approximate the solution as $\hat{u}(\mathbf{x}, t) = \frac{1}{N}\sum_{s=1}^{N} u^{(s)}$.

This was later generalized to allow for nonlinear PDEs such as,

$$\frac{\partial u}{\partial t} - \mathcal{L}(\mathbf{x})u - c\left(\sum_{i=0}^{\infty}\alpha_i u^i - u\right) = 0\,,$$

where c is a positive constant. Solutions to such nonlinear PDEs can be represented through so-called *branching diffusion processes* (see [34] for a brief overview on this as well as super diffusions). Again the life time of the particles is governed by an exponential random variable with intensity c and the SDE driving the particle is dependent on $\mathcal{L}$.

Assumption 4.1 (Classical Branching Diffusions [43]) *In the classical setting the stochastic representation of the above PDE relies on the following two conditions for the coefficients*α*;* $\alpha_i \geq 0$ *such that* $\sum_{i=0}^{\infty}\alpha_i = 1$.

The Branching diffusion algorithm is similar to Algorithm 4.1, but with a crucial difference. Here when the simulated exponential time is reached, m new particles are created with probability α_m. We see that taking $\alpha_0 = 1$ and the other $\alpha_k = 0$, reduces this system to the linear case above since when then exponential time has expired the particle produces no descendants. In the event of multiple new particles being created, each new particle is completely independent of the previous one, each particle has its own exponential random variable to indicate the remaining time and own driving Brownian motion. In this context c is given the name "Branching rate". We then iteratively continue to simulate exponential times and particles until every particle has either died or a boundary has been hit. We provide a detailed algorithm for branching diffusions in Section 5.1.

To give an idea of branching diffusions we illustrate a sample path for a given PDE. Let us consider the PDE (one spatial dimension),

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - \frac{1}{2}\left(1 + u^2\right) + u = 0, \qquad u(x, 0) = \psi(x), \quad x \in \mathbb{R}.$$

From the work above there are two possible outcomes at a branching time, either the

particle dies, or splits into two descendants, the probability of each of these events is $\frac{1}{2}$. A possible trajectory (to compute $u(x_0, T)$) is shown in Figure 4.2.

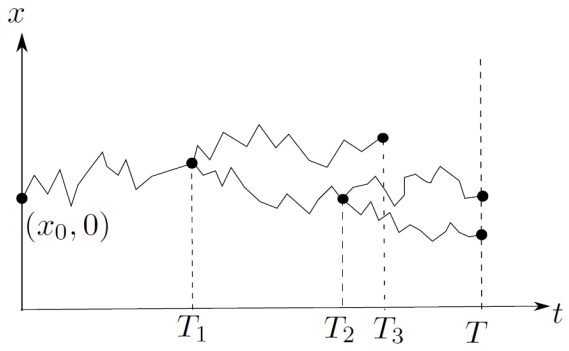

Figure 4.1. Representation of a branching diffusion, the first particle starts at point x_0 at time $t = 0$, then after time T_1 branches into two particles. The first of these later dies at T_3 with no descendants, the other particle branches again in two particles at time T_2. Both of these (third generation) particles hits the terminal time, T boundary.

Consider the same initial and boundary value as in (4.1). The solution u can then be written as the expected value of the product of the surviving particles, in the case with no boundaries this is

$$u(\mathbf{x}, t) = \mathbb{E}_{\mathbf{x},0}\left[\prod_{i=1}^{N_t} \psi(X_t^{(i)})\right], \tag{4.2}$$

where $X_t^{(i)}$ is the location of the ith particle surviving at time t and N_t is the number of surviving particles at time t.

In the presence of spatial boundaries, as in (4.1), we obtain a similar representation, however, the particle stops as soon as it hits the boundary and therefore we also have a boundary term. The stochastic representation is given by (see [7]),

$$u(\mathbf{x}, t) = \mathbb{E}_{\mathbf{x},0}\left[\prod_{i=1}^{N_t} \left(\psi\big(X_t^{(i)}\big)\mathbb{1}_{\{t<\tau_{\partial\Omega}\}} + g\big(X_{\tau_{\partial\Omega}}^{(i)}, t - \tau_{\partial\Omega}\big)\mathbb{1}_{\{t\geq\tau_{\partial\Omega}\}}\right)\right].$$

Although this set up allows for more complex PDEs than those considered in Section 4.1, it is not difficult to see that the computational complexity is greater for these types of problems. Moreover, because of the nature of the branching, we are mainly confined to cases which are not overly non-linear with time horizons that are not too long. Otherwise, we may end up with "explosions" whereby we are required to keep track of an extremely large number of particles, which will also tend to increase the number of branchings. The latter is an extremely important issue which also appears in the next section, thus we postpone its discussion (and assumptions for non-explosiveness).

