École Polytechnique Fédérale de Lausanne

Switzerland

Stochastic modeling of pollutant transport in aquifers

Stochastic simulations project

Lionel Constantin
lionel.constantin@epfl.ch

Nicola Ischia
nicola.ischia@epfl.ch

January 9, 2019

1 Introduction

In this project, we will analyze models describing the trajectories of particles of pollutants in groundwater. We consider an "infinite" homogeneous aquifer occupying the region $D = R^2 \setminus B(0;\mathrm{R})$. The ball B represents a circular well of radius R from which drinking water is extracted.
We are interested in the probability for a particle, starting at a position X_0 at time 0, of going into the well in less than a final time T. The motion of this particle is considered to be a random motion in a porous media, driven by a velocity field $\mathbf{u}$ given by:

$$\mathbf{u}(x,y) = \mathbf{u}^{\text{steady}}(x,y) + \frac{q}{2\pi}\nabla \log\left(\sqrt{x^2+y^2}\right) \tag{1}$$

Parameters In our project, we used T = 1, R = 1 and $q = 1$. For what concerns $\mathbf{u}^{\text{steady}}(x,y)$, we decided to use (5,0) in order to obtain a quite strong drift in the x-direction. Finally, the diffusion due to the porous medium (σ in Eq. 2 and 4) has been set to 1.

2 Simulations of the aquifer

2.1 Simulation of a SDE

The problem can be tackled solving the following stochastic differential equation, that describes the motion of a particle starting at X_0:

$$\begin{aligned} dX(t) &= \mathbf{u}_1(X(t),Y(t))dt + \sigma dW_1(t),\ 0 \le t \le T \\ dY(t) &= \mathbf{u}_2(X(t),Y(t))dt + \sigma dW_2(t),\ 0 \le t \le T \\ (X(0),Y(0)) &= X_0 \in D \end{aligned} \tag{2}$$

where $W_1(t), W_2(t)$ are two standard Brownian motions.
In order to solve this equation, we used the Euler-Maruyama method. In particular, once we have partitioned the interval $[0,\mathrm{T}]$ in N subintervals of length Δt, the SDE (Eq. 2) can be discretized as follows:

$$\begin{aligned} X_{k+1} &= X_k + u_1(X_k,Y_k)\Delta t + \sigma\sqrt{\Delta t}Z_k,\ Z_k \sim N(0,1) \\ Y_{k+1} &= Y_k + u_2(X_k,Y_k)\Delta t + \sigma\sqrt{\Delta t}Z'_k,\ Z'_k \sim N(0,1) \end{aligned} \tag{3}$$

Then, the simulation is straight-forward: we choose a starting point X_0, and we generate a walk by updating the position of the particle at every step according to Eq. 3. If we arrive to a point inside the well ($|X| < R$) we stop and we consider that the well has been polluted. If we have not reached the well after time T, we say the well has not been polluted. Then, we generate a large number of walks starting from the same point and the probability of polluting the well starting from that point is simply the mean of the outcomes of every walk. The only hyper-parameter that has to be chosen is the time step.
In Figure 1 we investigate the variation of the probability with the time step. We can clearly see that using a too big time step leads to an underestimation of P, and that a maximum size of order $\Delta t = 10^{-3}$ should be chosen.

2.2 Deterministic approach

This problem can also be addressed using a deterministic approach, i.e solving a backward parabolic PDE system that can be easily recast into the forward-in-time system 4. Once we have solved it, the probability for a particle starting from X_0 at $t_0 = 0$ of going into the well in less than T is given by $\varphi(X_0,\mathrm{T})$.

$$\begin{aligned} \varphi_t - ((\mathbf{u}\cdot\nabla)\varphi + \frac{\sigma^2}{2}\Delta\varphi) &= 0 && in\ D \text{ x } [0,\mathrm{T}] \\ \varphi &= 1 && on\ \partial B \text{ x } [0,\mathrm{T}] \\ \varphi(\mathbf{x},t) &\to 0 && as\ |\mathbf{x}| \to \infty \\ \varphi(\mathbf{x},0) &= 0 && in\ D \end{aligned} \tag{4}$$

