uc3m | Universidad Carlos III de Madrid

e-Archivo

This is a postprint version of the following published document:

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

DOI: https://doi.org/10.1016/j.camwa.2016.07.030

© 2016 Elsevier Ltd.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

A multigrid-like algorithm for probabilistic domain decomposition

Francisco Bernal [*] [‡] Juan A. Acebrón [†*]

Abstract

We present an iterative scheme, reminiscent of the Multigrid method, to solve large boundary value problems with Probabilistic Domain Decomposition (PDD). In it, increasingly accurate approximations to the solution are used as control variates in order to reduce the Monte Carlo error of the following iterates–resulting in an overall acceleration of PDD for a given error tolerance. The key ingredient of the proposed algorithm is the ability to approximately predict the speedup with little computational overhead and in parallel. Besides, the theoretical framework allows to explore other aspects of PDD, such as stability. One numerical example is worked out, yielding an improvement of between one and two orders of magnitude over the previous version of PDD.

Keywords: PDD, domain decomposition, scalability, high-performance supercomputing, variance reduction, Feynman-Kac formula.

1 Introduction

Probabilistic and deterministic domain decomposition. In the solution of large boundary value problems (BVPs) arising in realistic applications, the discretization of the BVP on a domain Ω leads to algebraic systems of equations that can only be solved on a parallel computer with $P \gg 1$ processors by means of domain decomposition. The idea is to divide Ω into a set of $m \geq P$ overlapping subdomains, $\Omega = \cup_{i=1}^{m} \Omega_i$, and have processor j solve the restriction of the partial differential equation (PDE) to the subdomain Ω_j. Not only does parallellization need parallel computers but parallel algorithms as well. State-of-the-art methods—which we will refer to as 'deterministic'–are iterative and require updating on the interfaces [18], a step which unavoidably involves interprocessor communication and thus is intrinsically sequential. Regardless of the sophistication of the deterministic method, this will eventually set an upper limit to the scalability of the algorithm according to Amdahl's law. Whether or

[*]INESC-ID\IST, TU Lisbon. Rua Alves Redol 9, 1000-029 Lisbon, Portugal.

[†]ISCTE - Instituto Universitário de Lisboa Departamento de Ciências e Tecnologias de Informação. Av. das Forças Armadas 1649-026 Lisbon, Portugal. (juan.acebron@ist.utl.pt)

[‡]Center for Mathematics and its Applications, Department of Mathematics, Instituto Superior Técnico. Av. Rovisco Pais 1049-001 Lisbon, Portugal. (francisco.bernal@ist.utl.pt)

not this is a practical concern depends on the size of the problem, or, equivalently: the size of the problems which can be tackled is ultimately determined by the scalability limit of the domain decomposition algorithm.

An alternative to deterministic methods which is specifically designed to circumvent the scalability issue is the probabilistic domain decomposition (PDD) method, which has been successfully applied to elliptic [1] and parabolic BVPs [2][3]. PDD consists of two stages. In the first stage, the solution is calculated only on a set of interfacial nodes along the fictitious interfaces, by solving the probabilistic representation of the BVP with the Monte Carlo method. More precisely, the pointwise solution of the BVP is $u(t,\mathbf{x}) = E[\phi(\mathbf{X}_s)|\mathbf{X}_t = \mathbf{x}]$, i.e. the expected value of a functional ϕ of a given stochastic process $\mathbf{X}_s$ conditioned to $(t,\mathbf{x})$. It is then possible to reconstruct (approximately) the solution on the interfaces, so that the PDE restricted to each of the subdomains is now well posed, and can be independently solved–the second stage of PDD. Note that both stages in PDD are embarrasingly parallel by construction. Moreover, PDD is naturally fault-tolerant.

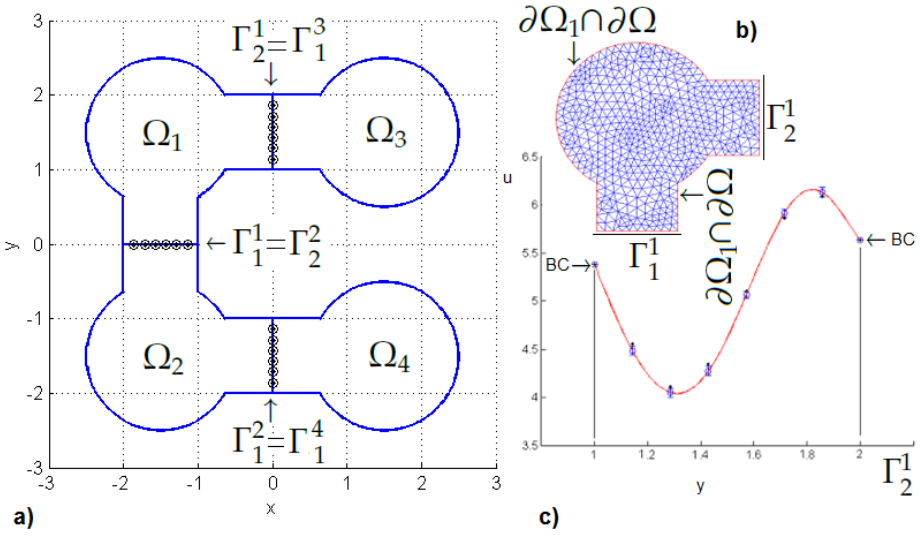

Figure 1: **a)** PDD partition into 4 subdomains and 3 interfaces with 6 nodes each ($n = 18$). **b)** FEM mesh on subdomain Ω_1. **c)** The nodal values interpolated with RBFs along the interface $\Gamma_2^1 = \Gamma_1^3$, making up a Dirichlet BC for Ω_1 and Ω_3.

Nomenclature of PDD. In this paper, we shall exclusively deal with elliptic BVPs. Let us introduce some terminology (see Figure 1). The domain $\Omega \subset \mathbb{R}^D$, $D \geq 2$ (not necessarily simply connected) on which the BVP is being solved is partitioned into m *nonoverlapping* subdomains $\Omega_1, \ldots, \Omega_m$. The boundary $\partial\Omega_k$ of a subdomain Ω_k contains several ($m_k \geq 1$) *artificial interfaces*–each of which is shared between Ω_k and another adjacent subdomain–which are labeled

$\Gamma_1^k, \ldots, \Gamma_{m_k}^k$ (note that this labeling is not unique). A subdomain boundary $\partial\Omega_k$ may or not contain some portion of the actual boundary. In sum, $\partial\Omega_k = \left(\partial\Omega_k \cap \partial\Omega\right) \cup \left(\cup_{j=1}^{m_k} \Gamma_j^k\right)$. Artificial interfaces are discretized into *interfacial nodes* (or simply, nodes) uniquely labeled $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Definition 1 *Assume m functions $f_i(\mathbf{x})$ defined on m domains D_i such that $f_i(\mathbf{y}) = f_j(\mathbf{y})$ if $\mathbf{y} \in D_i \cap D_j$. Then, their direct sum is defined as*

$$\oplus_{k=1}^m f_k(\mathbf{x}) = \begin{cases} f_i(\mathbf{x}) & \text{if } \mathbf{x} \in D_i \, (i = 1, \ldots, m), \\ 0 & \text{if } \mathbf{x} \notin \cup_{1 \le k \le m} D_k. \end{cases} \tag{1}$$

The solution of the BVP on the nodes is calculated by resorting to the probabilistic formulation of the BVP with a Monte Carlo method, yielding the *nodal values* (or *nodal solutions*) $\{u_1, \ldots, u_n\}$. Consider a subdomain Ω_k. A Dirichlet BC can then be provided on every Γ_j^k by interpolation of the nodal values $\{u_i \,|\, \mathbf{x}_i \in \Gamma_j^k\}$. Along with the actual BCs which apply on $\partial\Omega_k \cap \partial\Omega$, the BVP on Ω_k now is well posed and can be solved right away, yielding $v_k(\mathbf{x})$. Once the subdomain solutions $v_1(\mathbf{x}), \ldots, v_m(\mathbf{x})$ are available, they are put together to form a global PDD solution: $\tilde{u}(\mathbf{x}) = \oplus_{k=1}^m v_k(\mathbf{x})$. (We reserve $u(\mathbf{x})$ for the exact solution of the BVP and denote the global PDD approximations with $\tilde{u}(\mathbf{x})$.) Since adjacent subdomains Ω_i and Ω_j share a Dirichlet BC on their common interface, the PDD solution is continuous in Ω–although not necessarily differentiable.

The cost of computing the nodal solutions. The convenience of PDD depends on whether a suitable stochastic representation for the BVP under consideration is available, and on the cost involved in numerically solving it. Due to the poor accuracy of the Monte Carlo method (compared with deterministic ones), the bulk of the cost of PDD falls on the calculation of the nodal solutions to within a required accuracy. More precisely, given a *nodal error tolerance a* and a *confidence interval* P_q, the cost of solving the BVP on an interfacial node scales as $O(a^{-2-1/\delta})$, where δ is the weak convergence rate of the numerical integration scheme (also called the *bias*). This poor rate of convergence is due to the slow convergence of both the statistical error and the bias [15], which have to be tackled simultaneously. For BVPs with Dirichlet BCs there quite a few linear integrators (i.e. with $\delta = 1$) [6, 13, 17].

Regarding the statistical error, replacing the mean by a Multilevel estimator of the expected value of Feynman-Kac functionals has recently been shown to dramatically reduce the cost to $O(|\log a|^3 a^{-2})$ [11]. When the bias law is sharp, extrapolation á la Talay-Tubaro or regression methods in the spirit of [16] can further improve the accuracy at virtually no extra cost.

Using rougher numerical solutions to reduce the variance. By construction, PDD offers an additional device to accelerate the Monte Carlo simulation of the nodal values, namely the possibility of calculating and exploiting rougher estimates of the global solution of the BVP. Assuming that a numerical solution $\tilde{u}_0(\mathbf{x})$ with a *target nodal error tolerance* a_0 is required, it may be worth to calculate

before a rougher approximation $\tilde{u}_1(\mathbf{x})$, with an $a_1 > a_0$ tolerance; and then use it to draw the stochastic pathwise nodal control variate

$$\xi = -\int_0^\tau e^{\int_0^t c(\mathbf{X}_s)ds} \sigma^T \nabla \tilde{u}_1 \, d\mathbf{W}_t, \tag{2}$$

alongside the Monte Carlo realizations of the Feynman-Kac functional. (In (2), the integrals are Ito's, c is the BVP potential, and σ and τ are the diffusion matrix and first-exit time from Ω of the stochastic process (10) driven by a Wiener process $\mathbf{W}_t$.) This allows one to construct afterwards an estimator of the Feynman-Kac functional involving the control variate (2), which has the same expected value but a much smaller variance. This notion is what we call IterPDD. In order to fix ideas, let us introduce the following notation:

$$\tilde{u}(a) = \text{PlainPDD}(a), \text{ or simply PlainPDD}(a) \tag{3}$$

means that $\tilde{u}(\mathbf{x})$ is a PDD approximation obtained with tolerance a and no variance reduction; while

$$[\tilde{u}_0(a), \xi_0(\tilde{u}_1)] = \text{IterPDD}(a_0, a_1), \text{ or simply IterPDD}(a_0, a_1) \tag{4}$$

means that first $\tilde{u}_1(a_1) = \text{PlainPDD}(a_1)$ is calculated without variance reduction, then differentiated in order to construct ξ according to (2), which in turn is used as control variate in order to reduce the variance in calculating $\tilde{u}_0$ with a target tolerance a_0, which is the ultimate goal. Because the nodal values of $\tilde{u}_0$ can now be calculated with much less variance, statistical errors are smaller, and the time (or cost) it takes the computer to hit the tolerance a_0 is also less. In fact,

$$\begin{aligned} \text{cost of IterPDD}(a_0, a_1) \approx \text{cost of PlainPDD}(a_1) + \\ \Big(1 - \rho^2[\phi_0, \xi_0(\tilde{u}_1)]\Big) \times \text{cost PlainPDD}(a_0), \end{aligned} \tag{5}$$

where $\rho[\phi_0, \xi_0(\tilde{u}_1)]$ is Pearson's correlation. As (5) indicates, there is a tradeoff between the effort invested in calculating $\tilde{u}_1(a_1)$, and the reduction of variance yielded by $\xi_0(\tilde{u}_1)$, which depends on the quality of $\tilde{u}_1(a_1)$. The most straightforward procedure is to simply guess some $a_1 > a_0$. While numerical tests indicate that the IterPDD strategy can be quite successful, a poorly chosen a_1 may well result in an overall cost of IterPDD(a_0, a_1) *larger* than that of PlainPDD(a_0). Therefore, at the heart of IterPDD lies an optimization problem for a_1. In order to make educated guesses of a_1 given a_0, two questions must be tackled: i) how does the cost of a $\tilde{u}(a)$ =PlainPDD(a) simulation depend on a; and ii) how does $\rho[\phi_0, \xi_0\Big(\tilde{u}(a)\Big)]$ depend on a. The former requires that the SDE integrator have a predictable and sharp order of weak convergence. Deriving a *sensitivity formula* for ii) is one of the main points of this paper.

A multigrid-like PDD algorithm. With such a sensitivity formula in place which can predict an optimal (or more realistically, good enough) initial tolerance a_1 for $[\tilde{u}_0(a_0), \xi_1(\tilde{u}_1)]$ =IterPDD(a_0, a_1), it is natural to try and compute $\tilde{u}_1(a_1)$

faster by finding $a_2 > a_1$ which minimizes the cost of $[\tilde{u}_1(a_2), \xi_2(\tilde{u}_2)]$ =IterPDD(a_1, a_2). Much like in the Multigrid method, a number J and an *optimal sequence* of nested IterPDD simulations can be envisioned with tolerances $a_J > \ldots > a_2 > a_1 > a_0$ which fully exploits the potential of control variates for a given BVP. However, in IterPDD, given a target tolerance a_0, the number J and the sequence $a_J > ... > a_0$ must be determined *before* actually running one single PDD simulation. To compound matters, all of J, $a_J, \ldots, a_1$, and in general any result provided by any such *scheduling algorithm* will be affected by the randomness introduced by Monte Carlo, making its performance meaningful only in terms of its expected value and variance.

Outline of the paper. We start by revisiting the probabilistic formulation of elliptic BVPs with Dirichlet BCs in Section 2, and identifying the pathwise control variates. Section 3 formally poses the problem–namely, acceleration of PDD with the IterPDD scheme–and presents our own theoretical results concerning the aforesaid sensitivity $\rho[\phi_0, \xi_0\big(\tilde{u}(a)\big)]$. Section 4 links the nodal target error tolerance, a_0, to the global PDD error tolerance $\epsilon$and discusses the stability of PDD. In Section 5, the formal restrictions in Section 3 are relaxed leading to practical, but partially heuristic, scheduling algorithm/sensitivity formula (Algorithm 3) and final iterative, multigrid-like PDD loop (Algorithm 4). They are numerically tested in Section 6, and Section 7 concludes the paper.