4.3 Marked Branching Diffusions

We end this section with the most general branching diffusion representations of non-linear PDEs available at the present time. The idea of *marked branching diffusions* was

originally developed by [34] as a solution to pricing credit valuation adjustments (CVAs) (see Section 5.2), but since then has been generalized by [36] and [35]. The first major step towards a generalization was presented in [34], the idea was to study a terminal value PDE of the form[4],

$$\frac{\partial u}{\partial t} + \mathcal{L}(\mathbf{x})u + c\left(\sum_{i=0}^{M} \alpha_i u^i - u\right) = 0\,, \qquad u(\mathbf{x}, T) = \psi(\mathbf{x}) \qquad \mathbf{x} \in \mathbb{R}^d, \tag{4.3}$$

the notation follows that in Section 4.2 with M a positive integer

Assumption 4.2 (Marked Branching Diffusions [34]) *We* drop *the conditions for non-negative and sum of α's being equal to one and introduce a probability distribution p such that we can rewrite the sum as, $\sum_{i=0}^{M} \frac{\alpha_i}{p_i} p_i u^i$, where p is chosen so $p_i > 0$ if $\alpha_i \neq 0$.*

Although Assumption 4.2 is more general than Assumption 4.1, the representation is similar. The technique is the same as standard branching diffusions but here we have a branching of type k with probability p_k and the weights $\frac{\alpha_i}{p_i}$ appear in the representation of the solution, compare with (4.2). The representation of (4.3) (see [34]) is then given as[5],

$$u(\mathbf{x}, t) = \mathbb{E}_{\mathbf{x},t}\left[\prod_{i=1}^{N_T} \psi\big(X_T^{(i)}\big) \prod_{k=0}^{M} \left(\frac{\alpha_k}{p_k}\right)^{\omega_k}\right],$$

where ω_k is the number of times a branching of type k occurs, the SDE for X is completely determined by $\mathcal{L}$. A detailed discussion regarding the convergence of these processes (i.e. non-explosion) can be found in [34], we do not discuss it in favor of increasing the generality. It is also of note that the specific choice of probability distribution p (subject to assumptions) does not change the representation, however, from the view point of variance reduction an optimal choice exists, see [34].

In the context of Assumption 4.1 or 4.2, the α's are constants which is a clear restriction. We now discuss [36], where the α's *are allowed to be functions of time and space*. We rewrite (4.3) and consider terminal value PDEs of the form,

$$\frac{\partial u}{\partial t} + \mathcal{L}(\mathbf{x})u + c\left(\sum_{i=0}^{M} \alpha_i(\mathbf{x}, t) u^i - u\right) = 0, \qquad u(\mathbf{x}, T) = \psi(\mathbf{x})\,. \tag{4.4}$$

We now discuss sufficient conditions for our stochastic representation to be the unique viscosity solution of this PDE. The technique in [36] is to use the fact that a large class of PDEs can be represented by FBSDEs (forward backward stochastic differential equations), see [46]. The idea is to derive sufficient conditions for the FBSDE to be the unique bounded viscosity solution to a PDE of the form (4.4) and use this to obtain the results on branching diffusions. The key assumption is [36, Assumption 2.2] which we state.

[4] This PDE is presented as a backward problem. It is easy to switch between the forward and backward formulation by a classical time-change argument $\hat{t} = T - t$.

[5] The terminal value has changed the representation slightly. The key observation is that for the solution at time t one starts with a single particle located at $\mathbf{x}$ at time t

Denote the two functions for $s \in [0, \infty)$

$$l_0(s) := \sum_{k \geq 0} ||\alpha_k||_\infty s^k \quad \text{and} \quad l(s) := c\Big(||\psi||_\infty^{-1} \, l_0\big(s||\psi||_\infty\big) - s \Big),$$

where $|| \cdot ||_\infty$ is the L^∞-norm. We then have the following assumption

Assumption 4.3 (1) *The power series l_0 has a radius of convergence $0 < R \leq \infty$. Moreover, the function l satisfies one of the following conditions:*
(i) $l(1) \leq 0$,
(ii) $l(1) > 0$ *and for some* $\hat{s} > 1$, $l(s) > 0$, $\forall s \in [1, \hat{s})$ *and* $l(s) = 0$,
(iii) $l(s) > 0 \; \forall s \in [1, \infty)$ *and* $\int_1^{\bar{s}} \frac{1}{l(s)} ds = T$, *for some constant* $\bar{s} \in (1, \frac{R}{||\psi||_\infty})$.
(2) *The terminal function satisfies* $||\psi||_\infty < R$.

Under the above assumption, the stochastic equations associated to (4.3) have a unique bounded solution and yield the unique viscosity solution for PDE (4.3) (see Proposition 2.3 and Theorem 2.13 of [36]). The above assumption provides sufficient conditions for the solution not to explode and hence the representation by branching diffusions can be used.