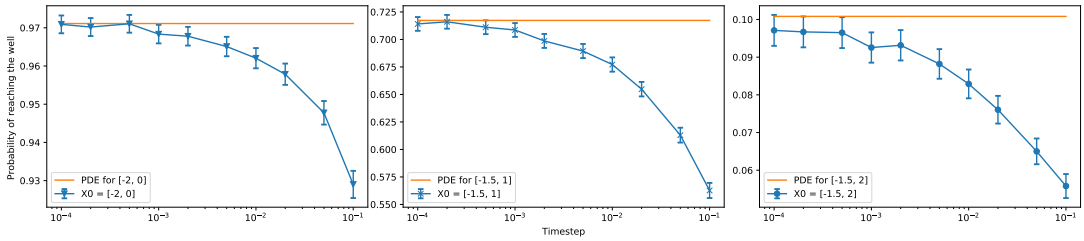

Figure 1: Plot of the probability with the 95% confidence intervals with respect to the time step for 3 different starting positions. For each point in the graphs, a number of 20000 walks was simulated. The probability calculated from the PDE simulation is also shown as reference.

As explained in chapter 10 of [4], we can search for weak solutions solving the following variational formulation:

$$\textit{Find } \varphi \in \mathrm{H}^1(0,\mathrm{T},\mathrm{V},\mathrm{V}^*),\ \textit{with } \mathrm{V} = \mathrm{H}^1_{\partial D \setminus \partial \mathrm{B}}(D),\ \textit{s.t. } \varphi = 1 \textit{ on } \partial B \text{ x } [0,\mathrm{T}] \textit{ and:}$$

$$\begin{aligned} \langle \dot{\varphi}(t), w \rangle_* + B\left(\varphi(t), w, t\right) &= 0 \qquad && \forall w \in \mathrm{H}^1_0(D),\ \textit{a.e in } (0,\mathrm{T}) \\ \varphi(0) &= 0 \end{aligned} \tag{5}$$

$$\textit{Where } B\left(\varphi(t), w, t\right) = \frac{\sigma^2}{2} \int_D \nabla\varphi(t) \cdot \nabla w - \int_D \left(\mathbf{u} \cdot \nabla\right) \varphi w \textit{ is a weak-coercive bilinear form.}$$

We solved this problem relying on the Python library FeniCS [3]. A square computational domain of dimensions (7 x 7) has been used, assuming that the true solution is sufficiently close to 0 at the external borders. For the space discretization, we used finite elements of degree 1 on a triangular mesh, as shown in figure 2, while we used an Euler method of order 1 for the time-derivative approximation.
Finally, in figure 2, the computed solution at time T is shown. As expected, the presence of the velocity field $\mathbf{u}$ plays an important role in the probability, indeed a particle is very likely to end in the well if the starting position is on its left.

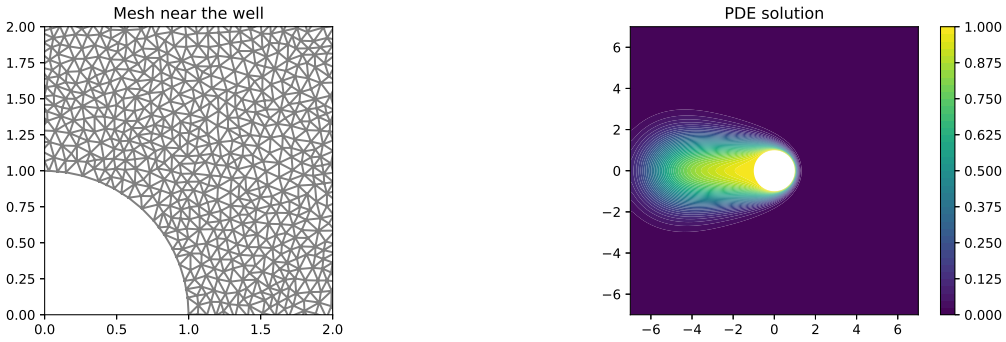

Figure 2: Left: mesh near to the well. Right: computed solution at time T, this represents the probability.

3 Adaptive Time step

Since choosing a too big time step can lead to errors in the computation of the probability and a small one can lead to very long computational time, we are asked to propose an adaptive time step. The time discretization should take into account both accuracy and computational costs, hence it has to be large when particles are far away from the well and become smaller when we are close to the well. We will

first show how we derive such a time step, then discuss its implementation and eventually show how it improves computations.

Derivation is shown in Appendix A. Two different formulas, labelled *Order 1* and *Order 2*, that depend on the position of the particle are proposed.

$$\Delta t_{\mathcal{O}1} = \frac{(R-r)^2}{2\sigma^2} \quad \Delta t_{\mathcal{O}2} = \frac{-\sigma^2 + \sqrt{\sigma^4 + (u_1^2 + u_2^2 + 4\sigma^4)(R-r)^2}}{u_1^2 + u_2^2 + 4\sigma^4}, \tag{6}$$

where r is the radial position of the particle. These two different ways of computing the time step are shown in Figure 3; $\Delta t_{\mathcal{O}2}$ evolves linearly with the distance, while $\Delta t_{\mathcal{O}1}$ evolves like a quadratic function and depends only on σ.