2 Pathwise control variates for elliptic BVPs

2.1 Probabilistic representation

Consider the linear elliptic BVPs with Dirichlet BCs in $\Omega \subset \mathbb{R}^D$:

$$\begin{cases} L(\mathbf{x})u + c(\mathbf{x})u = f(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = g(\mathbf{x}), & \text{if } \mathbf{x} \in \partial\Omega, \end{cases} \tag{6}$$

where the *differential generator* L is

$$L(\mathbf{x}) := \frac{1}{2} \sum_{i=1}^{D} \sum_{j=1}^{D} a_{ij}(\mathbf{x}) \frac{\partial^2}{\partial x_i \partial x_j} + \mathbf{b}(\mathbf{x}) \cdot \nabla, \tag{7}$$

and the functions $a_{11}, \ldots, a_{DD}, b_1, \ldots, b_D, c, f, g : \mathbb{R} \mapsto \mathbb{R}$ are regular enough that the solution to (6) exists and is unique [17]. In particular, if $c \leq 0$ and the matrix $A(\mathbf{x}) = [a_{ij}(\mathbf{x})]$ is such that

$$\Lambda \in \mathbb{R} := \sup_{\mathbf{x} \in \Omega \setminus \partial\Omega} \text{spectrum of } A(\mathbf{x}) \geq \inf_{\mathbf{x} \in \Omega \setminus \partial\Omega} \text{spectrum of } A(\mathbf{x}) =: \lambda > 0, \tag{8}$$

the pointwise solution to (6) admits the following probabilistic representation

$$u(\mathbf{x}_0) = E[\phi] := E\Big[\, g(\mathbf{X}_\tau) e^{\int_0^\tau c\left(\mathbf{X}_s\right) ds} \; - \; \int_0^\tau f\left(\mathbf{X}_t\right) e^{\int_0^t c\left(\mathbf{X}_s\right) ds} dt \,\Big], \tag{9}$$

where $E[\cdot]$ stands for the expected value, the functional ϕ is called *score*, and $\mathbf{X}_\tau$ is the value at $t = \tau > 0$ of the stochastic process $\mathbf{X}_t : [0, \tau] \to \mathbb{R}^D$, driven by the stochastic differential equation (SDE)

$$d\mathbf{X}_t = \mathbf{b}(\mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \qquad \mathbf{X}_0 = \mathbf{x}_0, \tag{10}$$

where the diffusion matrix σ is obtained from $\sigma\sigma^T = A$ (by Choleski's decomposition), and $\mathbf{W}_t$ is a standard D-dimensional Wiener process. The *first exit time* τ is defined as $\tau = \inf_t \mathbf{X}_t \in \partial\Omega$, i.e. the time when a solution of the SDE (10) first touches $\partial\Omega$ at the *first exit point* $\mathbf{X}_\tau$. The process $\mathbf{X}_t$, $0 \le t \le \tau$ can be thought of as a non-differentiable trajectory inside Ω. Equation (9) is Dynkin's formula, a particular case of the more general Feynman-Kac formula for parabolic BVPs–see [9] for more details.

2.2 Errors arising in a Monte Carlo simulation

In practice, solving (9) involves two levels of discretization. First, the SDE (10) has to be integrated numerically according to a numerical scheme Ξ (which we will call *integrator*) with a timestep $h > 0$, yielding a discretized score ϕ_h. This results in $E[\phi_h]$ being a biased estimator of $E[\phi]$, with *signed* bias

$$B_\Xi\Big(E[\phi_h]\Big) := E[\phi_h] - E[\phi]. \tag{11}$$

As suggested by the notation, the bias depends on both the timestep h and the specific integrator Ξ. It takes asymptotically the form of a power law [15]:

$$B_\Xi\Big(E[\phi_h]\Big) \to \beta h^\delta \text{ as } h \to 0^+, \tag{12}$$

where β is a signed constant independent of h. Second, the expected value in (9) is replaced by an estimator over N independent realizations $\phi_h^{(1)}, \ldots, \phi_h^{(N)}$ of the SDE (10)–which is the essence of the Monte Carlo (MC) method. Typically, that estimator is the mean, which, according to the central limit theorem, introduces a *statistical error* ($V[\cdot]$ is the variance):

$$\Big|E[\phi_h] - \frac{1}{N}\sum_{j=1}^{N} \phi_h^{(j)}\Big| \le q\sqrt{\frac{V[\phi_h]}{N}} \tag{13}$$

with probability $P_q = 68.3\%, 95.5\%$, and 99.7%, for $q = 1, 2, 3$ [15]. Moreover,

$$MSE_{N,h}[\phi_h] := E\Big[\Big(\phi_h - E[\phi_h]\Big)^2\Big] \le B_\Xi^2(E[\phi]) + V[\phi]/N, \tag{14}$$

where MSE stands for mean square error. Since $E^2[|\phi_h - E[\phi]|] \le E[(\phi_h - E[\phi_h])^2]$, $E[\phi_h] = u_h(\mathbf{x}_0)$, and $E[\phi] = u(\mathbf{x}_0)$, it holds

$$|u(\mathbf{x}_0) - u_h(\mathbf{x}_0)| \leq \sqrt{MSE(\phi_h)} = \sqrt{\left(|B_\Xi(E[\phi_h])| + \sqrt{V[\phi]/N}\right)^2 - 2|B_\Xi^2(E[\phi_h])|V[\phi]/N} \leq \quad (15)$$
$$\leq |B_\Xi(E[\phi_h])| + \sqrt{V[\phi]/N} \leq |B_\Xi(E[\phi_h])| + q\sqrt{V[\phi]/N},$$

with probabilities approaching P_q as $h \to 0^+$. Looking at (14) or (15), the two ways to speed up Monte Carlo simulations of (9) are: using an integrator Ξ with the highest possible δ for the BVP under consideration; and/or replacing the mean in (13) by another estimator of the expected value which has less variance, thus achieving the same statistical error with a smaller N.

2.3 Variance reduction based on pathwise control variates

In this paper, we investigate a technique of variance reduction based on pathwise control variates, which is difficult to implement in other contexts, but suits the framework of PDD ideally. In order to discuss it, we adopt the same ansatz as in [17, chapter 2]. Consider the system of SDEs:

$$\begin{cases} d\mathbf{X}_t = [b(\mathbf{X}_t) - \sigma(\mathbf{X}_t)\boldsymbol{\mu}(\mathbf{X}_t)]dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, & \mathbf{X}_0 = \mathbf{x}_0, \\ dY_t = c(\mathbf{X}_t)Y_t dt + \boldsymbol{\mu}^T(\mathbf{X}_t)Y_t d\mathbf{W}_t, & Y_0 = 1, \\ dZ_t = -f(\mathbf{X}_t)Y_t dt + \mathbf{F}^T(\mathbf{X}_t)Y_t d\mathbf{W}_t, & Z_0 = 0, \end{cases} \quad (16)$$

where $\boldsymbol{\mu}, \mathbf{F} : \Omega \to \mathbb{R}^D$ are smooth arbitrary fields, and let $(\mathbf{X}_\tau, Y_\tau, Z_\tau)$ be the evaluation of the processes $(\mathbf{X}_t, Y_t, Z_t)$ at $t = \tau$. If $\boldsymbol{\mu}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) = 0$, the system (16) is just another way of writing down the functional in (9):

$$g(\mathbf{X}_\tau)Y_\tau + Z_\tau = \begin{cases} \phi, & \text{if } \boldsymbol{\mu}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) = 0, \\ \tilde{\phi}, & \text{otherwise.} \end{cases} \quad (17)$$

It turns out–see [17] for details–that

$$E[g(\mathbf{X}_\tau)Y_\tau + Z_\tau] = E[\tilde{\phi}] = E[\phi] \quad (18)$$

regardless of the arbitrary functions $\boldsymbol{\mu}$ and $\mathbf{F}$, but the variance

$$V[\phi] \neq V[\tilde{\phi}] = E\Big[\int_0^\tau Y_t^2 \parallel \sigma^T\nabla u + u\boldsymbol{\mu} + \mathbf{F} \parallel_2^2 \, dt\Big] \quad (19)$$

does depend on them; and in fact, making the choice

$$\sigma^T\nabla u + u\boldsymbol{\mu} + \mathbf{F} = 0 \quad (20)$$

then $V[\tilde{\phi}] = 0$–meaning that one single realization would yield the exact pointwise solution $u(\mathbf{x}_0)$ deterministically. Certainly this is a moot point, for in order to do so, the exact solution would be required in the first place; and the system (16) would still have to be integrated numerically–so that the variance of the discretized approximation would be small, but finite.

Let $\tilde{u}(\mathbf{x}) \approx u(\mathbf{x})$. Choosing $\mathbf{F} = 0$ in (20),

$$\boldsymbol{\mu} = -\frac{1}{\tilde{u}}\boldsymbol{\sigma}^T \nabla \tilde{u} \tag{21}$$

leads to the pathwise equivalent of importance sampling, while if $\boldsymbol{\mu} = 0$,

$$\mathbf{F} = -\boldsymbol{\sigma}^T \nabla \tilde{u} \tag{22}$$

can be interpreted as the stochastic equivalent of control variates. Both importance sampling and control variates are well-known variance reduction techniques in statistics [12]. There are several reasons why the second method is the more convenient:

- Control variates do not need to store $\tilde{u}(\mathbf{x})$.
- For inadequate $\tilde{u}$ and $\nabla\tilde{u}$, importance sampling may actually lead to *increased* variance. This is much less likely so with control variates (as it will be seen in a moment).
- From the point of view of numerical integration, introducing a non-zero $\boldsymbol{\mu}$ in (16) leads to a system of stochastic equations with multiplicative noise: $dY_t = c(\mathbf{X_t})Y_t dt + \boldsymbol{\mu}^T(\mathbf{X_t})Y_t d\mathbf{W}_t$. Such SDEs can be unstable [15].
- Finally, the trajectories are not affected by control variates, while if $\boldsymbol{\mu} \neq 0$, the 'particles' are actually 'pushed' around depending on $\nabla\tilde{u}$. The first fact facilitates the analysis of IterPDD.

Therefore, we will settle for (22) henceforth. In statistics, a control variate ξ for ϕ is a variable which is drawn alongside ϕ and has a large correlation with it (i.e. close to ± 1). Then, the variable

$$\tilde{\phi}(\gamma) = \phi - \gamma(\xi - E[\xi]) \tag{23}$$

is such that $E[\tilde{\phi}] = E[\phi]$, but the variance is not. At the unique critical point $\gamma_* = cov[\phi, \xi]/V[\xi]$, $V[\tilde{\phi}]$ is minimal and yields a reduction of variance

$$\frac{V[\tilde{\phi}]}{V[\phi]} = 1 - \rho^2[\phi, \xi] \tag{24}$$

(see [12] for details), where ρ is Pearson's correlation:

$$\rho[\phi, \xi] := \frac{Cov[\phi, \xi]}{\sqrt{V[\phi]V[\xi]}} \qquad \text{(such that } \rho^2[\phi, \xi] \leq 1\text{)}. \tag{25}$$

By comparison of (23) and (16), the pathwise control variate is Ito's integral

$$\xi := -\int_0^\tau Y_t \boldsymbol{\sigma}^T(\mathbf{X}_t)\nabla\tilde{u}\, d\mathbf{W}_t. \tag{26}$$

(Note that $k\xi$, with $k \neq 0$, yields the same $\rho[\phi, \xi]$ as ξ and thus the same reduction of variance.) When $\tilde{u} = u$, we will refer to ξ as the *exact control variate*. In that case, the correlation is perfect and the variance is zero (in the limit $h \to 0^+$). If $\tilde{u} \approx u$, ξ is close to the minimizer γ_* of (23) and the equality in (24) still holds. But even if $\tilde{u}$ is so poor that (24) breaks down becoming a mere lower bound, any reasonable substitute of u will still produce a ξ with sufficient correlation as not to increase the variance. Summing up, the method of pathwise control variates is intrinsically robust–unlike pathwise importance sampling (21). The control variate ξ can be drawn by a quadrature of Ito's integral (26), or alternatively, by enlarging system (16) with one extra equation:

$$d\xi_t = -Y_t \boldsymbol{\sigma}^T(\mathbf{X}_t) \nabla \tilde{u}(\mathbf{X}_t)\, d\mathbf{W}_t, \qquad \xi_0 = 0. \tag{27}$$

Importantly, the same pathwise control variate can be used for parabolic BVPs and BCs involving derivatives. We close this section by adressing the choice of integrator Ξ for solving (16) with $\mathbf{F} = -\boldsymbol{\sigma}^T \nabla \tilde{u}$ and $\boldsymbol{\mu} = 0$. The simplest integrator for bounded SDEs is the Euler-Maruyama scheme plus a naive boundary test. This yields $\delta = 1/2$, which is much poorer than the linear rate of weak convergence of the Euler-Maruyama integrator in free space [15]. However, if the solution, coefficients, and boundary of the BVP are smooth enough, then there are a variety of methods which manage to raise δ to one–see [6] for a recent review. In particular, the integrator of Gobet and Menozzi [13] restores weak linearity of the Euler-Maruyama scheme by simply shrinking the domain (Algorithm 1). (The reader is referred to that paper as to why the local shrinkage is that on line 3 below.)

Algorithm 1 Gobet and Menozzi's integrator with pathwise control variates.

1: **Data:** $h > 0$, $\mathbf{X}_0 = \mathbf{x}_0 \in \Omega$, $Y_0 = 1 \xi_0 = 0, Z_0 = 0$, $(d_0 < 0, \mathbf{N}_0)$
2: **for** $k = 0, 1, 2, \ldots$ until hitting the shrunken boundary **do**
3: **if** $d_k > -0.5826 \|\sigma_k \mathbf{N}_k\| \sqrt{h}$ **then**
4: Take one unbounded step according to:

$$\begin{cases} \mathbf{X}_{k+1} = \mathbf{X}_k + h\mathbf{b}_k + \sqrt{h}\sigma_k \mathcal{N}_D, \\ Y_{k+1} = Y_k + hY_k c_k, \\ \xi_{k+1} = \xi_k + hY_k \boldsymbol{\sigma}^T \nabla \tilde{u}(\mathbf{X}_k), \\ Z_{k+1} = Z_k + hY_k f_k + \xi_{k+1}. \end{cases} \tag{28}$$

5: Compute $(d_{k+1}, \mathbf{N}_{k+1})$ according to the distance map of Ω
6: **else**
7: Finish: $\tau_h = kh$, $\mathbf{X}_\tau = \Pi_{\partial\Omega}(\mathbf{X}_k)$, $\xi = \xi_k$ and $\phi_h = g(\mathbf{X}_{\tau_h}) Y_{\tau_h} + Z_{\tau_h}$.
8: **end if**
9: **end for**

In Algorithm 1, $d(\mathbf{x})$ is the signed distance from $\mathbf{x} \in \mathbb{R}^D$ to $\partial\Omega$ with the convention that it is negative inside Ω (see [5] for further details as to how

to produce it); τ_h is the approximation to τ; $\mathcal{N}_D$ is a $D-$dimensional standard normal distribution; $\Pi_{\partial\Omega}(\mathbf{x})$ is the closest point on $\partial\Omega$ to $\mathbf{x}$, and $\mathbf{N}$ is the unit outward normal at $\Pi_{\partial\Omega}(\mathbf{x})$ (Ω is supposed smooth enough that it exists).