We are now most of the way to stating the branching diffusion representation in this more general set up. However, it will beneficial for us to introduce notation to keep track of the particles. Let T_n the nth branching time of the system where at time T_n one of the particles branches into k particles, with probability p_k. We denote by I_n the number of descendants created at time T_n, hence $I_n \in \{0, \dots, M\}$. Denote by $M_{T-t} := \sup\{n : t + T_n \leq T\}$ the number of branches that occurred between time t and T. Further denote by $\mathcal{K}_t$ the index of all particles alive at time t, therefore $\mathcal{K}_{T-t}$ corresponds to the index of all particles alive at time $T - t$. Finally, denote by K_n the index of the particle that has branched at time T_n (hence $K_n \in \mathcal{K}_{T_n}$). The representation for the solution of the PDE (4.4) at $(\mathbf{x}, t)$ is given as

$$u(\mathbf{x}, t) = \mathbb{E}_{\mathbf{x},t} \left[\prod_{k \in \mathcal{K}_{T-t}} \psi(X_T^{(i)}) \prod_{n=1}^{M_{T-t}} \left(\frac{\alpha_{I_n}(t + T_n, X_{t+T_n}^{(K_n)})}{p_{I_n}} \right) \right].$$

4.3.1 *Age-Marked Branching Diffusions*

In [35], the authors generalize the representation to a wider class of nonlinear PDEs still building on the paradigm of nonlinear PDEs with polynomial source terms f. Consider for some integer $m \geq 0$ a set $L \subset \mathbb{N}^{m+1}$ and a sequence of functions $(c_l)_{l \in L}$ and $(h_i)_{i=1,\cdots,m}$, where $c_l : [0, T] \times \mathbb{R}^d \to \mathbb{R}$ and $h_i : [0, T] \times \mathbb{R}^d \to \mathbb{R}$. Further, denote by $|l| = \sum_{i=0}^m l_i$ for every $l \in L$. Let the source term function $f : [0, T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ be of the form

$$f(t, x, y, z) = \sum_{l=(l_0,\dots,l_m) \in L} c_l(t, x) y^{l_0} \prod_{i=1}^m (h_i(t, x) \cdot z)^{l_i}.$$

With such a function in mind (under some regularity assumptions) [35] provides a stochastic representation for PDEs of the form,

$$\frac{\partial u}{\partial t} + \mathcal{L}(\mathbf{x})u + f(\cdot, u, Du) = 0\,, \quad \text{with } u(\cdot, T) = \psi(\cdot), \quad t \in [0, T],\ \mathbf{x} \in \mathbb{R}^d\,,$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is a bounded Lipschitz function. As was alluded to earlier, the stochastic representation holds under the conditions of small maturity or small nonlinearity (to stop the branching diffusions from exploding), for a full treatment the reader should consult [35]. The two main increases in generality here is that, firstly, we allow for polynomial nonlinearities in the solution and gradient of the solution, this allows to tackle known PDEs such as Burgers equation or those with terms[6] $u|\nabla u|^2$. The second generalization is highlighted in the next section. Age-marked branching diffusions is more technical than the previous cases considered and, for completeness, we mention it here without giving many details.

4.3.2 *More general PDEs through approximation*

The stochastic representations shown in this section can be used to deal with many different types of PDE. Namely, *these representations allow us to consider PDEs whose source term f can be well approximated by polynomials*. An example of this is due to [34] where the objective was to price a so-called Credit Valuation Adjustment (CVA) options (see Section 5.2 below).

CVAs options are tricky to deal with because the PDE's source term involves functions of the type $u^+ := \max(u, 0)$, where u is the solution to the PDE (see (5.1) below). A direct approach using representation without branching diffusion would require regressions [29] or a type of Monte Carlo of Monte Carlo approach to deal with. Using a priori knowledge (that φ is bounded and hence u is also bounded), the idea here is to then to fit a polynomial (which is a function of u) to u^+, this reduces the PDE to one whose solution may be represented as a marked branching diffusion not as a KPP type equation. We will discuss this specific case in detail in Section 5.2.

Although the above example involves CVAs, many other financial contracts rely on maximums between two entries (so called obstacle PDEs), these appear in American options for example. Moreover, we are not limited to obstacle PDEs, [17] use the polynomial representation of trigonometric functions to calculate the non-linear Poisson-Boltzmann equation.

Obstacle PDEs

A common problem in financial mathematics involves computing the solution to certain obstacle PDE, in stochastics this is deeply related to optimal stopping problems. We write such PDEs in their variational formulation: for a exercise payoff φ the price function

[6] Equations of the type $u_t = \Delta u + u|\nabla_x u|^2$ for $x \in \Omega$, $t \geq 0$ and where Ω is a smooth bounded domain of $\mathbb{R}^d$ appear in many applications, for instance, to construct harmonic homotopic maps [51]; in the theory of ferromagnetic materials (through the Landau-Lifschitz-Gilbert equation or models of magnetostriction); and the theory of liquid crystals [13].

u solves

$$\max\Big(\frac{\partial u}{\partial t}+\mathcal{L}u,\varphi(\mathbf{x})-u\Big)=0,\qquad u(T,x)=\varphi(\mathbf{x}),\qquad (\mathbf{x},t)\in[0,T]\times\mathbb{R}^d.$$

In [9], the authors showed how this PDE can be converted into a semilinear PDE

$$\frac{\partial u}{\partial t}+\mathcal{L}u=\mathbb{1}_{\{\varphi(\mathbf{x})\geq u\}}\mathcal{L}\varphi,\qquad u(T,\mathbf{x})=\varphi(\mathbf{x}),\qquad (\mathbf{x},t)\in[0,T]\times\mathbb{R}^d.$$

A stochastic representation for these equations is well-understood and the methodology we have presented so far can be applied to them, see [34, Section 2.3].