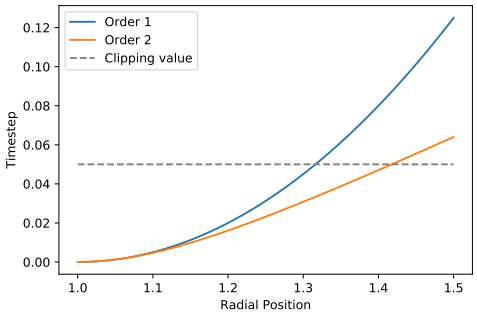

Figure 3: Time step increase with the radial distance from the center of the well on the axis $\theta = \frac{3}{4}\pi$.

Implementation For what concerns the implementation, some more things have to be taken into account. First, the time step can be multiplied by a constant smaller than 1 to have a stronger bound. This could be useful since we are dealing with a random variable, and it is a way to take into account the effect of its variance. Second, when the particle goes very close to well, the time step is very close to zero, leading to a very small jump and the position will change very little in expectation. This could lead to very long, maybe infinite calculations, so we fix a lower bound for the time step. The lower bound must be smaller than the values for which the time step is too large (as seen in Section 2.1), so we decided to set this threshold at 10^{-4}. We also decided to add a max value (*Clipping value* in Fig. 3) to avoid too large step distances if particles are very far away from the well.

Improvements To see how the adaptive time step is performing, we compared it to the Basic MC with a time step of 10^{-4}, the default lower bound of the adaptive MC. The computational time is drastically reduced: *Order 1* takes only around 0.55% of the Basic Walk time ($\sim$ 200 times faster), while *Order 2* takes approximately 0.80% of the time. If we compare *Order 1* with *Order 2*, the first one needs about 75% of the second's walking time.
This huge improvement of the adaptive time steps is due to the fact that they do not need to do a lot of steps when the particles are far away from the well. *Order 1* is faster than *Order 2* because its calculation is less complex; moreover, from Fig. 3 we can see that the time step is also larger, leading to less steps in the walk.

4 Variance Reduction Technique

4.1 Antithetic variables

Firstly, we tried a quite simple approach, trying to find couples of antithetic variables. Recall that in this framework one random variable is the outcome of a random walk $\{(X_k, Y_k)\}_k$ given by Eq. 3. We

tried to achieve negatively correlated couples of random variables by generating, let us say, N couples of random walks with opposite updates $\{\pm(Z_k, Z_k^{'})\}$ at each step k instead of generating $2N$ random walks with totally independent updates.
This choice revealed to be not successful in achieving a variance reduction, especially for starting points that are far from the well.

4.2 Multi Level Monte Carlo (MLMC)

We also tried a more advanced variance reduction technique as defined in [2]. We will not explain the details of the theory here, we will just explain how to apply a MLMC to our problem.

If we have a sequence of estimators $P_0, ..., P_{L-1}$ that approximates P_L (which is the P we want to calculate) with increasing cost and accuracy, then

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}] \tag{7}$$

The MLMC unbiased estimator of $\mathbb{E}[P_L]$ is given by:

$$\hat{P} = \frac{1}{N_0} \sum_{n=1}^{N_0} P_0^{(0,n)} + \sum_{l=1}^{L} \left(\frac{1}{N_l} \sum_{n=1}^{N_l} (P_l^{(l,n)} - P_{l-1}^{(l,n)}) \right) \tag{8}$$

The variance of the estimator is given by:

$$\mathbb{V}[\hat{P}] = \sum_{l=0}^{L} \frac{V_l}{N_l}, \qquad V_l \equiv \mathbb{V}[P_l - P_{l-1}] \tag{9}$$

To perform a MLMC, we need to find different levels of correlated variables P_l. We propose two different ways of doing this. First, we assign each level to a simulation with different starting points but same coordinates update. The second MLMC comes from chapter 5 of [2], where each level have different time steps.