3 Cost and correlation of the control variates in PDD

3.1 Complexity of a PDD simulation

Definition 2 *A balanced MC simulation with accuracy a and confidence interval P_q is such that: statistical error = $\frac{a}{2}$ = absolute value of signed bias.*

A balanced MC simulation is thus one guaranteed to attain a total $MSE_{N,h}(u_h(\mathbf{x})_\mathbf{0})$ (14) smaller than a (with a probability P_q) with the least computational complexity (cost), because it takes the largest possible h and the fewest possible realizations compatible with $a/2$ and P_q, namely

$$N = \frac{4q^2 V[\phi]}{a^2} \qquad \text{and} \qquad h = \left(\frac{a}{2|\beta|}\right)^{1/\delta}. \tag{29}$$

(We remark in passing that, while splitting the total error between bias and variance looks natural enough, it is definitely suboptimal [14].) The average number of steps before hitting the boundary is

$$\nu = E[\tau]/h, \tag{30}$$

where mean first exit time, $E[\tau]$, is well defined for a given BVP and $\mathbf{x}_0$. On average, every trajectory makes ν visits to the integrator Ξ. For a balanced MC simulation and a tolerance a, the expected cost is, then:

$$\text{nodal cost}(\mathbf{x}_0) = N\nu = NE[\tau]/h = 4q^2 E[\tau](2|\beta|)^{1/\delta}\frac{V[\phi]}{a^{2+1/\delta}} := K\frac{V[\phi]}{a^{2+1/\delta}}. \tag{31}$$

Using control variates to reduce the variance, the trajectories remain the same, but the cost per timestep is increased by a factor $\kappa \gtrsim 1$. This is due to the extra cost of interpolating the control variates from a lookup table (a $D-$dimensional array holding pointwise values) and evaluating the extra term $\mathbf{F}$ in (16). Inserting (24), the pointwise cost with control variates is

$$\text{nodal cost with control variates}(\mathbf{x}_0) = \kappa K\frac{V[\phi+\xi](1-\rho^2[\phi,\xi])}{a^{2+1/\delta}}. \tag{32}$$

The cost of a global PDD approximation $u_0(\mathbf{x})$ with PlainPDD(a_0) is then

$$\text{Cost of PlainPDD}(a_0) = \Pi + \sum_{i=1}^{n}\frac{K_i V_i[\phi]}{a_0^{2+1/\delta}}, \tag{33}$$

Π is the cost (independent of a_0) involved in solving all the subdomains using the deterministic subdomain solver once the interfacial values are available. Finally,

$$\text{Cost of IterPDD}(a_0, a_1) = 2\Pi + \tilde{\Pi} + \frac{1}{a_0^{2+1/\delta}} \sum_{i=1}^{n} K_i V_i[\phi] \left(\left(\frac{a_0}{a_1} \right)^{2+1/\delta} + \kappa \left(1 - \rho_i^2[\phi, \xi] \right) \right). \tag{34}$$

Above, $\tilde{\Pi}$ is the cost (independent of a_0 and a_1) of constructing and storing $\mathbf{F} = -\boldsymbol{\sigma}^T \nabla \tilde{u}_1(\mathbf{x})$ with $\tilde{u}_1$ =PlainPDD(a_1), and $\rho_i[\phi, \xi]$ depends on a_1 only as long as there are no quadrature errors (ie as $h \to 0^+$).

3.2 The correlation $\rho[\phi, \xi(a)]$ in the limit $h \to 0^+$

The results in this subsection are exact (within the assumptions) in the limit $h \to 0^+$; approximations are left to Section 5. To the best of our knowledge, they are also new. Let us introduce the following notation:

- $\eta \sim P$, with $\eta \in \mathbb{R}^s$ means that η is some realization of the distribution P. The notation $\eta = P(\boldsymbol{\omega})$ denotes a specific realization, labeled with the 'chance variable' $\boldsymbol{\omega} \in \mathbb{R}^s$. Thus, $\eta = P(\boldsymbol{\omega})$ is the same as $\eta \sim P$, but the former considers $\boldsymbol{\omega}$ fixed and treats η as a scalar (or vector).
- $\mathcal{N}$ is an s-dimensional standard normal distribution, with $s = 1$ (by default), or known from the context. For instance, $\mathcal{N}(\boldsymbol{\omega})$ is the realization labeled by $\boldsymbol{\omega} \in \mathbb{R}^s$.
- The notation $\eta(\cdot; \boldsymbol{\omega})$ means that the stochastic variable η (which may depend on several parameters represented by the first argument) is constructed based on a PDD simulation with nodal statistical errors labeled with $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$, where ω_i, $1 \le i \le n$ labels the statistical error on node $\mathbf{x}_i$. (An arbitrary node is denoted by $\mathbf{x}_0$ and ω_0.)

Lemma 3 *Let ϕ and η be stochastic variables, ξ an exact control variate for ϕ, and assume that $V[\phi]$, $V[\xi]$ and $V[\eta]$ are finite. It holds:*

1. $V[\xi] = V[\phi]$.
2. $Cov[\phi, \xi] = -V[\phi] \le 0$.
3. $Cov[\xi, \eta] = -Cov[\phi, \eta]$.

Proof. We will make use of the Cauchy-Schwarz inequality $|Cov[a, b]| \le \sqrt{V[a]V[b]}$ for a,b with finite variances. Notice first that $V[\phi + \xi] = 0 = V[\phi] + V[\xi] + Cov[\phi, \xi]$, so that $Cov[\phi, \xi]$ cannot be positive. For the first result, note that $|Cov[\phi, \xi]| = -Cov[\phi, \xi]$, and using the Cauchy-Schwarz inequality, $-2\sqrt{V[\phi]V[\xi]} - 2Cov[\phi, \xi] \le 0$. Summing this and $V[\phi] + V[\xi] + 2Cov[\phi, \xi] = 0$ yields

$$0 \ge V[\phi] + V[\xi] - 2\sqrt{V[\phi]V[\xi]} = \left(\sqrt{V[\phi]} - \sqrt{V[\xi]} \right)^2 \tag{35}$$

which can only happen if $V[\phi] = V[\xi]$. Then, it is clear that $2Cov[\phi, \xi] = -2V[\phi]$. Finally, $|Cov[\phi + \xi, \eta]| \leq \sqrt{V[\phi + \xi]V[\eta]} = 0$, so that $Cov[\phi + \xi, \eta] = Cov[\phi, \eta] + Cov[\xi, \eta] = 0$ and the third result follows. □

The central limit theorem behind the estimates for the statistical error in (13) assumes that Monte Carlo errors are normally distributed. Moreover, they are biased due to discretization. Lemma 4 makes this assumption explicit.

Lemma 4 *Let $u_h \sim u + \beta h^\delta + \sqrt{V[\phi]/N}\mathcal{N}$. Then $E[u_h - u] = \beta h^\delta$ and $E[(u_h - u)^2] = \beta^2 h^{2\delta} + V[\phi]/N$.*

Proof. The first moment is trivial. For the MSE, just notice that $E[(u_h - u)^2] = E[(\beta h^\delta + \sqrt{V[\phi]/N}\mathcal{N})^2] = \beta^2 h^{2\delta} + (V[\phi]/N)E[\mathcal{N}^2] + \beta\sqrt{V[\phi]/N}E[\mathcal{N}]$. □

The distribution of u_h defined in Lemma 4 results from combining the central limit theorem in (13) with the $MSE(\phi_h)$ in (14). It is natural enough and further justified by the match of the first two moments of the error, but rigourous only asymptotically as $h \to 0^+$.

In order to track errors from the nodal values into the subdomains, one needs to know how they are propagated by the interfacial interpolators. Definition 5 introduces the relevant notation and properties.

Definition 5 *Let λ, μ be real constants, $\{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$ a set of $r > 1$ distinct points in $\Gamma \subset \mathbb{R}^D$; $u_1, \ldots, u_r$ and $v_1, \ldots, v_r$ two sets of scalars associated to the points in Γ, and $z : \Gamma \mapsto \mathbb{R}$ a function. Let $R[z(\mathbf{x}_i) \,|\, \mathbf{x}_i \in \Gamma](\mathbf{x}) : \Gamma \mapsto \mathbb{R}$ be a smooth approximation to z obtained by interpolation of $\{z(\mathbf{x}_1), \ldots, z(\mathbf{x}_r)\}$. We say that R is a linear interpolator if*

$$R[\lambda u_i + \mu v_i \,|\, \mathbf{x}_i \in \Gamma](\mathbf{x}) = \lambda R[u_i \,|\, \mathbf{x}_i \in \Gamma](\mathbf{x}) + \mu R[v_i \,|\, \mathbf{x}_i \in \Gamma](\mathbf{x}), \qquad \mathbf{x} \in \Gamma. \tag{36}$$

For some norm $\|.\|$ in Γ we call the interpolation error

$$\epsilon_z := \|R[z(\mathbf{x}_i) \,|\, \mathbf{x}_i \in \Gamma] - z\|. \tag{37}$$

The most common interpolation schemes are linear in the sense of Definition 5–for instance, RBF interpolation [8].

Definition 6 *The error propagation function $w_k(\mathbf{x}; \boldsymbol{\omega})$ for the elliptic BVP (6) in the subdomain Ω_k $(1 \leq k \leq m)$ is defined as*

$$w_k(\mathbf{x}; \boldsymbol{\omega}) = \bar{w}_k(\mathbf{x}) + \frac{1}{q}\tilde{w}_k(\mathbf{x}; \boldsymbol{\omega}) \tag{38}$$

where $\bar{w}_k(\mathbf{x})$ and $\tilde{w}_k(\mathbf{x}; \boldsymbol{\omega})$ are respectively the solution of the deterministic BVP and of the BVP with stochastic BCs

$$\left\{\begin{array}{lll} L\bar{w}_k(\mathbf{x}) + c\bar{w}_k(\mathbf{x}) = L\tilde{w}_k(\mathbf{x}; \boldsymbol{\omega}) + c\tilde{w}_k(\mathbf{x}; \boldsymbol{\omega}) = 0, & \textit{if} & \mathbf{x} \in \Omega_k, \\ \bar{w}_k(\mathbf{x}) = \tilde{w}_k(\mathbf{x}; \boldsymbol{\omega}) = 0, & \textit{if} & \mathbf{x} \in \partial\Omega_k \cap \partial\Omega, \\ \left.\begin{array}{l} \bar{w}_k(\mathbf{x}) = R[\, sign(\beta_i) \,|\, \mathbf{x}_i \in \Gamma_j^k\,](\mathbf{x}) \\ \tilde{w}_k(\mathbf{x}; \boldsymbol{\omega}) = R[\, \mathcal{N}(\omega_i) \,|\, \mathbf{x}_i \in \Gamma_j^k\,](\mathbf{x}) \end{array}\right\} & \textit{if} & \mathbf{x} \in \Gamma_j^k,\ 1 \leq j \leq m_k. \end{array}\right. \tag{39}$$

Definition 7 *Let τ, σ, $\mathbf{X}_t$, Y_t, and $\mathbf{W}_t$ be the same as in (26). For a fixed $\boldsymbol{\omega} \in \mathbb{R}^n$,*

$$\psi(\boldsymbol{\omega}) := -\int_{t=0}^{\tau} \sigma^T(\mathbf{X}_t)\Big(\oplus_{k=1}^{m} \nabla\big(\bar{w}_k(\mathbf{X}_t) + \frac{1}{q}\tilde{w}_k(\mathbf{X}_t;\boldsymbol{\omega})\big)\Big) Y_t d\mathbf{W}_t, \tag{40}$$

Thanks to the smoothness of the iterfacial interpolator, gradients inside Ω_k are well defined, and thus the integral in (40) is also well defined regardless of the continuity of $\oplus_{k=1}^{m}\nabla w_k(\mathbf{x};\boldsymbol{\omega})$ across the interfaces. We are now prepared to state the main theoretical result.

Lemma 8 *Assume an elliptic BVP with Dirichlet BCs like (6) with exact solution $u(\mathbf{x})$, and let $\tilde{u}(\mathbf{x}, a;\boldsymbol{\omega})$ be a PDD simulation of it in the limit $h \to 0^+$, with accuracy $a > 0$, nodal statistical errors labeled by $\boldsymbol{\omega}$, balanced MC simulations, and smooth linear interpolator R. Let $\xi(a;\boldsymbol{\omega})$ be the control variate at a given interfacial node $\mathbf{x_0} \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ constructed from $\tilde{u}(\mathbf{x}, a;\boldsymbol{\omega})$ according to (26). Then*

$$V[\xi(a;\boldsymbol{\omega})] = V[\phi] - Cov[\phi, \psi(\boldsymbol{\omega})]\, a + V[\psi(\boldsymbol{\omega})]\Big(\frac{a}{2}\Big)^2 + O\Big(\sum_{k=1}^{m}\sum_{j=1}^{m_k} \epsilon_u|_{\Gamma_j^k}\Big) + O\Big(\sum_{k=1}^{m} \epsilon_{\Omega_k}(a)\Big) \tag{41}$$

and

$$\rho[\phi, \xi(a;\boldsymbol{\omega})] = -\sqrt{\frac{V[\phi]}{V[\xi(a;\boldsymbol{\omega})]}} + \frac{Cov[\phi, \psi(\boldsymbol{\omega})]}{\sqrt{V[\phi]V[\xi(a;\boldsymbol{\omega})]}}\frac{a}{2} + O\Big(\sum_{k=1}^{m}\sum_{j=1}^{m_k} \epsilon_u|_{\Gamma_j^k}\Big) + O\Big(\sum_{k=1}^{m} \epsilon_{\Omega_k}(a)\Big), \tag{42}$$

where $\psi(\boldsymbol{\omega})$ is the auxiliar variate defined by (40), $\epsilon_u|_\Gamma$ the error of interpolating u along an interface Γ, and ϵ_Ω is the error of the PDD subdomain solver.