5 Data Science Examples

We apply the theory of branching diffusions for non-linear PDEs to detail the scope of the parallel structure of PDD to enhance computational performance in solving PDEs. We have chosen PDEs from various areas and with different structures to show the versatility of stochastic representations and the PDD as a broadly applicable method.

5.1 Reaction-Diffusion: KPP Equation

The first example comes from the area of reaction diffusion equations, this class of equations are used in many areas of science such as physics and biology For illustration purposes we consider the most well known of these, the Kolmogorov-Petrovsky-Piskunov (KPP) equation [40], linked to applications in plasma physics. The general form of the KPP is the following nonlinear parabolic PDE,

$$\frac{\partial u}{\partial t}-D\frac{\partial^2 u}{\partial x^2}-ru(1-u)=0\,,\qquad (x,t)\in\mathbb{R}\times[0,\infty)$$

where D and r are constants. Following [7], taking $r=D=1$ and the initial value as,

$$u(x,0)=1-\Big(1+\exp\big(x/\sqrt{6}\big)\Big)^{-2}\ \big(=\psi(x)\big)\,,$$

we can write the true solution of this problem as,

$$u(x,t)=1-\Big(1+\exp\big(\frac{x}{\sqrt{6}}-\frac{5t}{6}\big)\Big)^{-2}.$$

The advantage of such an example is that we can also compare the error of each method. Following (4.2), the stochastic representation is simply,

$$u(x,t)=\mathbb{E}_{\mathbf{x},0}\left[\prod_{i=1}^{N_t}u\big(W_t^{(i)}+x,0\big)\right]=\mathbb{E}_{\mathbf{x},0}\left[\prod_{i=1}^{N_t}\left(1-\Big(1+\exp\Big(\big(W_t^{(i)}+x\big)/\sqrt{6}\Big)\Big)^{-2}\right)\right],$$

where N_t is the number of particles at time t, and $W_t^{(i)}$ is the Brownian motion for particle i at time t. Since the operator $\mathcal{L}$ is $\partial^2/\partial x^2$, the process X is Brownian motion started at point x.

Algorithm for the problem

We solve this problem in the one dimensional setting[7] over the domain $x \in [-1000, 1000]$ for $t \in [0, 1]$. For 6 processors we split the domain into $[-1000, -500]$, $[-500, 0]$, $[0, 500]$ and $[500, 1000]$, this of course may not be the most optimal approach, but our goal is only to show how even a simple PDD approach can dramatically improve the computational time required. We further calculate the solution at 11 equally spaced time points (including the origin), which we denote by the set of T_i, satisfying $0 = T_0 < T_1 < \cdots < T = 1$. Denote by Γ the set of nodes at which we approximate the boundaries (true and artificial), hence for M domain points, with Θ time points, Γ is an M-by-Θ matrix. We denote by Γ_k for $k \in \{1, \dots, M\}$ the column vectors of Γ, i.e. the approximation of the boundary at the kth domain point through time.

It is clear from the form of the PDE that all branching types will be two (the non-linear part is a squared). Moreover the driving process for the branching diffusions is Brownian motion. This allows for us to take large step sizes since the simulation of Brownian motion is unbiased. Algorithm 5.1 provides a summary of this method (can compare this to Algorithm 4.1).

Remark 5.1 (Controlling the growth of the branching) *A useful technique to control the branching is the "pruning", which is a method whereby we truncate the number of branches and comes from the observation that the pruned branches contribute very little to the expectation and are extremely computationally expensive. We do not discuss this further here, but direct the interested reader to [6].*

Results Error Analysis

Remark 5.2 (Straightforward implementation for Monte Carlo) *For this example we used only a standard Monte Carlo implementation. Therefore, the results presented should be seen as the lower bound for what is achievable. It possible to adapt some of the methods discussed in Section 3 and references therein to vastly improve the performance (speed and/or accuracy) of the algorithm.*

Figure 5.1 compares the absolute error of Monte Carlo versus using only the PDE solver over the domain $[-5, 5]$. We observe the maximum error of the the pure PDE solver is approximately $1/3$ that of PDD. However, the largest error for PDD is concentrated in one region to the left hand side of the boundary and outside this small region the two errors are comparable.

[7] MATLAB was used for the implementation. The simulations ran on a Dell PowerEdge R920 with four intel xeon E7-4830 processors. All polynomial interpolation were carried out using MATLAB's "polyfit" and the PDE is solved using MATLAB's "pdepe" function. To keep this consistent with the error we expect from Monte Carlo, we have set the relative and absolute error in the solver as 10^{-3}.