Initial position dependant MLMC The idea is to use for every smaller level a starting position near to X_0 and closer to the well, and then use the same update vector for all the levels. Positions closer to the well will have a smaller computational time, as they will go inside the well more often. By choosing points very close to each other (high correlation), we achieve a variance reduction. However, the computation becomes longer, because instead of simulating the probability at only one point, we now need to simulate it for more than one point and with the same time step accuracy.
We decided to compute the probability for $X_0 = [-1.5, 1]$. The starting points for every level has been taken on the line from [-1.5,1] to [-1,0] (notice that [-1,0] is at the border of the well and its P is so close to 1 that it is very likely to have a variance of 0). A simulation of level L takes L+1 points linearly distributed on this line, borders included. We also had to choose the number of walks on every level. We arbitrarily attributed 1000 walks to level L and tried different values for the others by proposing a linear distribution between level 0 and level L. We investigated how changing the number of walks on the level 0 performs in Figure 4, for different numbers of levels. It turns out that for all numbers of levels, there is a "step value" (around 1100 on Fig. 4) for the initial number of walks that lead to a variance reduction. A larger number of levels or walks increases the variance reduction, but also the computational cost.

Time step dependant MLMC With this method, the correlated variables are computed updating the position of the particle using different time steps, but the same direction (at every step, using the same normal variables from Eq. 3). We propose the following formula for the time step at each level: $\Delta t_l = \Delta t_0 m^l$, with $0 < m < 1$. To compute m, for a chosen number of levels, the time step of level 0 and

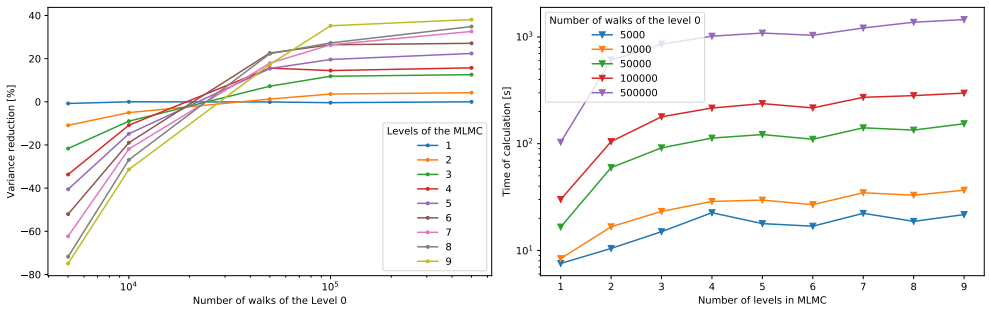

Figure 4: Left: variance reduction achieved for different levels and different number of walks for the position 0, the variance reduction was calculated as being: $\frac{\mathbb{V}(P_L)-\mathbb{V}(\hat{P})}{\mathbb{V}(P_L)}$. Right: time taken by every simulation (ran on our laptop)

of level L has been fixed. From [2], the optimal number of walks should be proportional to 2^{-l} on the finer levels. We performed simulations for small number of levels (from 1 to 9) and different Δt_0s, but with Δt_L fixed to 10^{-4}. The number of walks on each level is given by 20000×2^{-l}.
Results are shown in Figure 5: in general, the variance is more reduced for high number of levels (because it implies the variance of P_L to be high as the number of walk of L is small). Except for one case, we get a real reduction only when at level 0 we set $\Delta t \sim 10^{-4}$, since the variables are highly correlated. However, with such a small Δt in the initial level all the levels have small time steps and the computational cost is not reduced.

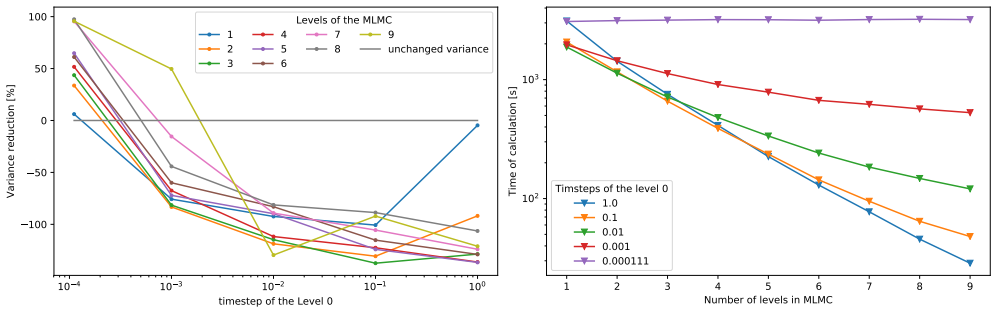

Figure 5: Left: variance reduction achieved for different levels and starting time step. Right: time taken by every simulation(ran on our laptop), with 20000×2^{-l} walks on level l.