Proof. Let us consider the integral representation of the solution of (6)

$$u(\mathbf{x}) = \int_\Omega G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d^D\mathbf{y} + \int_{\partial\Omega} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} g(\mathbf{y}) d^{D-1}\mathbf{y}, \tag{43}$$

where $\partial/\partial\mathbf{N} = \mathbf{N}\cdot\nabla$ and $G(\mathbf{x}, \mathbf{y})$ is Green's function, defined as the solution of

$$\begin{cases} LG(\mathbf{x}, \mathbf{y}) + cG(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}) & \text{if} \quad \mathbf{x} \in \Omega, \\ G(\mathbf{x}, \mathbf{y}) = 0 & \text{if} \quad \mathbf{x} \in \partial\Omega. \end{cases} \tag{44}$$

Under the adequate smoothness requirements on L and $\partial\Omega$ the solution $G(\mathbf{x}, \mathbf{y})$ to the homogeneous BVP (44) exists and is unique, which ensures the validity of (43) [7]. Consider the solution restricted to subdomain Ω_k. Since $G(\mathbf{x}, \mathbf{y})$ does not depend on the boundary data, $u|_{\Omega_k}(\mathbf{x})$ can also be represented as

$$u|_{\Omega_k}(\mathbf{x}) = \int_{\Omega_k} G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d^D\mathbf{y} + \int_{\partial\Omega_k \cap \partial\Omega} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} g(\mathbf{y}) d^{D-1}\mathbf{y} + \sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} u(\mathbf{y}) d^{D-1}\mathbf{y}. \tag{45}$$

On the other hand, the PDD subdomain approximation is affected by the error of the subdomain solver, $\epsilon_{\Omega_k}(a)$, which depends on a due to the BC along the interfaces:

$$\tilde{u}|_{\Omega_k}(\mathbf{x}, a; \boldsymbol{\omega}) = \int_{\Omega_k} G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d^D\mathbf{y} + \int_{\partial\Omega_k \cap \partial\Omega} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} g(\mathbf{y}) d^{D-1}\mathbf{y} + \\ + \sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} \tilde{u}(\mathbf{y}, a; \boldsymbol{\omega}) d^{D-1}\mathbf{y} + \epsilon_{\Omega_k}(a). \tag{46}$$

In the limit $h \to 0^+$, the nodal values are given by the distribution in Lemma 4. Running a PDD simulation with balanced MC simulations, tolerance a and probability P_q, is then equivalent to fixing $\boldsymbol{\omega}$ and taking

$$\tilde{u}(\mathbf{x}_0, a; \boldsymbol{\omega}) = u(\mathbf{x}_0) + \frac{a}{2} sign\Big(\beta(\mathbf{x}_0)\Big) + \frac{a}{2q} \mathcal{N}(\omega_0). \tag{47}$$

(Note that the statistical error, and hence $\boldsymbol{\omega}$, are only known after running the PDD simulation.) The Dirichlet BC condition on an interface Γ is then

$$\tilde{u}|_\Gamma(\mathbf{x}, a; \boldsymbol{\omega}) = R[\tilde{u}_i \,|\, \mathbf{x}_i \in \Gamma](\mathbf{x}) = u|_\Gamma(\mathbf{x}) + \frac{a}{2} R\Big[\, sign\Big(\beta(\mathbf{x}_i)\Big) + \frac{1}{q} \mathcal{N}(\omega_i) \,|\, \mathbf{x}_i \in \Gamma \,\Big](\mathbf{x}) + O(\epsilon_u|_\Gamma). \tag{48}$$

Inserting (48) into (46):

$$\tilde{u}|_{\Omega_k}(\mathbf{x}, a; \boldsymbol{\omega}) = u|_{\Omega_k}(\mathbf{x}) + \frac{a}{2} \sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} R\Big[\, sign\Big(\beta(\mathbf{x}_i)\Big) + \frac{1}{q} \mathcal{N}(\omega_i) \,|\, \mathbf{x}_i \in \Gamma_j^k \,\Big](\mathbf{y}) d^{D-1}\mathbf{y} + \\ + O\Big(\sum_{j=1}^{m_k} \epsilon_u|_{\Gamma_j^k} \Big) + \epsilon_{\Omega_k}(a). \tag{49}$$

Since R is a smooth interpolator, gradients are well defined inside Ω_k and along the interfaces, but not necessarily across them. In order to circumvent this issue we take the direct sum of subdomain gradients,

$$\oplus_{k=1}^m \nabla\tilde{u}(\mathbf{x}, a; \boldsymbol{\omega}) = \oplus_{k=1}^m \nabla u(\mathbf{x})|_{\Omega_k} + \frac{a}{2} \mathbf{V}(\mathbf{x}; \boldsymbol{\omega}) + O\Big(\sum_{k=1}^m \sum_{j=1}^{m_k} \epsilon_u|_{\Gamma_j^k} \Big) + O\Big(\epsilon_{\Omega_k}(a)\Big), \tag{50}$$

where

$$\mathbf{V}(\mathbf{x}; \boldsymbol{\omega}) = \oplus_{k=1}^m \nabla \Bigg(\sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial \mathbf{N}} R\Big[\, sign\Big(\beta(\mathbf{x}_i)\Big) + \frac{1}{q} \mathcal{N}(\omega_i) \,|\, \mathbf{x}_i \in \Gamma_j^k \,\Big](\mathbf{y}) d^{D-1}\mathbf{y} \Bigg). \tag{51}$$

Note that the quantity in parentheses in (51) is the Green representation of the BVP with solution $w_k(\mathbf{x}; \boldsymbol{\omega}) = \bar{w}_k(\mathbf{x}) + \frac{1}{q}\tilde{w}_k(\mathbf{x}; \boldsymbol{\omega})$ (38), by linearity. Now, according to definitions (26) and (40) for node $\mathbf{x}_0$,

$$\xi(a;\boldsymbol{\omega}) = \xi + \frac{a}{2}\psi(\boldsymbol{\omega}) + O\Big(\sum_{k=1}^{m}\sum_{j=1}^{m_k}\epsilon_u|_{\Gamma_j^k}\Big) + \sum_{k=1}^{m}\epsilon_{\Omega_k}(a). \tag{52}$$

In (52), we have used the fact that $\oplus_{k=1}^{m}\nabla u|_{\Omega_k} = \nabla u$ except on the interfaces. Since the interfaces are smooth and the random paths are not, the trajectories cross the interfaces at isolated points which make no contribution to the integral, so that $-\int_{t=0}^{\tau}\boldsymbol{\sigma}^T\big(\oplus_{k=1}^{m}\nabla u|_{\Omega_k}\big)Y_t d\boldsymbol{W}_t = \xi$. Taking the variance of (52) and using Lemma 3 yields (41). Moreover,

$$Cov[\phi,\xi(a;\boldsymbol{\omega})] = Cov[\phi,\xi] + \frac{a}{2}Cov[\phi,\psi(\boldsymbol{\omega})] + O\Big(\sum_{k=1}^{m}\sum_{j=1}^{m_k}\epsilon_u|_{\Gamma_j^k}\Big) + O\Big(\sum_{k=1}^{m}\epsilon_{\Omega_k}(a)\Big). \tag{53}$$

Finally, equation (42) follows from $\rho[\phi,\xi(a;\boldsymbol{\omega})] = Cov[\phi,\xi(a;\boldsymbol{\omega})]/\sqrt{V[\phi]V[\xi(a;\boldsymbol{\omega})]}$ recalling that the correlation with the exact control variate is -1 by Lemma 3. □

The point of Lemma 8 is to predict the correlation between the score ϕ and an approximate control variate $\xi(a;\boldsymbol{\omega})$ without actually running a PDD simulation to produce the latter–but rather "simulating" it. In exchange, the variable $\psi(\boldsymbol{\omega})$ must be computed, but it can be constructed on a subdomain-per-subdomain basis, i.e. in a fully parallellizable way. The drawback is that the formulas derived so far are only rigourous at $h = 0$, and that they depend on quite a few problem-dependent constants–many of them node-dependent as well. These difficulties will be addressed in Section 5. But before that, we shall examine the issues of global error and stability of PDD, and introduce some more results which will later be useful in assessing the necessary simplifications.

4 Global error and stability of PDD

The main tool is Theorem 3.7 in Gilbarg and Trudinger [10]:

Theorem 9 *Let v be the solution of an elliptic BVP like (6) with $c \le 0$, such that v is continuous on $\Omega\backslash\partial\Omega$ and twice differentiable on $\partial\Omega$. Then*

$$\sup_{\mathbf{x}\in\Omega}|v| \le \sup_{\mathbf{x}\in\partial\Omega}|g| + Q\sup_{\mathbf{x}\in\Omega}\frac{|f|}{\lambda} \tag{54}$$

where Q is a constant depending only on diam(Ω) and $\sup_\Omega\|\mathbf{b}\|_2/\lambda$. In particular, if Ω lies between two parallel planes a distance d apart, then (54) is satisfied with $Q = \exp\left[(\sup_\Omega\|\mathbf{b}\|_2/\lambda)/d\right] - 1$.

Recall that $\lambda = \inf_{\mathbf{x}\in\Omega} \lambda_{min}[A(\mathbf{x})]$, and $diam(\Omega)$ is the largest distance between two points in Ω. It is convenient to introduce the following parameter:

Definition 10 *Let R be a linear interpolator and Γ a subdomain interface with nodes $\mathbf{x}_i \in \Gamma$, $1 \le i \le p$. Let $z_1, \ldots, z_p$ be p scalars such that $|z_i| = 1$. The overshoot constant of R with respect to the discretization $\mathbf{x}_1, \ldots, \mathbf{x}_p$ of Γ is defined as*

$$\gamma_R^\Gamma := \sup_{z_1,\ldots,z_p} \sup_{\mathbf{x}\in\Gamma} \Big| R[z_i|\mathbf{x}_i \in \Gamma](\mathbf{x}) \Big|. \tag{55}$$

Analogously, let $\gamma_R^{\Omega_k} := \sup_{\Gamma\in\Omega_k} \gamma_R^\Gamma$ and $\gamma_R := \sup_{1\le k\le m} \gamma_R^{\Omega_k}$.

The overshoot constant measures the excess of the reconstructed function over any of the interpolated values which are 1 in absolute value. For piecewise interpolation, $\gamma_R = 1$, but for more accurate interpolators, $\gamma_R > 1$ due to the Runge and Gibbs phenomena [8]–see Figure 3 (right) for illustration.

We are interested in bounding the largest PDD error throughout Ω. Assuming as always balanced MC simulations, and neglecting the error of the subdomain solver, the PDD error in subdomain Ω_k obeys

$$\begin{cases} L(\tilde{u}-u)|_{\Omega_k} + c(\tilde{u}-u)|_{\Omega_k} = 0 & \text{if } \mathbf{x} \in \Omega_k, \\ (\tilde{u}-u)|_{\Omega_k} = 0 & \text{if } \mathbf{x} \in \partial\Omega_k \cap \partial\Omega, \\ (\tilde{u}-u)|_{\Omega_k} = \frac{a}{2} R[sign(\beta_i) + \frac{1}{q}\mathcal{N}(\omega_i)\,|\,\mathbf{x}_i \in \Gamma_j^k](\mathbf{x}) & \text{if } \mathbf{x} \in \Gamma_j^k,\ 1 \le j \le m_k. \end{cases} \tag{56}$$

Therefore, $(\tilde{u}-u)|_{\Omega_k} = (\tilde{u}_k - u|_{\Omega_k}) = \frac{a}{2} w_k(\mathbf{x};\boldsymbol{\omega})$–hence the name of error propagation function. Applying Theorem 9 to (56) and by linearity of R,

$$\Big|\tilde{u}_k - u|_{\Omega_k}\Big| \le \frac{aQ_k}{2} \sup_{\mathbf{x}\in\partial\Omega_k\setminus\partial\Omega} \Big| \oplus_{\Gamma_j^k\in\partial\Omega_k} R[sign(\beta_i) + \frac{1}{q}\mathcal{N}(\omega_i)\,|\,\mathbf{x}_i \in \Gamma_j^k](\mathbf{x}) \Big| \le \frac{aQ_k\gamma_R^{\Omega_k}}{2}\Big(1 + \frac{1}{q}\sup_{\mathbf{x}_i\in\Omega_k} |\mathcal{N}(\omega_i)|\Big), \tag{57}$$

where Q_k depends on the size and shape of Ω_k and $L|_{\Omega_k}$. Thanks to the symmetry of the standard normal distribution around zero, $\sup_{1\le j\le s} |\mathcal{N}(\omega_j)| = 2\sup_{1\le j\le s} \mathcal{N}(\omega_j)$, so that the *global* PDD error (neglecting the error of the subdomain solver) can be bounded by

$$|\tilde{u} - u| \le \sup_{1\le k\le m} \frac{a\gamma_R^{\Omega_k} Q_k}{2}\Big(1 + \frac{2}{q}\sup_{\mathbf{x}_i\in\partial\Omega_k} \mathcal{N}(\omega_i)\Big). \tag{58}$$

Equation (58) reflects that the global PDD error depends on PDE coefficients in (6), on the shape and size $|\Omega_k|$ of the subdomains (typically, $|\Omega|/m$), and on the number of nodes per subdomain (typically, around n/m), rather than on $|\Omega|$ and n. Therefore, PDD is intrinsically stable provided that the number, shape and discretization of the subdomains are so chosen that the subdomain errors are controlled.