Algorithm 5.1 Outline to solve the KPP equation using PDD

Split Ω into subdomains Ω_i and select set of nodes Γ on the artificial boundary.
Take a series points $T_0 < T_1 < \cdots < T$ and set the "pruning" level P.
for Each $\Gamma_k \in \Gamma$ **do**
  for N Monte Carlo simulations **do**
    while $t < T$ **do**
      Set N_p as the number of particles and t as the current time.
      Find T_i, such that $T_{i-1} \leq t < T_i$ and simulate $t_c \sim \text{Exp}(c \times N_p)$.
      Simulate each particle over $\min(t + t_c, T_i)$
      if we reach time $t + t_c$ **then**
        Pick a particle (with equal probability) to branch into two
      else
        Evaluate $\psi^{(s)}(T_i) = \prod_{j=1}^{N_p} \psi(X_{T_i}^{(j)})$.
      end if
      if $N_p > P$ **then**
        Restart this iteration of the for loop
      end if
    end while
  end for
end for
for Each T_i **do**
  Calculate $\Gamma_k(T_i) = \frac{1}{N} \sum_{s=1}^{N} \psi^{(s)}(T_i)$
end for
Interpolate over each Γ_k to create artificial boundaries and solve the PDE on each Ω_i.

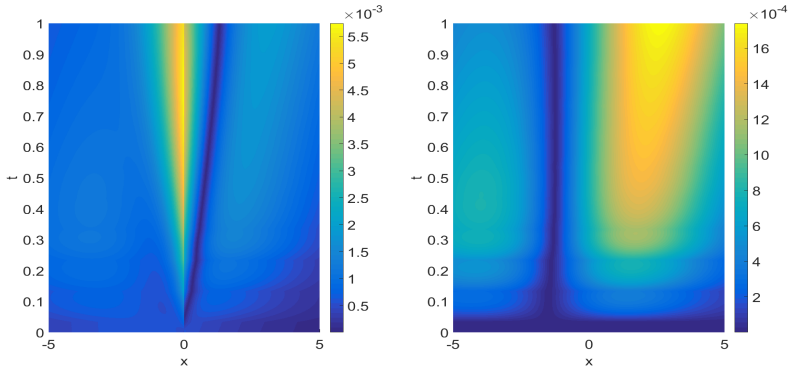

Figure 5.1. Absolute error in the region $[-5, 5]$, for $N = 10^5$ MC simulations (left), versus the pure PDE solver (right) with the PDE solver mesh set as $\Delta x = 10^{-2}$ and $\Delta t = 10^{-4}$.

Scalability and running times of the algorithm

The goal of this example is to show PDD as an efficient scalable method to solve large PDE problems, such as those arising in data science. Its main appeal is that we can completely separate the PDE problem once we have calculated the artificial boundary points. One would expect that the method should take $1/n$ of the time on n processors as it did on one and this is indeed what one can observe in Figure 5.2. Unsurprisingly, the time Monte Carlo takes to run is independent of the number of domains. Therefore, given enough processors we can reduce the computational time required to solve the problem to approximately the time taken to Monte Carlo simulation time (which rather

usefully does not suffer the curse of dimensionality like its deterministic counterparts). Of course the full time taken to solve the problem will always be above the Monte Carlo time, since we are required to estimate interpolating functions etc. However, these steps should be negligible compared to the Monte Carlo time.

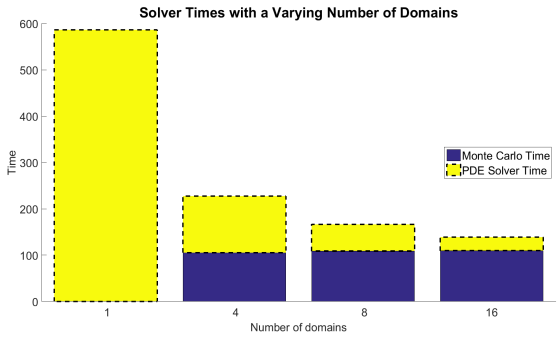

Figure 5.2. Running times (in seconds) for the PDE solver and PDD solver (Monte Carlo simulation and the PDE solver) as the number of processors increases. See Remark 5.2.

5.2 Option Pricing: Credit Valuation Adjustment (CVA)

The next example we consider is a different problem than the KPP above, but highlights the marked branching diffusion and its application scope highlighted in Section 4.3.2 very well. This example concerns option (derivative) pricing with Credit Valuation Adjustments (CVAs). Since the financial crisis in 2008, risk, especially default risk (the risk that a counter party fails to meet future obligations) has been at the forefront of many policy decisions and regulation changes for financial firms. CVAs play a crucial role here by adjusting the price of a derivative (a financial contract) with the knowledge that the counter party may default. [33] give a derivation of the PDEs arising in this problem. The specific form of the PDE depends on the way in which one chooses to model the mark-to-market value (the fair value) of the derivative at the time of default, but we obtain PDEs of the form,

$$\frac{\partial u}{\partial t} + \mathcal{L}(x)u + r_0 u + r_1 u^+ = 0, \qquad u(x,T) = \psi(x), \qquad x \in \mathbb{R}$$

where the r_i are functions of x and t and we denote by$y^+ := \max(0, y)$.