We will now derive the nodal tolerance $a_0(\epsilon)$ required to enforce a set global PDD error tolerance $\epsilon > 0$. Recall that n_k is the total number of interfacial nodes

sitting on $\partial\Omega_k$. The distribution $\sup_{\mathbf{x}_i\in\Omega_k} \mathcal{N}(\omega_i) = \sup_{1\le j\le n_k} \mathcal{N}(\omega_j)$ is an *extreme value* distribution. The value n_k for which its maximum is less than x with probability P_q is to be extracted from

$$x = CDF[\sup_{1\le j\le n_k} \mathcal{N}(\omega_j)]^{-1}(P_q) \text{ (such that } Pr[\sup_{1\le j\le n_k} \mathcal{N}(\omega_j) < x] = P_q.) \tag{59}$$

where $CDF[\cdot]$ and $CDF[\cdot]^{-1}$ stand for the cumulative distribution function (CDF) of a given distribution and its inverse, respectively. Let

$$k_{max} = \arg\max_{1\le k\le m} \frac{a\gamma_R^{\Omega_k} Q_k}{2}\Big(1 + \frac{2}{q} CDF[\sup_{\mathbf{x}_i\in\partial\Omega_k} \mathcal{N}(\omega_i)]^{-1}(P_q)\Big) \tag{60}$$

and define

$$Q_{max} = Q_{k_{max}} \gamma_R^{\Omega_{k_{max}}}/\gamma_R, \qquad s = n_{k_{max}}, \qquad S_s = \sup_{1\le j\le s} \mathcal{N}(\omega_j) \text{ (\textit{i.i.d.}).} \tag{61}$$

Since the s standard normals in S_s are i.i.d., the CDF is

$$CDF[S_s](x) = \int_{-\infty}^{x} S_s(t)dt = Pr[\Big(\mathcal{N}_1 \le x\Big) \cap \ldots \cap \Big(\mathcal{N}_s \le x\Big)] = \Big(CDF[\mathcal{N}](x)\Big)^s. \tag{62}$$

For the standard normal distribution

$$CDF[\mathcal{N}](x) = \frac{1}{2\pi}\int_{-\infty}^{x} e^{-t^2}dt = \frac{1}{2}\Big(1 + \mathrm{erf}(x/\sqrt{2})\Big), \tag{63}$$

where $\mathrm{erf}(x) = (2/\sqrt{\pi})\int_0^x e^{-t^2}dt$ is the error function. The nodal target tolerance a_0 can then be related to the global PDD error tolerance by

$$\epsilon = \frac{a_0\gamma_R Q_{max}}{2}\Big(1 + \frac{2\sqrt{2}}{q}\mathrm{erf}^{-1}(2P_q^{1/s} - 1)\Big). \tag{64}$$

As Figure 2 shows, $a_0/\epsilon = a_0(\epsilon = 1)$ depends very mildly on the typical number of nodes per subdomain. Anyway, the bound (64) will be a large overestimation in many cases, for the effect of the statistical errors decays fast away from the interfaces. If the moments of $a(\epsilon)$ were required, a useful fact is that as s grows, S_s tends to the Gumbel distribution

$$\lim_{s\to\infty} CDF[\frac{S_s - l_s}{b_s}](x) = G\Big(\frac{x - l_s}{b_s}\Big) := \exp[-e^{-x}] \tag{65}$$

with location and scaling parameters $l_s = -CDF[\mathcal{N}]^{-1}(1/s)$ and $b_s = 1/l_s$.

5 Approximations leading to a practical algorithm

In order to construct an implementable multigrid-like IterPDD algorithm, several approximations are needed, listed as heuristics **H1** through **H5** below.

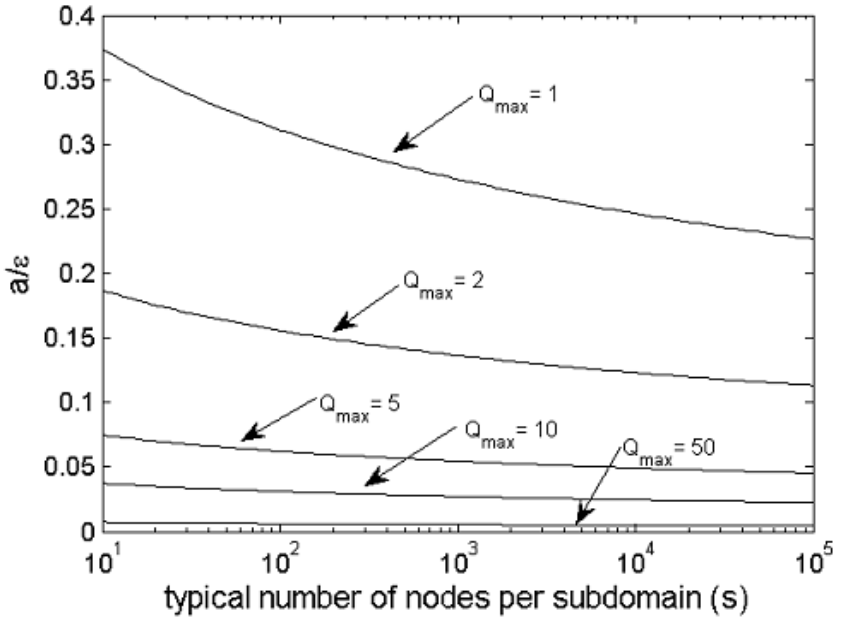

Figure 2: The ratio a_0/ϵ from (64) as a function of s for several values of Q_{max} and $q = 2, \gamma_R = 1.5$ (notice the different scales).

H1: a_0 and $a > a_0$ are small enough

This assumption is used on several occasions in **H2**-**H5**.

H2: Interpolation and subdomain-solver errors will be dropped

Interfacial interpolation errors in (41) and (42) will be dropped. Then, inserting (41) into (42) yields, after some manipulation,

$$1 - \rho^2[\phi, \xi(a;\boldsymbol{\omega})] \gtrsim \frac{\frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}\Big(1 - \rho^2[\phi, \psi(\boldsymbol{\omega})]\Big)(a/2)^2}{1 - \sqrt{\frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}}\rho[\phi, \psi(\boldsymbol{\omega})]a + \frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}(a/2)^2}, \tag{66}$$

where the sign $\gtrsim$ has replaced the equality to make up for dropping the interpolation and subdomain-solver errors. Next, note that

$$\frac{V[\phi + \xi(a;\boldsymbol{\omega})]}{V[\phi]} \gtrsim 1 - \rho^2[\phi, \xi(a;\boldsymbol{\omega})], \tag{67}$$

since (24) only strictly holds for the minimizer of (23), and with $a > 0$, the PDD solution $\tilde{u}$ yields a control variate off the minimizer, regardless of h. Note also that the denominator in (66) is positive, since $|\rho[\phi, \psi(\boldsymbol{\omega})]| \leq 1$:

$$1 - \sqrt{\frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}}\rho[\phi,\psi(\boldsymbol{\omega})]a + \frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}(a/2)^2 \geq \left(1 - \sqrt{\frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}}(a/2)\right)^2. \quad (68)$$

For small a (see **H1**), the denominator in (66) can be dropped. More precisely, since the MC simulations are balanced,

$$\sqrt{\frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}}\frac{a}{2} = q\sqrt{\frac{V[\psi(\boldsymbol{\omega})]}{N}}, \quad (69)$$

which is negligible if $V[\psi(\boldsymbol{\omega})] << N$–recall that this $N >> 1$ is meant without variance reduction, and $V[\psi(\boldsymbol{\omega})] = O(V[\bar{\psi}])$ is bounded by (82) in **H4** below. Assuming this and putting all together, it holds

$$\frac{V[\phi + \xi(a;\boldsymbol{\omega})]}{V[\phi]} \gtrsim \frac{V[\psi(\boldsymbol{\omega})]a^2}{4V[\phi]}\Big(1 - \rho^2[\phi,\psi(\boldsymbol{\omega})]\Big). \quad (70)$$

The importance of (70) is that the two factors affecting IterPDD–namely a and the random statistical errors on the nodes–have been separated. Moreover, the latter has been expressed in terms of $V[\psi(\boldsymbol{\omega})]$ and $\rho[\phi,\psi(\boldsymbol{\omega})]$. Also,

- If the score ϕ and the auxiliar variate $\psi(\boldsymbol{\omega})$ were perfectly correlated (i.e. $\rho^2[\phi,\psi(a,\boldsymbol{\omega})] = 1$), then $V = [\phi + \psi(a,\boldsymbol{\omega})] = 0$, meaning that $\psi(a,\boldsymbol{\omega}) = k\xi$. Since this can only happen if $f = g = 0$, the solution would be $u = 0$.
- The opposite limit, $\rho[\phi,\psi(a,\boldsymbol{\omega})] = 0$, yields asymptotically

$$\lim_{a\to 0^+}\frac{V[\xi(a;\boldsymbol{\omega})]}{V[\phi]} = \frac{V[\psi(\boldsymbol{\omega})]}{V[\phi]}(a/2)^2. \quad (71)$$

H3: Small variance with respect to ω

The propagation of nodal statistical errors (labeled with $\boldsymbol{\omega}$) onto the subdomain solutions in critical in PDD. In Section 4, it was shown that the effect of $\boldsymbol{\omega}$ on the PDD aggregate error can be controlled on a subdomain-per-subdomain basis. In IterPDD(a_0, a_1), there are two further aspects to $\boldsymbol{\omega}$. First, how much the variance reduction produced by $\xi(a_1;\boldsymbol{\omega})$ depends on the chance variable. Second, how it affects the decrease predicted by (70)–for a_1 will be determined based on that formula. In particular, the nodal simulations will in turn be "simulated" themselves by randomly drawing a chance variable–say $\boldsymbol{\omega}'$–and computing $V[\psi(\boldsymbol{\omega}')]$ and $\rho[\phi,\psi(\boldsymbol{\omega}')]$.

Lemma 11 *Let $E_\omega[\cdot]$ be the expected value relative to $\boldsymbol{\omega}$ and assume that interpolation errors are negligible. Then, $E_\omega[\tilde{w}_k(\mathbf{x};\boldsymbol{\omega})] = 0$ $(1 \leq k \leq m)$.*

Proof. The Green function representation of $\tilde{w}_k$ is

$$\tilde{w}_k(\mathbf{x};\boldsymbol{\omega}) = \sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x},\mathbf{y})|_{\Omega_k}}{\partial \mathbf{N}} R[\mathcal{N}(\omega_i) \,|\, \mathbf{x}_i \in \Gamma_j^k](\mathbf{y}) d^{D-1}\mathbf{y}, \tag{72}$$

where $G(\mathbf{x},\mathbf{y})$ is determined by (44) and is deterministic. By linearity of the integral, the interpolator, and the expected value,

$$E_{\boldsymbol{\omega}}[\tilde{w}_k(\mathbf{x},\boldsymbol{\omega})] = \sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x},\mathbf{y})|_{\Omega_k}}{\partial \mathbf{N}} R\Big[E_{\boldsymbol{\omega}}[\mathcal{N}(\omega_i)] \,|\, \mathbf{x}_i \in \Gamma_j^k\Big](\mathbf{y}) d^{D-1}\mathbf{y} = \tag{73}$$

$$\sum_{j=1}^{m_k} \int_{\Gamma_j^k} \frac{\partial G(\mathbf{x},\mathbf{y})|_{\Omega_k}}{\partial \mathbf{N}} \Big(0 + \epsilon_0|_{\Gamma_j^k}\Big) d^{D-1}\mathbf{y},$$

where $\epsilon_0|_{\Gamma_j^k}$ is the error in the reconstruction of the constant function $z = 0$ on the interface Γ_j^k with R, and which is zero by hypothesis. □

As a consequence of Lemma 11, $E_{\boldsymbol{\omega}}[\nabla \tilde{w}_k] = \nabla E_{\boldsymbol{\omega}}[\tilde{w}_k] = 0$, and thus

$$\bar{\psi} := E_{\boldsymbol{\omega}}[\psi(\boldsymbol{\omega})] = \int_0^\tau Y_t \sigma^T \sum_{k=1}^{m} \nabla\Big(\bar{w}_k + \frac{1}{q} E_{\boldsymbol{\omega}}[\tilde{w}_k]\Big) d\mathbf{W}_t = \int_0^\tau Y_t \sigma^T \sum_{k=1}^{m} \nabla \bar{w}_k d\mathbf{W}_t. \tag{74}$$

The variance can be calculated by Ito's isometry:

$$V[\psi(\boldsymbol{\omega})] = E\Big[\int_0^\tau Y_t^2(\mathbf{X}_t) \|\sigma^T(\mathbf{X}_t) \sum_{k=1}^{m} \nabla w_k(\mathbf{X}_t,\boldsymbol{\omega})\|_2^2 \, dt \Big], \tag{75}$$

where $Y_t = \exp \int_0^t c(\mathbf{X}_s)\, ds$. By the triangular inequality and the inequalities

$$\|\sigma^T \nabla \tilde{w}_k\|_2 \le \|\sigma^T\|_2 \cdot \|\nabla \tilde{w}_k\|_2 \tag{76}$$

and

$$\|\sigma^T(\mathbf{x})\|_2 = \sqrt{\lambda_{max}\Big(\sigma(\mathbf{x})\sigma^T(\mathbf{x})\Big)} = \sqrt{2\lambda_{max}\Big(A(\mathbf{x})\Big)} \le \sqrt{2\Lambda}, \text{ if } \mathbf{x} \in \Omega, \tag{77}$$

one has

$$E_{\boldsymbol{\omega}}\Big[V[\psi(\boldsymbol{\omega})] \Big] \le V[\bar{\psi}] + \frac{2\Lambda\, E\Big[\int_0^\tau Y_t^2(\mathbf{X}_t) dt\Big]}{q^2} E_{\boldsymbol{\omega}}\Big[\Big(\sup_{\mathbf{x}\in\Omega} \| \sum_{k=1}^{m} \nabla \tilde{w}_k(\mathbf{x};\boldsymbol{\omega})\|_2^2 \Big) \Big]. \tag{78}$$

To the best of the authors' knowledge, there are no interior, a priori estimates of the gradient of the solution of (6) (with $f = 0$) as sharp as the bound provided by (9) for the solution itself. Therefore, based on more particular results such

as [10, Theorem 3.9] and [10, Problem 3.6], we make the reasonable assumption that for the PDE $(L+c)u = 0$ in Ω_k with Dirichlet BCs on $\partial\Omega_k$

$$\sup_{\mathbf{x}\in\Omega_k} \|\nabla u(\mathbf{x})\|_2 \leq K'_k + K''_k \sup_{\mathbf{x}\in\partial\Omega_k} |u(\mathbf{x})|, \tag{79}$$

where K'_k and K''_k are positive and may depend on anything but the value of the Dirichlet BC. Then, there exist positive constants K', K'' and s such that

$$\left.\begin{array}{l} \sum_{k=1}^{m} \|\bar{w}_k(\mathbf{x})\|_2 \leq K' + \gamma_R K'' \\ \sum_{k=1}^{m} \|\tilde{w}_k(\mathbf{x};\boldsymbol{\omega})\|_2 \leq K' + 2\gamma_R K'' \sup_{1\leq j\leq s} \mathcal{N}(\omega_j) \end{array}\right\} \text{ if } \mathbf{x}\in\Omega, \tag{80}$$

The values (K', K'') are the (K'_k, K''_k) of the subdomain k with the largest gradient estimate, and s its number of nodes. The γ_R shows up due to the interpolation along the interfaces; and $2\sup_j \mathcal{N}(\omega_j) = \sup_j |\mathcal{N}(\omega_j)|$. This leads to the bounds

$$V[\psi(\boldsymbol{\omega})] \leq V[\bar{\psi}] + 2\Lambda E[\int_0^\tau Y_t^2 dt]\frac{1}{q^2}\Big(K' + 2\gamma_R K'' \sup_{1\leq j\leq s} \mathcal{N}(\omega_j)\Big)^2 =: V[\bar{\psi}] + \tilde{v}(\boldsymbol{\omega}), \tag{81}$$

$$V[\bar{\psi}] \leq 2\Lambda E[\int_0^\tau Y_t^2 dt](K' + \gamma_R K'')^2 =: \bar{v}. \tag{82}$$

As the final preparatory step, let us calculate the noise-to-signal ratio (NSR)–defined as $NSR[\cdot] = \sqrt{V[\cdot]}/E[\cdot]$–of the variable $\bar{v} + \tilde{v}(\boldsymbol{\omega})$ (83):

$$\frac{\sqrt{V_{\boldsymbol{\omega}}[\bar{v} + \tilde{v}(\boldsymbol{\omega})]}}{E_{\boldsymbol{\omega}}[\bar{v} + \tilde{v}(\boldsymbol{\omega})]} = \frac{\sqrt{V_{\boldsymbol{\omega}}\Big[\Big(1 + 2\gamma_R \frac{K'}{K''} \sup_{1\leq j\leq s} \mathcal{N}(\omega_j)\Big)^2\Big]}}{q^2\Big(1 + \gamma_R \frac{K'}{K''}\Big)^2 + E_{\boldsymbol{\omega}}\Big[\Big(1 + 2\gamma_R \frac{K'}{K''} \sup_{1\leq j\leq s} \mathcal{N}(\omega_j)\Big)^2\Big]}. \tag{83}$$

We are finally in a position to precisely estate our claim and the supporting heuristic. The ultimate goal is to use the deterministic value $V[\bar{\psi}]$ instead of a random $V[\psi(\boldsymbol{\omega})]$ yielded by one simulation (note that $V[\bar{\psi}] \leq E_{\boldsymbol{\omega}}\Big[V[\psi(\boldsymbol{\omega})]\Big]$). Therefore, it is important that the ratio $\sqrt{V_{\boldsymbol{\omega}}\Big[V[\psi(\boldsymbol{\omega})]\Big]}/E_{\boldsymbol{\omega}}\Big[V[\psi(\boldsymbol{\omega})]\Big]$ be small so that $V[\psi(\boldsymbol{\omega} = 0)] := V[\bar{\psi}] \approx V[\psi(\boldsymbol{\omega})]$. This ratio is problem-dependent but, in order to provide a rough estimate, we substitute $V[\psi(\boldsymbol{\omega})]$ by its upper bound $\bar{v} + \tilde{v}(\boldsymbol{\omega})$. Then, we simulate the NSR of the bound (Table 1) for realistic values $\gamma_R = 1, 2$ and over a broad range of K''/K' (which captures the effect of the L, c, the geometry Ω, and the PDD partition $\{\Omega_k\}_{k=1}^m$), and the typical number of nodes per subdomain, s. Since the NSR of the proxy is negligible in most of the scenarios (and specially as s grows), we argue that the same should hold for $V[\psi(\boldsymbol{\omega})]$. Obviously, specific problems would allow for sharper estimates.

	nodes per subdomain (s)						nodes per subdomain (s)				
K'/K''	10	10^2	10^3	10^4	10^5	K'/K''	10	10^2	10^3	10^4	10^5
0	.54	.31	.20	.15	.12	0	.54	.31	.20	.15	.12
10^{-2}	.54	.31	.20	.15	.12	10^{-2}	.54	.31	.21	.15	.12
10^{-1}	.51	.29	.20	.15	.12	10^{-1}	.53	.30	.20	.15	.12
1	.30	.21	.15	.12	.10	1	.40	.25	.18	.13	.11
10	.048	.037	.031	.027	.024	10	.094	.073	.060	.052	.046
10^2	.0047	.0035	.0029	.0025	.0023	10^2	.0094	.0070	.0059	.0051	.0046
	$\gamma_R = 1$						$\gamma_R = 2$				

Table 1: Simulated NSR over 10^5 realizations of formula (83) for $q = 2$. If $K' = 0$ the NSR is independent of $\gamma_R K''$.

H4: Loss of correlation due to discretization

Given a_0 and its corresponding timestep h_0 from (29), the correlation between ϕ and $\xi(a;\omega)$ is better than between their discretized counterparts ϕ_{h_0} and $\xi_{h_0}(a;\omega)$. This leads to an overestimation of the predicted decrease of variance and has a significant effect, especially if a and a_0 are comparable, or if $\rho[\phi,\xi] \gtrsim .99$. We invoke **H1** to justify the following perturbative ansatz:

$$Cov[\phi_{h_0}, \xi_{h_0}(a;\boldsymbol{\omega})] \approx Cov[\phi, \xi(a;\boldsymbol{\omega})] + B_\Xi\Big(Cov[\phi_{h_0}, \xi_{h_0}]\Big), \tag{84}$$

where $B_\Xi\Big(Cov[\phi_{h_0}, \xi_{h_0}]\Big)$ is the discrete covariance bias. Since $Cov[\phi_{h_0}, \xi_{h_0}] = E[\phi_{h_0}\xi_{h_0}] - u_{h_0}E[\xi_{h_0}])]$ and h_0 is small enough, $B_\Xi\Big(Cov[\phi_{h_0}, \xi_{h_0}]\Big) = \boldsymbol{O}(h_0^\delta)$. That covariance bias does not seem accessible without having ξ_{h_0}, but we can use the fact that $Cov[\phi,\xi] = -V[\phi]$ (Lemma 3) to argue that

$$\Big|B_\Xi\Big(Cov[\phi_{h_0}, \xi_{h_0}]\Big)\Big| \approx \Big|B_\Xi\Big(V[\phi_{h_0}]\Big)\Big| =: |\alpha| h^\delta, \tag{85}$$

which can be extracted from a fit (see **H5**). Then, assuming further that

$$\frac{1}{V[\phi_{h_0}]} \approx \frac{1}{V[\phi]} \qquad \text{and} \qquad \frac{1}{V[\xi_{h_0}(a;\boldsymbol{\omega})]} \approx \frac{1}{V[\xi(a)]}, \tag{86}$$

the squared correlation of the discretized variables is

$$\rho^2[\phi_{h_0}, \xi_{h_0}(a;\boldsymbol{\omega})] \approx \rho^2[\phi, \xi(a;\boldsymbol{\omega})] - \frac{2\Big|\alpha\rho[\phi, \xi(a;\boldsymbol{\omega})]\Big|}{V[\phi]} h_0^\delta, \tag{87}$$

and since $|\beta| h_0^\delta = a_0/2$, the effective variance reduction in IterPDD(a_0, a) is

$$\frac{V[\phi_{h_0} + \xi_{h_0}(a;\boldsymbol{\omega})]}{V[\phi_{h_0}]} \approx \frac{V[\phi + \xi(a;\boldsymbol{\omega})]}{V[\phi]} + \Big|\frac{\alpha\rho[\phi, \xi(a)]}{\beta V[\phi]}\Big| a_0. \tag{88}$$

For a_0 small enough, as $a \to a_0^+$, $|\rho[\phi, \xi(a,\boldsymbol{\omega})]| \to 1^-$, so (88) is capped by

$$\lim_{a \to a_0^+} \frac{V[\phi_{h_0} + \xi_{h_0}(a;\boldsymbol{\omega})]}{V[\phi_{h_0}]} = \Big|\frac{\alpha}{\beta V[\phi]}\Big| a_0 = \frac{2|B_\Xi(V[\phi])|}{V[\phi]}. \tag{89}$$

This makes intuitively sense, because the variance of the score can hardly drop below its discretization error, even with an exact control variate.

H5: Fast estimation of constants

In order to apply the sensitivity formula (93) and Algorithms 4 and 3, a number of constants must be estimated. Here, we describe a fast way of accomplishing this. We stress the fact that any Monte Carlo simulation (whether or not related to PDD) also would require δ, β and $V[\phi]$ in order to enforce a set error tolerance.

Global constants (δ and κ). As already discussed, with smooth BVPs, integrators with at least approximately known δ should be available [6]. The constant κ can easily be found by comparing the time taken by the computer to complete a number of visits to the implementations of (16) with and without **F**.

Nodal constants related to first moments. They are $E[\tau]$ (needed for K_i), $Cov[\phi, \bar{\psi}]$, and β. We consider a generic discretized random variable $\eta_h = \{\tau_h, \bar{\psi}_h\phi_h, \phi_h\}$, and assume that its first moment obeys the noisy model

$$E[\eta_h] \sim E[\eta_0] + \mathcal{N}(B^{(1)}h^\delta, V[\eta_0]). \tag{90}$$

Let $h_1 > h_2 > \ldots > h_{\hat{M}}$ be a 'cloud' of $\hat{M}$ equispaced timesteps. Set a number $\hat{N}$ and let $\hat{E}[\eta_h]$ be the mean of $\hat{N}$ realizations of η_h. After computing $\hat{M}$ MC independent simulations, the cloud of data $\left(h_1, \hat{E}[\eta_{h_1}]\right), \ldots, \left(h_{\hat{M}}, \hat{E}[\eta_{h_{\hat{M}}}]\right)$ is fitted to the noisy model (90) in order to extract $E[\eta_0]$ and $B^{(1)}$. (In order to do so, $V[\eta_0]$ in (90) can be replaced by the mean of the sample variances $\hat{V}[\eta_{h_1}, \ldots, \hat{V}[\eta_{h_{\hat{M}}}]$.)

Here, we provide a rougher recipe to carry out the fit with Matlab (see also [16]). For this purpose, it is convenient to think of model (90) as a member of the generalized linear model (GLM) family. Since $E[\mathcal{N}(B^{(1)}h^\delta, V[\eta_0])] = B^{(1)}h^\delta$, the link is the identity and the fit is readily carried out by issuing the Matlab command

```
Coeff= glmfit((h.^delta),Eh,'normal','link','identity')
```

where `h` and `Eh` above are Matlab arrays with respectively $(h_1, ..., h_{\hat{M}}$ and $(\hat{E}[\eta_{h_1}], ..., \hat{E}[\eta_{h_{\hat{M}}}])$; and the components of the output, `Coeff(1)` and `Coeff(2)`, are the fitted values to $E[\eta_0]$ and $B^{(1)}$, respectively (check the Matlab documention for getting error bounds alongside).

By applying this recipe, one gets approximations to the following quantities: if $\eta_h = \bar{\psi}_h\phi_h$, to $Cov[\phi, \bar{\psi}] \approx \hat{Cov}[\phi, \bar{\psi}]$ = `Coeff(1)` (and to its bias as a byproduct); and if $\eta_h = \phi_h$, to $\beta \approx \hat{\beta}$ = `Coeff(2)` and to $E[\phi] \approx \hat{E}[\phi]$ = `Coeff(1)` as a byproduct. The values $\hat{E}[\tau_{h_1}], ..., \hat{E}[\tau_{h_{\hat{M}}}]$ are obtained along the means $\hat{E}[\phi_{h_1}], ..., \hat{E}[\phi_{h_{\hat{M}}}]$ (see last line in Algorithm 1), and are used to fit $E[\tau]$.

Nodal constants related to second moments. Regarding the fitting of variances, it is well-known that variances of i.i.d. Gaussian distributions obey

a scaled chi-squared PDF [15]. Accommodating the discretization bias, the appropriate noisy model is

$$V[\eta_h] \sim B^{(2)}h^\delta + \frac{V[\eta_0]}{\hat{N}-1}\chi^2_{\hat{N}-1} = B^{(2)}h^\delta + \Gamma\Big(\frac{\hat{N}-1}{2}, \frac{2V[\eta_0]}{\hat{N}-1}\Big) \tag{91}$$

where $\chi^2_{\hat{N}-1}$ is the Chi-squared distribution with $\hat{N}-1$ degrees of freedom, and $\Gamma(p_1, p_2)$ is a Gamma distribution with shape parameter $p_1 = (\hat{N}-1)/2$ and scale parameter $p_2 = 2V[\eta_0]/(\hat{N}-1)$. Then, (91) can be identified with a GLM where the noise is Gamma and the link function is the identity, since

$$E\Big[V[\eta_h]\Big] = B^{(2)}h^\delta + E\Big[\Gamma\Big(\frac{\hat{N}-1}{2}, \frac{2V[\eta_0]}{\hat{N}-1}\Big)\Big] = B^{(2)}h^\delta + V[\eta_0]. \tag{92}$$

Let $\hat{V}[\eta_h] = \sum_{j=1}^{\hat{N}}(\eta_h^{(j)} - \hat{E}[\eta_h])^2\big/(\hat{N}-1)$ be the sample variance. After filling the Matlab array `Vh` with $\hat{V}[\phi_{h_1}], \ldots, \hat{V}[\phi_{h_{\hat{M}}}]$, issuing the command

```
Coeff= glmfit((h.^delta),Vh,'gamma','link','identity')
```

yields `Coeff(1)`$\approx V[\eta_0]/(\hat{N}-1)$ and `Coeff(2)`$\approx B^{(2)}$. Particularizing to $\eta_h = \phi_h$ allows to estimate $\hat{V}[\phi] \approx V[\phi]$ and $\hat{\alpha} \approx \alpha$; and to $\eta_h = \bar{\psi}_h$ yields the fitted $\hat{V}[\phi] \approx V[\bar{\psi}]$ (and its bias).

Algorithm 2 Fast fit of nodal constants

for $k = 1, \ldots, m$ **do**
  Solve $\bar{w}_k(\mathbf{x})$ in (38)
end for
Store $\nabla\bar{w}(\mathbf{x}) = \oplus_{k=1}^m \nabla\bar{w}_k(\mathbf{x})$
Estimate Π and $\tilde{\Pi}$
for $i = 1, \ldots, n$ **do**
  set $\hat{M}, \hat{N}, h_1, h_{\hat{M}}$ (may be point-dependent)
  for $j = 1, \ldots, \hat{M}$ **do**
    Run $\hat{N}$ realizations of Algorithm 1 at h_j, drawing $\bar{\psi}_h$ alongside
    Compute $\hat{E}[\phi_{h_j}], \hat{V}[\phi_{h_j}], \hat{E}[\tau_{h_j}], \hat{E}[\bar{\psi}_{h_j}], \hat{V}[\bar{\psi}_{h_j}\phi_{h_j}]$
  end for
  Fit the nodal constants for this node
end for
output: the fitted values $\{\hat{E}_i[\phi], \hat{\beta}_i, \hat{V}_i[\phi], \hat{\alpha}_i, \hat{K}_i, \hat{\rho}_i[\phi, \bar{\psi}]\}_{i=1}^n$

The full fitting procedure is sketched as Algorithm 2. Note that the same sets of trajectories can be used for all the poinwise constants–it is only the functionals that change.