Although at first glance such a PDE seems beyond the scope of the stochastic representations in Section 4, [34] provided a simple yet powerful trick to deal with such PDEs. For simplicity we will consider only the one dimensional case. The first trick, is to rescale the problem, by assuming the payoff is bounded. Then we may consider the following function $v := u/||\psi||_\infty$, such a function satisfies the PDE

$$\frac{\partial v}{\partial t} + \mathcal{L}(x)v + r_0 v + r_1 v^+ = 0 \tag{5.1}$$

with terminal value bounded above by 1. Let us consider instead a PDE of the form

$$\frac{\partial v}{\partial t} + \mathcal{L}(x)v + c\,(F(v) - v) = 0, \qquad v(x,T) = \psi/\|\psi\|_\infty$$

where F is a polynomial (see (4.3)) and hence this equation is of the type considered previously. The great insight in [34] was to use a polynomial to approximate the semilinear component v^+. Therefore we have transformed the PDE from one outside the capabilities of marked branching diffusions, into one which can be. [34] uses a polynomial of order 4 to do this, which provides a good approximation since v clearly only takes values from -1 to 1. The approximation can be seen in Fig. 5.2.

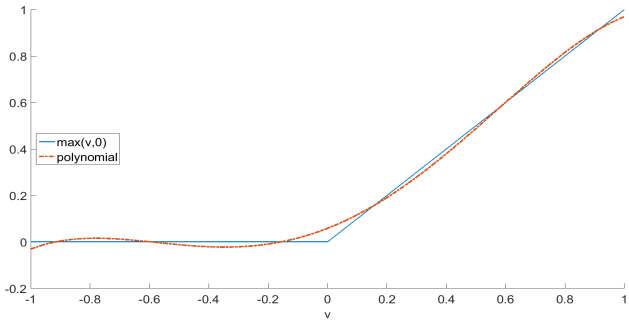

Figure 5.3. Approximation of $\max(v,0)$ over $v \in [-1,1]$ by a 4th-order polynomial.

Under this framework, using stochastic representations to calculate the solution to this PDE is well understood. Of course, there does not seem much scope for PDD here since the domain is rather small. However, in higher dimensional problems, the PDD method can very easily provide the computational advantages highlighted in the first sections.

5.3 Some other PDE examples in data science amenable to PDD

In mathematical biology one often wants to consider the affect on a population of a species given the presence of another species and the interaction between them. Such models have been considered in works such as [21], see [30] as well. In the paper by Escher and Matioc [23] they present a model to describe the growth of tumours. The model is a nonlinear two dimensional system with two decoupled Dirichlet problems.

As highlighted in Section 4.3.2, the methodology described in this work can also be used to deal with PDE obstacle problems. Apart from the well-known problem of pricing American type options we point out to the so-called *Travel agency problem* [30, Section 5.2] where a travel agency needs to decide when to offer travels depending on currency and weather forecast.

If one returns to finance, the approximation method highlighted in Section 4.3.2 combined with the age-marked branching diffusions can be used to tackle the problems in actuarial science, namely the "Reinsurance and Variable Annuity" problem (see [33]) or the pricing of contracts with different borrowing/lending rates [20, Section 3.3].

6 Conclusion and outlook

We presented an overview of a hybrid method combining stochastic methods with deterministic PDEs ones to truly take advantage parallel architectures and deal with high-dimensionality problems. The presentation was complemented by recent breakthroughs in stochastic representations techniques via branching diffusions, which expanded the applicability scope of the methods greatly. Some ingredients of this algorithm are not straightforward and require a mixed knowledge of PDE analysis and computing, tempered with notions in stochastics and probabilistic numerics. Nonetheless, the method has a huge potential and all core tools are in place for researchers to exploit it in new applications.

It is a vibrant time to be a scientist working in data science. Future years will see a rise in researchers able to fertilize ideas across multiple applied fields–such as financial mathematics, mathematical biology, and uncertainty quantification, to name but a few. We hope that raised awareness of hybrid schemes such as PDD will prompt researchers to seek elements in their data-driven, large-scale applications which are suitable for hybrid deterministic/probabilistic descriptions, with the aim of acknowledging the need for scalability already in the modelling phase.

Acknowledgements

G. dos Reis gratefully thanks the partial support by the *Fundação para a Ciência e a Tecnologia* (Portuguese Foundation for Science and Technology) through the project UID/MAT/00297/2013 (*Centro de Matemática e Aplicações*).

References

[1]Acebrón, J. A., Busico, M. P., Lanucara, P. and Spigler, R. [2005*a*], 'Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and quasi-Monte Carlo methods', *SIAM J. Sci. Comput.* **27**(2).

[2]Acebrón, J. A., Busico, M. P., Lanucara, P. and Spigler, R. [2005*b*], 'Probabilistically induced domain decomposition methods for elliptic boundary-value problems', *J Comput Phys* **210**(2), 421–438.

[3]Acebrón, J. A. and Ribeiro, M. A. [2016], 'A Monte Carlo method for solving the one-dimensional Telegraph equations with boundary conditions', *Journal of Computational Physics* **305**, 29–43.