Sensitivity formula, scheduling-, and final algorithms

Combining heuristics **H1** through **H5** we put forward the sensitivity formula (93), based on (70) and (88) with $\bar{\psi}$ replacing $\psi(\boldsymbol{\omega})$:

$$\rho^2[\phi_{h_0}, \xi_{h_0}] \approx \rho^2[\phi, \xi] - \Big|\frac{\alpha\rho[\phi, \xi]}{\beta}\Big|a_0, \text{ with } \rho^2[\phi, \xi] \approx \frac{V[\bar{\psi}]a^2}{4V[\phi]}\Big(1 - \rho^2[\phi, \bar{\psi}]\Big). \quad (93)$$

With it, a scheduling algorithm could be as Algorithm 3. There, the stopping criterion should take into account the quality of the available estimates and thus be problem-dependent. (We discuss this aspect further on Section 6.)

Algorithm 3 Scheduling algorithm based on sensitivity formula (93).

1: **data**: δ, $\hat{\kappa}$, and $\{\hat{K}_i, \hat{\beta}_i, \hat{\alpha}_i, \hat{V}_i[\phi], \hat{V}_i[\bar{\psi}], \hat{\rho}_i[\phi, \bar{\psi}]\}_{i=1}^n$
2: Set $j = 0$, and a_0 is the nodal target error tolerance.
3: **while** stopping criterion not fulfilled **do**
4: Solve the minimization problem

$$a_{j+1} = \arg\min_{a>a_j} \sum_{i=1}^{n} \hat{K}_i \hat{V}_i[\phi]\left(\hat{\kappa}\Big(1 - \hat{r}_i^2(a) + \Big|\frac{\hat{\alpha}_i \hat{r}_i(a)}{\hat{\beta}_i}\Big|a_j\Big) + \Big(\frac{a_j}{a}\Big)^{2+1/\delta}\right), \quad (94)$$

with $\hat{r}_i^2(a) = 1 - \frac{\hat{V}_i[\bar{\psi}]a^2}{4\hat{V}_i[\phi]}\Big(1 - \hat{\rho}_i^2[\phi, \bar{\psi}]\Big)$.

5: Set $j = j + 1$
6: **end while**
7: **output:** $J = j$ and $a_J > a_{J-1} > \ldots > a_1 > a_0$

Finally, we summarize the new version of PDD in Algorithm 4.

6 Numerical experiment

We consider a BVP like (6) with on the two-dimensional domain sketched in Figure 1 with $m = 4$ subdomains, 3 interfaces and $n = 18$ nodes. The PDE is

$$\nabla^2 u + \frac{\cos(x + y)}{1.1 + \sin(x + y)}\Big(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\Big) - \frac{x^2 + y^2}{1.1 + \sin(x + y)}u + f(x, y) = 0, \quad (95)$$

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 BC–see Figure 3 (left). The coefficients of the stochastic representation of (95) are: $\sigma = \sqrt{2}I_2$ (I_2 is the two-dimensional identity matrix), $\mathbf{b} = \cos(x + y)[1, 1]\,/\Big(1.1 + \sin(x + y)\Big)$, and $c = -(x^2 + y^2)/\Big(1.1 + \sin(x + y)\Big) < 0$. The integrator is Algorithm 1, for which we assume that $\delta = 1$; R is an RBF interpolator with multiquadrics [8], and the subdomain solver is FEM. The parameters of R and FEM were so chosen that their errors are negligible compared with the nodal errors.

Algorithm 4 Iterative ('multigrid') probabilistic domain decomposition

Data: a global error tolerance $\epsilon > 0$ for a BVP like (6) in Ω, a confidence interval P_q
Choices: PDD partition $(\{\mathbf{x}_i\}_{i=1}^n, \{\Omega_k\}_{k=1}^m)$, integrator Ξ with known δ, subdomain solver, interfacial interpolator R
Set the nodal target tolerance $a_0(\epsilon)$ according to (64)
Solve $\{\bar{w}_k(\mathbf{x})\}_{k=1}^m$ in (39) in parallel and construct $\bar{\psi}$ from Definition 7 and (74)

Estimate $\hat{\kappa}$ and the nodal constants with Algorithm 2
Find $a_J > ... > a_1$ with the scheduling algorithm (Algorithm 3)
(Optional) Construct $\tilde{u}_J(\mathbf{x})$ based on the 'fitted nodal values' $\hat{E}_1[\phi], \ldots, \hat{E}_n[\phi]$

for $j = J...1$ **do**
  for $i = 1...n$ **do**
    For node $\mathbf{x}_i$ and a_j, determine h and N (29)
    Run N independent realizations of Algorithm 1
    Calculate the nodal value $u_i^{(j-1)}$
  end for
  for $k = 1...m$ **do**

$$\text{Solve the subdomain BVP } \begin{cases} Lv_k + cv_k = f, & \text{if } \mathbf{x} \in \Omega_k, \\ v_k = g, & \text{if } \mathbf{x} \in \partial\Omega_k \cap \partial\Omega, \\ v_k = R[u_i^{(j-1)} | \mathbf{x}_i \in \Gamma_p^k], & \text{if } \mathbf{x} \in \Gamma_p^k,\ 1 \le p \le m_k. \end{cases}$$

  end for
  Construct and store $\tilde{u}_{j-1}(\mathbf{x}) = \oplus_{k=1}^m v_k(\mathbf{x})$ and $\nabla\tilde{u}_{j-1}(\mathbf{x}) = \oplus_{k=1}^m \nabla v_k(\mathbf{x})$
end for

Objectives. With just four small subdomains, it obviously does not take a parallel computer to solve this toy problem–but it serves to test the idea, theory and approximations introduced in this paper in a controlled environment. For the sake of clarity, we stress that we will *not* compare the new IterPDD algorithm with results from deterministic domain decomposition, but with the previous version of PDD, PlainPDD. IterPDD inherits the scalability properties of PlainPDD, and comparisons of PlainPDD with deterministic domain decomposition methods can be found in [1, 2, 3] and references therein. Above all, we will focus on the speedup of IterPDD over PlainPDD, defined as

$$S(a_{j-1}, a_j) = \frac{\text{cost of PlainPDD}(a_{j-1})}{\text{cost of IterPDD}(a_{j-1}, a_j)}. \tag{96}$$

The algorithms have been coded in fully vectorized Matlab, and the nodal values and subdomain BVPs solved on single processors sequentially, for ease of implementation. This suffices to study the computational costs and hence

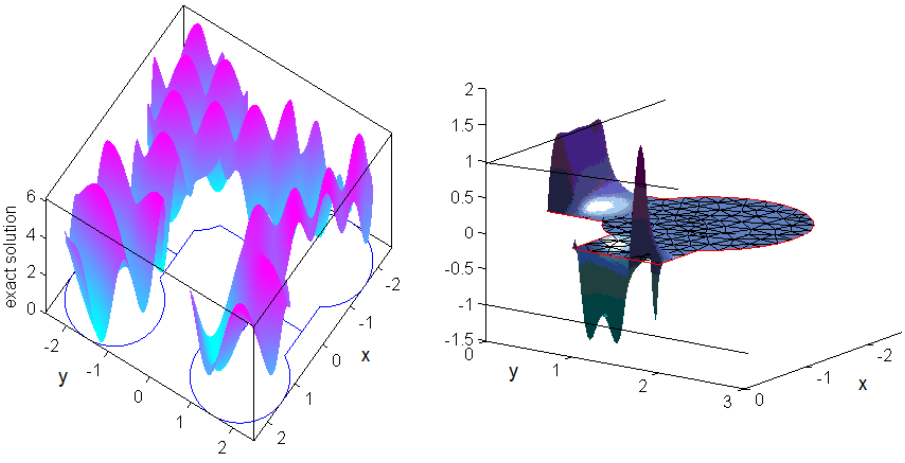

Figure 3: **Left.** Exact solution of (95). **Right.** The bias-related function $\bar{w}_1(x, y)$ (38) used to generate $\bar{\psi}$ (40) on Ω_1 in Figure 1. Notice the various signs of β_i and the overshoots of the RBF interpolator. Here, $\gamma_R \approx 1.5$.

the speedup (96). (The unit cost is a visit to Algorithm 1). Particularly, we are interested in checking whether:

- the heuristics introduced in Section 5 are adequate;
- the theoretically derived sensitivity formula (93), correctly predicts the correlation of scores with pathwise control variates; and
- the scheduling algorithm (Algorithm 3) gives a good approximation to the fastest sequence of IterPDD simulations yielding a final accuracy a_0.

6.1 Preliminary numerical study of the speedup with perfectly balanced Monte Carlo simulations

Actual IterPDD simulations shown later will not be strictly balanced in the sense of Definition 2 because they will rely of heuristic estimates **H5** of the nodal constants according to Section 5. It is therefore informative to first carry out an exploration of the parameter space, using (nearly strictly) balanced Monte Carlo simulations. They will provide best-case reference values against which later, realistic IterPDD simulations in Section 6.2 can be assessed.

As a preparatory step, we carefully compute κ and the nodal constants (see Section 5). These values can be deemed exact and, in particular, the precise values of $\{\beta_i\}_{i=1}^{18}$ ensure that the pointwise MC simulations are balanced. Then, we pick $a_0 = \{.0025, .005, .01, .02, .04\}$; $a_1 = \{.02, .04, .06, .10, .14, \ldots, .62\}$; and run a set of $[\tilde{u}_1, \xi_0(\tilde{u}_1)]$ =IterPDD(a_0, a_1) simulations that will serve as reference. For a target accuracy $a_0 = .01$, Table 2 shows the dependence of some typical

Illustrative values	**Accuracy a_1 of a previous rough PDD solution**					
	exact	lookup	.02	.10	.26	.62
$\sum_{i=1}^n N_0^{(i)} v_0^{(i)}$ (with $\xi_0\big(\tilde{u}_1(a_1)\big)$)	1.03×10^8	1.01×10^8	1.05×10^8	1.18×10^8	1.58×10^8	5.82×10^8
$V[\phi_{h_0} + \xi_{h_0}(a_1)]$	0.16	0.16	0.16	0.17	0.19	0.47
cost of PlainPDD(a_1)			1.60×10^9	1.47×10^7	8.43×10^5	65989
$V[\phi_{h_1}]$			4.92	5.10	5.45	4.08
$\Big(\sum_{i=1}^n \lvert u(\mathbf{x}_i) - \tilde{u}_1(\mathbf{x}_i)\rvert\Big)/n$	use exact u instead of $\tilde{u}(a_1)$		0.012	0.06	0.11	0.77
$\sup_\Omega \lvert u - \tilde{u}_1\rvert$			0.03	0.12	0.24	0.34
$\sup_\Omega \lVert\nabla\tilde{u}_1\rVert_2$			0.87	0.94	2.14	6.61
$\lvert\rho\rvert$	0.9859	0.9859	0.9859	0.9851	0.9833	0.9595
$S(a_0, a_1)$	128.16	72.61	7.37	58.14	46.39	22.67

Table 2: Acceleration by pathwise control variates, and other illustrative quantities, for reference (perfectly balanced) IterPDD($a_0 = .01, a_1$) simulations.

quantities with respect to the accuracy $a_1 > a_0$ used to construct the rough $\tilde{u}(a_1)$ global solution. (Recall that ξ_0 is shorthand for ξ_{h_0}, etc.)

On the column labeled 'exact', the control variate from the exact solution $\xi_0(u_{ex})$ has been used, so that all of the error is quadrature error. This represents the maximum benefit that could possibly be extracted from pathwise control variates. Otherwise, the approximation $\nabla\tilde{u}_1$ is stored on a lookup table (here, a grid), and interpolated from there to compute $\xi_0(\tilde{u}_1)$–except on the column 'lookup', where the lookup table has been filled with the exact ∇u_{ex}, in order to gauge purely the effect of interpolation. Since the cost of solving the subdomains and filling the lookup tables is negligible (i.e. $\Pi \approx 0 \approx \tilde{\Pi}$), the cost of IterPDD($a_0, a_1$) is essentially measured as:

$$\text{cost of IterPDD}(a_0, a_1) = \sum_{i=1}^{n} \Big(N_1^{(i)} v_1^{(i)} + \kappa N_0^{(i)} v_0^{(i)} \Big), \tag{97}$$

where $N_0^{(i)}$ is the number of trajectories from node $\mathbf{x}_i$ at $h = h_0$, and so on.

Let us explain in detail the rightmost column of Table 2. Without control variates, PlainPDD solves the BVP (95) to within a nodal accuracy $a_0 = .01$ at a cost 1.32×10^{10}. If, instead, we get first a rougher PlainPDD solution (at $a_1 = .62$), and use it to construct pathwise control variates, the total cost is 65989 (for the rougher simulation, at a_1) plus 5.82×10^8 (for the finer, at a_0); i.e. nearly 22.67 times less. This is so because the average variance over the $n = 18$ interfacial nodes has dropped, for the finer simulation at h_0, from about 4.90 to 0.47. Just how much the variance drops depends on the correlation $\rho[\phi_{h_0}, \xi_{h_0}(a_1)]$; in this case the average correlation (in absolute value) is $|\rho| = 0.9595$, which is quite good given that a_1 is 62 times larger than a_0. The actual mean interfacial error, 0.77, actually overshoots a_1; this can happen with a small probability, or if h_1 is not small enough. The global quality of the approximation $\tilde{u}_1 = \tilde{u}(a = a_1)$ is gauged by $\sup_\Omega |u - \tilde{u}_1|$. The connection between $\sup_\Omega |u - \tilde{u}_1|$ and $\sup_\Omega \|\nabla\tilde{u}_1\|_2$ with the mean interfacial error were discussed in Section 4 and **H5**, and the values in this concrete example are given here for the sake of illustration.