[4]Acebrón, J. A. and Rodríguez-Rozas, Á. [2011], 'A new parallel solver suited for arbitrary semilinear parabolic partial differential equations based on generalized random trees', *J. Comput. Phys.* **230**(21), 7891–7909.

[5]Acebrón, J. A. and Rodríguez-Rozas, Á. [2013], 'Highly efficient numerical algorithm based on random trees for accelerating parallel Vlasov–Poisson simulations', *J. Comput. Phys.* **250**, 224–245.

[6]Acebrón, J. A., Rodríguez-Rozas, Á. and Spigler, R. [2009], 'Domain decomposition solution of nonlinear two-dimensional parabolic problems by random trees', *Journal of Computational Physics* **228**(15), 5574–5591.

[7]Acebrón, J. A., Rodríguez-Rozas, Á. and Spigler, R. [2010*a*], 'Efficient parallel solution of nonlinear parabolic partial differential equations by a probabilistic domain decomposition', *J. of Sci Comput.* **43**(2), 135–157.

[8]Acebrón, J. A., Rodríguez-Rozas, Á. and Spigler, R. [2010*b*], 'A fully scalable algorithm

suited for petascale computing and beyond', *Computer Science-Research and Development* **25**(1-2), 115–121.

[9]Benth, F. E., Karlsen, K. H. and Reikvam, K. [2003], 'A semilinear Black and Scholes partial differential equation for valuing American options', *Finance and Stochastics* **7**(3), 277–298.

[10]Bernal, F. and Acebrón, J. A. [2016*a*], 'A comparison of higher-order weak numerical schemes for stopped stochastic differential equations', *Comm. Comput. Phys.* **20**, 703–732.

[11]Bernal, F. and Acebrón, J. A. [2016*b*], 'A multigrid-like algorithm for probabilistic domain decomposition', *Computers & Mathematics with Applications* **72**(7), 1790–1810.

[12]Bernal, F., Acebrón, J. A. and Anjam, I. [2014], 'A stochastic algorithm based on fast marching for automatic capacitance extraction in non-Manhattan geometries', *SIAM Journal on Imaging Sciences* **7**(4), 2657–2674.

[13]Bertsch, M., Dal Passo, R. and van der Hout, R. [2002], 'Nonuniqueness for the heat flow of harmonic maps on the disk', *Arch. Ration. Mech. Anal.* **161**(2), 93–112.

[14]Bihlo, A. and Haynes, R. D. [2014], 'Parallel stochastic methods for PDE based grid generation', *Computers & Mathematics with Applications* **68**(8), 804–820.

[15]Bihlo, A. and Haynes, R. D. [2016], A stochastic domain decomposition method for time dependent mesh generation, *in* 'Domain Decomposition Methods in Science and Engineering XXII', Springer, pp. 107–115.

[16]Bihlo, A., Haynes, R. D. and Walsh, E. J. [2015], 'Stochastic domain decomposition for time dependent adaptive mesh generation', *arXiv:1504.00084* .

[17]Bossy, M., Champagnat, N., Leman, H., Maire, S., Violeau, L. and Yvinec, M. [2015], 'Monte Carlo methods for linear and non-linear Poisson-Boltzmann equation', *ESAIM: Proceedings and Surveys* **48**, 420–446.

[18]Bouchard, B., Elie, R. and Touzi, N. [2009], Discrete-time approximation of BSDEs and probabilistic schemes for fully nonlinear PDEs, *in* 'Advanced financial modelling', Vol. 8 of *Radon Ser. Comput. Appl. Math.*, Walter de Gruyter, Berlin, pp. 91–124.

[19]Cheridito, P., Soner, H. M., Touzi, N. and Victoir, N. [2007], 'Second-order Backward Stochastic Differential Equations and fully nonlinear parabolic PDEs', *Comm. Pure Appl. Math.* **60**(7), 1081–1110.

[20]Crisan, D. and Manolarakis, K. [2010], 'Probabilistic methods for semilinear partial differential equations. Applications to finance', *Mathematical Modelling and Numerical Analysis* **44**(5), 1107.

[21]Dancer, E. N. and Du, Y. H. [1994], 'Competing species equations with diffusion, large interactions, and jumping nonlinearities', *J. Differential Equations* **114**(2), 434–475.

[22]El Karoui, N., Kapoudjian, C., Pardoux, E., Peng, S. and Quenez, M. C. [1997], 'Reflected solutions of backward SDEs, and related obstacle problems for PDEs', *Ann. Probab.* **25**(2), 702–737.

[23]Escher, J. and Matioc, A.-V. [2010], 'Radially symmetric growth of nonnecrotic tumors', *Nonlinear Differential Equations and Applications NoDEA* **17**(1), 1–20.

[24]Evans, L. C. [1998], *Partial Differential Equations*, American Mathematical Society, Providence, R.I.

[25]Frei, C. and dos Reis, G. [2013], 'Quadratic FBSDE with generalized Burgers' type nonlinearities, perturbations and large deviations', *Stochastics and Dynamics (SD)* **13**(02).

[26]Freidlin, M. [1985], *Functional integration and Partial Differential Equations*, Vol. 109 of *Annals of Mathematics Studies*, Princeton University Press, Princeton, NJ.