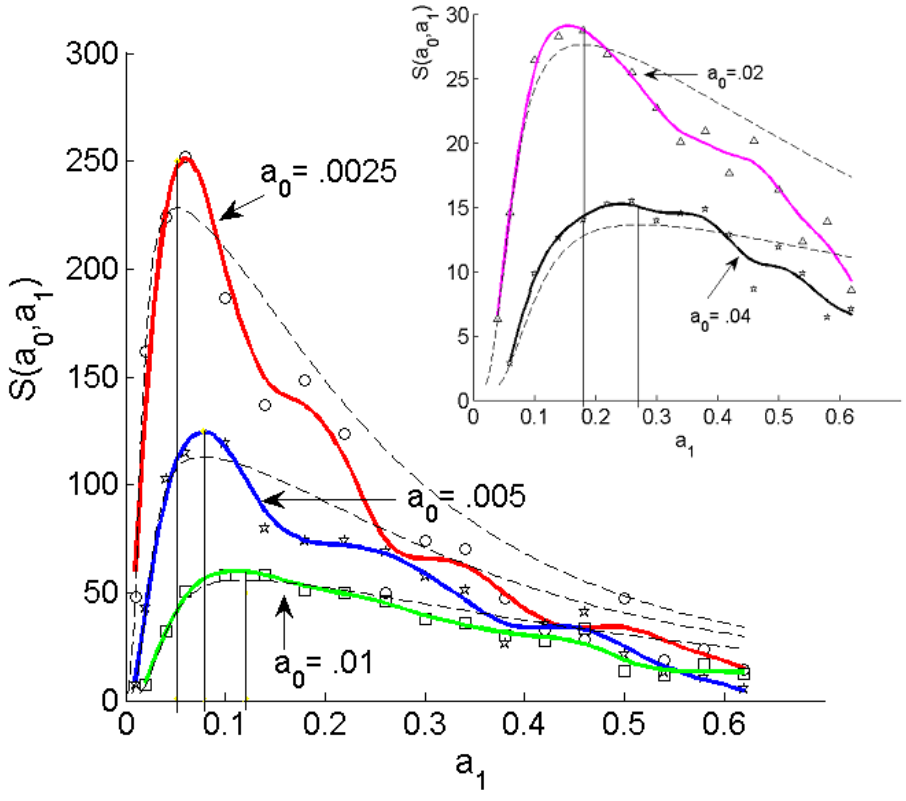

Figure 4: Speedup of IterPDD(a_0, a_1) over PlainPDD(a_0). The symbols denote the reference (strictly balanced) PDD simulations. In order to show the trend, they have been approximated with smoothing splines (solid color curves). **Dashed curves**: predicted speedup, based on the sensitivity formula (93) and fast estimates of constants (Algorithm 2). Although the latter do not lead to balanced MC simulations as with the reference simulations, their maxima calculated with Algorithm 3 (vertical lines) are coincident–see (*) on Table 3.

On Figure 4, the values of $S(a_0, a_1)$ for the full set of reference simulations are depicted with symbols. Recall that those speedups are realizations of some underlying distribution, labeled with $\boldsymbol{\omega}$ (see Section 5). Nonetheless, as a guide for the eye we have joined those symbols corresponding to the same target accuracy a_0 with smoothing splines. As a_1 grows, the effect of randomness on $S(a_0, a_1)$ is more significant.

The symbols and solid lines on Figure 4 summarize our preliminary numerical investigation. In the most favourable test (when $a_0 = .0025$ and $a_1 \approx 0.05$, i.e. the peak of the red curve), a speedup of 250 times was achieved. Of course, this maximum is obtained *a posteriori* and relying of perfect information (the exact nodal constants). The object of this paper is to *predict* the maxima on

Figure 4. In what follows, we will show that i) the optimal a_1 for each target accuracy a_0 can be quite accurately predicted by the sensitivity formula (93); ii) the speedup itself can also be well predicted; and iii) it is possible to improve further the speedup by iterating the method, as dictated by the scheduling algorithm (Algorithm 3). Those predictions rely on the fast estimates of nodal constants by heuristic **H5** (Section 5), as well as on the auxiliar global variable $\bar{\psi}$. Critically, all the needed ingredients can be obtained at a relatively negligible cost and in parallel.

6.2 Heuristics H1-H5 and actual IterPDD simulations

Now, we proceed to test Algorithm 4 proper. First, we perform fast estimates of the nodal constants with Algorithm 2, taking (without trying to optimize in any sense), $\hat{M} = 100$ equispaced timesteps h in $[.001, .01]$ and $\hat{N} = 1000$. (As intended, the cost of this is comparatively very small.) A good estimate $\hat{\kappa} = 1.8$ is straightforward to produce. Moreover, we solve (39) on each subdomain to construct $\bar{\psi}$ by Definition 7 and by (74)–see Figure 3 (right). Based on the resulting fitted constants, we run the scheduling Algorithm 3 in order to get the 'optimal sequences' $a_J > a_{J-1} > \ldots > a_1 > a_0$ for the same set $a_0 = \{.0025, .005, .01, .02, .04\}$ as with the reference simulations before. The minimization (line 4 in Algorithm 3) is carried out with Matlab's `fmincon`, and stopped as soon as the predicted $S(a_j, a_{j+1}) < 1.5$. The calculated optimal sequence for each a_0 are given on the left section of Table 3. Several illustrative quantities, both those predicted by Algorithm 3, and those actually attained after the simulation (using the same set of $\{\hat{\beta}_i\}_{i=1}^{18}$) are listed on the central and right sections of Table 3, respectively.

Let us explain the top vertical data block in Table 3. We wish to get a PDD global solution with nodal accuracy $a_0 = .04$. Algorithm 3 calculates that the fastest way of getting it is by running three PDD iterations altogether, the last two of which use control variates. First, PlainPDD($a_2 = .92$), next IterPDD($a_1 = .27, a_2$), and finally IterPDD($a_0 = .04, a_1$). It predicts that the total cost of this sequence is 13.93 times less than directly computing PlainPDD(a_0)–which directly translates into being 13.93 times faster, given the embarrassingly parallel quality of PDD. This is called 'cumulative speedup', i.e. the cost of IterPDD($a_0, a_1, \ldots, a_J$) over that of PlainPDD(a_0). It also predicts that there is no gain in getting one further previous iteration (at some $a_3 > a_2$), because already $S(a_2, a_1) = 1.95$ is close to the set threshold 1.5. Compare also some of the values predicted by Algorithm 3 with those actually observed *a posteriori*, on the two rightmost columns of Table 3. Let us now move to the inset in Figure 4. In the reference simulations reported in Section 6.1, the experimental curve for $S(a_0 = .04, a_1)$ is represented by the smoothing spline in solid black, whose peak takes place at about $a_1 = 0.25$. Algorithm 3 works with a model of that spline depicted by the dashed black curve hovering near it. The maximum of that approximation takes place at $a_1 = .27$ (vertical line), which is very close, both in terms of the position and magnitude of the maximum. Moreover, Algorithm 3 decides that it can still improve on that by running a previous

PlainPDD(a_2 = .92) simulation, and exploiting the resulting control variates.

This is repeated for $a_0 = \{.0025, .005, .01, .02, .04\}$. First, predictions made by Algorithm 3 are shown on Table 3–and compared with actually observed quantities. Second, the speedup curves predicted by Algorithm 3 are also plotted with dashed lines on Figure 4, and their maxima highlighted with vertical lines. A few comments are in order.

	predicted with Algorithm 3			**observed with Algorithm 4**	
'optimal sequence' $a_J \to \ldots \to a_1 \to a_0$	**cost of IterPDD**(a_{j-1}, a_j)	$\lvert\rho\rvert$	$S(a_{j-1}, a_j)$	**cost of IterPDD**(a_{j-1}, a_j)	$\lvert\rho\rvert$
$a_2 = .92 \to a_1$	3.38×10^5	0.677	1.95	3.64×10^5	0.830
$a_1 = .27(*) \to a_0$	1.54×10^7	0.960	13.63	1.49×10^7	0.970
$\mathbf{a_0 = .04}$	predicted cumulative speedup= 13.93				
$a_2 = .69 \to a_1$	7.84×10^5	0.776	2.97	9.09×10^5	0.864
$a_1 = .18(*) \to a_0$	6.07×10^7	0.981	27.62	5.41×10^7	0.985
$\mathbf{a_0 = .02}$	predicted cumulative speedup= 28.34				
$a_2 = .53 \to a_1$	1.80×10^6	0.862	4.54	2.06×10^6	0.915
$a_1 = .12(*) \to a_0$	2.40×10^8	0.991	55.90	2.19×10^8	0.992
$\mathbf{a_0 = .01}$	predicted cumulative speedup= 57.42				
$a_2 = .41 \to a_1$	4.12×10^6	0.918	6.93	4.07×10^6	0.942
$a_1 = .078(*) \to a_0$	9.49×10^8	0.995	113.01	8.80×10^8	0.996
$\mathbf{a_0 = .005}$	predicted cumulative speedup= 116.00				
$a_3 = 1.03 \to a_2$	2.49×10^5	0.657	1.68	3.18×10^5	0.776
$a_2 = .32 \to a_1$	9.40×10^6	0.948	10.59	8.78×10^6	0.962
$a_1 = .051(*) \to a_0$	3.76×10^9	0.9977	228.22	3.94×10^9	0.9980
$\mathbf{a_0 = .0025}$	predicted cumulative speedup= 233.84				

Table 3: Comparison of results predicted by the 'optimal sequence' obtained with the scheduling algorithm and those actually obtained after running Algorithm 4 with the constants fitted with Algorithm 2. See Figure 4 for the approximate maxima marked with (*), and text for further details.

Even though Algorithm 3 relies on fast estimates of the constants, non-balanced simulations, and neglects the effects of randomness, the predicted speedup curves resemble quite well those obtained from the reference simulations. In particular, the position of the maxima of both sets are nearly coincident. Moreover, for a_0, a_1 small enough, Algorithm 3 provides estimates of the costs, speedup and mean correlation (on Figure 4 and Table 3) which are consistently conservative, as predicted by the theory in Section 5. On the other side, when the tolerances a_{j-1}, a_j are not so small, heuristics **H1-H4** break down, and the fact that $V[\bar{\psi}] < E_\omega\Big[V[\psi(\boldsymbol{\omega})]\Big]$ begins to tell, so the estimates may not be conservative, but still they are acceptable. At least with example (95), most of the cumulative speedup is attained on the last IterPDD simulation.

Finally, let us mention that running the scheduling algorithm with nodal constants based on $\psi(\boldsymbol{\omega})$ instead of $\bar{\psi}$ (as we did) leads to very similar results

to those reported. In fact, in this BVP, it seems that the effect of discretization on $\rho[\phi_h, \xi_h]$ (89) outweighs the effect of randomness in the range of a_0 studied. This is why the speedups in Table 3 grow linearly with a_0. Importantly, this means that one order of magnitude has been knocked down from the Monte Carlo cost estimate $O(1/a_0^{2+1/\delta})$ given in the Introduction, to $O(a_0^{-2})$.

7 Conclusions

In this paper we have laid out the theoretical foundations of a much improved version of PDD, which we have called IterPDD. The theoretical formulas have been derived for linear, smooth, second order elliptic BVPs with Dirichlet BCs. For this case, all the required ingredients of IterPDD are currently available: the probabilistic representation (Dynkin's formula), efficient SDE numerics (the Gobet-Menozzi integrator), and the Green's function representation of the solution. As long as those three items are in place, IterPDD can be extended to other BVPs (at least, linear ones)–although the specific formulas need to be adjusted correspondingly.

With this goal, the PDD programme is currently being further developed in three main directions: i) blending it with Giles' Multilevel method [11]; ii) extending it to parabolic BVPs and mixed BCs; and iii) adjusting for processors with insufficient memory.

On the other hand, hyperbolic problems are typically difficult to handle with stochastic approaches, although representations for some of them do exist [4]. Elliptic and parabolic problems with discontinuous coefficients or boundary singularities can be handled in some cases [5], but often need specific stochastic representations and/or tailored numerical methods, which may not yet be satisfactory. Finally, while mildly nonlinear BVPs could be accommodated into the current IterPDD framework by linearization, of far greater interest are probabilistic representations of nonlinear equations, such as in [2, 3].

The purpose of this mainly theoretical paper is to introduce the strategy of pathwise control variates into the framework of probabilistic domain decomposition, as well as heuristics and approximations which make the idea useful in practice. The numerical results on the example used for illustration are very encouraging, both in terms of confirming the heuristics and of performing hundreds of times faster than the previous version of PDD. It is clear, however, that substantially larger and more challenging problems must be tackled in order to assess the real potential of probabilistic domain decomposition.

8 Acknowledgements

This work was supported by Portuguese national funds through FCT under grants UID/CEC/50021/2013 and PTDC/EIA-CCO/098910/2008. FB also acknowledges FCT funding under grant SFRH/BPD/79986/2011.

References

[1] J.A. Acebrón, M.P. Busico, P. Lanucara and R. Spigler, *Domain decomposition solution of elliptic problems via probabilistic methods*. SIAM J. Sci. Comput. **27**, 440-457 (2005).

[2] J.A. Acebrón, A. Rodríguez-Rozas and R. Spigler, *Domain decomposition solution of nonlinear two-dimensional parabolic problems by random trees*. J. Comput. Phys. **228**, 5574-5591 (2009).

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

[4] J.A. Acebrón and M. Ribeiro, *A Monte Carlo method for solving the one-dimensional telegraph equations with boundary conditions*. J. Comput. Phys. **305**, 29-43 (2016).

[5] F. Bernal, J.A. Acebrón and I. Anjam, *A Stochastic Algorithm Based on Fast Marching for Automatic Capacitance Extraction in Non-Manhattan Geometries*. SIAM Journal on Imaging Sciences **7**(4), 2657-2674 (2014).

[6] F. Bernal and J.A. Acebrón, *A Comparison of Higher-Order Weak Numerical Schemes for Stopped Stochastic Differential Equations*. Submitted (2015).

[7] L. Bers, F. John and M. Schechter, *Partial differential equations*. Interscience (1964)

[8] G.E. Fasshauer, *Meshfree Approximation Methods with Matlab*. Interdisciplinary Mathematical Sciences–Vol. 6 World Scientific Publishers, Singapore (2007).

[9] M. Freidlin, *Functional integration and partial differential equations*. Annals of Mathematics Studies, vol. 109, Princeton University Press, Princeton (1985).

[10] D. Gilbarg and N.S. Trudinger, *Elliptic Partial Differential Equations of Second Order*. Revised 3^{rd} edition, Springer-Verlag Heidelberg Berlin (2001).

[11] M.B. Giles and F. Bernal, *Multilevel simulations of expected exit times and other functionals of stopped diffusions*. In preparation (2015).

[12] P. Glasserman, *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York (2004).

[13] E. Gobet and S. Menozzi, *Stopped diffusion processes: overshoots and boundary correction*. Stochastic Processes and their Applications, **120**, 130-162, (2010).

[14] A.L. Haji-Ali, F. Nobile, E. von Schwerin and R. Tempone, *Optimization of mesh hierarchies in multilevel Monte Carlo samplers*, ArXiv preprint: 1403.2480 (2014)

[15] P.E. Kloeden and E. Platten, *Numerical Solution of Stochastic Differential Equations*. Springer, Applications of Mathematics 23 (1999).

[16] S. Mancini, F. Bernal and J.A. Acebrón, *An efficient algorithm for accelerating Monte Carlo approximations of the solution to boundary value problems*. Accepted in the Journal of Scientific Computing (2015).

[17] G. N. Milstein and M. V. Tretyakov, *Stochastic Numerics for Mathematical Physics*, Springer-Verlag, Berlin, 2004.

[18] B.F. Smith, P.E. Bjorstad and W.D. Grop, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge (1996).