[27]Giles, M. B. [2015], 'Multilevel Monte Carlo methods', *Acta Numerica* **24**, 259.

[28]Gobet, E. [2001], 'Euler schemes and half-space approximation for the simulation of diffusion in a domain', *ESAIM Probab. Statist.* **5**, 261–297 (electronic).

[29]Gobet, E., Lemor, J.-P. and Warin, X. [2005], 'A regression-based Monte Carlo method to solve Backward Stochastic Differential Equations', *Ann. Appl. Probab.* **15**(3), 2172–2202.

[30]Gobet, E., Liu, G. and Zubelli, J. [2016], 'A non-intrusive stratified resampler for regression Monte Carlo: Application to solving non-linear equations'. hal-01291056.

[31]Gobet, E. and Maire, S. [2005], Sequential Monte Carlo domain decomposition for the

Poisson equation, *in* 'Proceedings of the 17th IMACS World Congress, Scientific Computation, Applied Mathematics and Simulation', Citeseer.

[32]Gobet, E. and Menozzi, S. [2010], 'Stopped diffusion processes: boundary corrections and overshoot', *Stochastic Processes and their Applications.* **120**, 130–162.

[33]Guyon, J. and Henry-Labordère, P. [2013], *Nonlinear option pricing*, CRC Press.

[34]Henry-Labordere, P. [2012], 'Counterparty risk valuation: A marked branching diffusion approach', *SSRN 1995503* .

[35]Henry-Labordere, P., Oudjane, N., Tan, X., Touzi, N. and Warin, X. [2016], 'Branching diffusion representation of semilinear PDEs and Monte Carlo approximation', *arXiv:1603.01727* .

[36]Henry-Labordere, P., Tan, X. and Touzi, N. [2014], 'A numerical algorithm for a class of BSDEs via the branching process', *Stochastic Processes and their Applications* **124**(2), 1112–1140.

[37]Higham, D. J., Mao, X., Roj, M., Song, Q. and Yin, G. [2013], 'Mean exit times and the multilevel Monte Carlo method', *SIAM/ASA Journal on Uncertainty Quantification* **1**(1), 2–18.

[38]Karatzas, I. and Shreve, S. [2012], *Brownian motion and stochastic calculus*, Vol. 113, Springer.

[39]Kloeden, P. E. and Platen, E. [1992], *Numerical solution of Stochastic Differential Equations*, Vol. 23 of *Applications of Mathematics (New York)*, Springer-Verlag, Berlin.

[40]Kolmogorov, A. N., Petrovsky, I. G. and Piskunov, N. S. [1937], 'Étude de l'équation de la diffusion avec croissance de la quantité de matiere et son application a un probleme biologique', *Moscow Univ. Math. Bull* **1**(6).

[41]Lionnet, A., dos Reis, G. and Szpruch, L. [2015], 'Time discretization of FBSDE with polynomial growth drivers and reaction–diffusion PDEs', *Ann. Appl. Probab.* **25**(5), 2563–2625.

[42]Lionnet, A., dos Reis, G. and Szpruch, L. [2016], Convergence and properties of modified explicit schemes for BSDEs with polynomial growth. arXiv:1607.06733.

[43]McKean, H. P. [1975], 'Application of Brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov', *Commun. Pure Appl. Math.* **28**(3), 323–331.

[44]Mendes, R. V. [2010], 'Poisson–Vlasov in a strong magnetic field: A stochastic solution approach', *Journal of Mathematical Physics* **51**(4), 043101.

[45]Milstein, G. N. and Tretyakov, M. V. [2013], *Stochastic numerics for mathematical physics*, Springer.

[46]Pardoux, É. and Peng, S. [1992], Backward stochastic differential equations and quasilinear parabolic partial differential equations, *in* 'Stochastic partial diff. equations and their applications (Charlotte, 1991)', Vol. 176 of *Lecture Notes in Control and Inform. Sci.*, Springer, Berlin, pp. 200–217.

[47]Peng, S. [1991], 'Probabilistic interpretation for systems of quasilinear parabolic Partial Differential Equations', *Stochastics Stochastics Rep.* **37**(1-2), 61–74.

[48]Ramirez, J. M. [2006], 'Multiplicative cascades applied to PDEs (two numerical examples)', *Journal of Computational Physics* **214**(1), 122–136.

[49]Skorokhod, A. V. [1964], 'Branching diffusion processes', *Teor. Verojatnost. i Primenen.* **9**, 492–497.

[50]Smith, B., Bjorstad, P. and Gropp, W. [2004], *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*, Cambridge university press.

[51]Struwe, M. [1996], Geometric evolution problems, *in* 'Nonlinear partial differential equations in differential geometry (Park City, UT, 1992)', Vol. 2 of *IAS/Park City Math. Ser.*, Amer. Math. Soc., Providence, RI, pp. 257–339.

[52]Watanabe, S. [1965], 'On the branching process for Brownian particles with an absorbing boundary', *J. Math. Kyoto Univ.* **4**, 385–398.