Conclusion of MLMC We face a problem with the MLMC: to reduce the variance, the variables belonging to "neighbouring" levels need to be correlated. In order to get correlated variables, it is required to reduce the difference between the levels, but doing this increases the cost of calculation and it would be more efficient to just increase the number of steps in a simple Monte Carlo. Also having a lot of levels implies that smaller levels have more walks than the finer ones, which can also increase the computational cost.
If these methods are not useful for estimating the probability for a single point with one determined time step, we can imagine situations where they could be useful: the position dependant MLMC could be used to find probability for a large number of points, by taking the probabilities calculated for every level (i.e. we do a MLMC of level 100 with 101 different starting points, and we get the probability for all these points with just one MLMC). Similarly we could use the time step dependant MLMC to produce results similar to Fig. 1.

5 Splitting method

The methods discussed so far are not suitable for the estimation of *rare events* probabilities. If one is interested in such small probabilities, relying on one of the methods above would result in a very costly computation. As we will see, the splitting method allows to reduce drastically the computational time required to compute small probability events.
We are going to explain the splitting method in our specific case, but it can be applied in much more general situations, as explained in [1].

5.1 Derivation

The key of this method is to substitute the computation of the quantity of interest with some other quantities, that require much less effort and that combined together can lead to a good approximation of the original quantity.

In our specific situation, we divide the domain around the well in annular sections centered around it, we will refer at these sections as *stages* and they will be indicated with C_i (see Figure 6).
In the naive Monte Carlo approximation, we would generate N path starting from an initial position X_0, and for each of them we would check if the particle went inside the well or not. In the framework of the splitting method, instead, we do not fix a priori the number of simulations N, but we choose into how many *"subwalks"* N_i one walk should *split* once the stage C_i has been crossed.
Our goal is to estimate the probabilities p_i that are, given the initial state X_0 and the division in stages C_i, the probabilities that a sample path starting at the lower threshold of stage C_i will hit the upper threshold before stopping, i.e before a time T has elapsed. As an example, if we consider the trivial case of Figure 6, we will estimate p_2 and p_1 as $1/2$ and $1/3$ respectively. If we are able to compute these probabilities, then the probability P for a particle of going into the well starting from the position X_0 before time T has elapsed, is simply given by the following formula:

$$P = \prod_{i=1}^{N_{\text{stages}}} p_i \tag{10}$$

This formula is proven in details in [1], here we limit ourselves to state the independency assumption behind it. In particular, we assume that in the first stage the paths are independent. Moreover, in the following stages, walks which come from the same path at the previous stage must be considered to be dependent, since they share a common history, but they progress independently after the split.

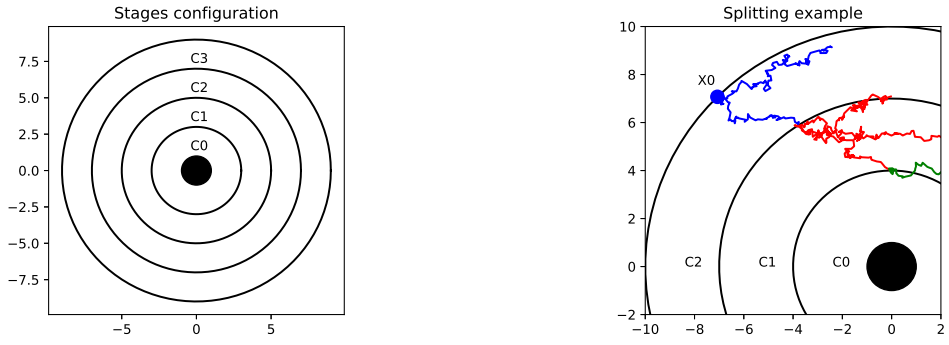

Figure 6: Left: example of 4 stage configuration. Right: example of splitting method with number of new walks per stage of 2, 3 and 1 in stages C_2, C_1, C_0 respectively.

5.2 Implementation and error estimates

The main problems to tackle when implementing the splitting method are two:

- Choice of the stages C_i
- Choice of N_i

We are going to see how these problems are treated, and then we are going to give an estimate for the variance of this method.

Choice of C_i We decided to split uniformly the space around the well, using equispaced circles. This choice is purely arbitrary, along with the shape of the stages.
Nevertheless, there is a practical reasoning behind this choice. First of all, from the point of view of the implementation this is the most natural and simple choice. Secondly and most importantly, we can easily have high values for the probabilities p_i by choosing a high number of such stages. This is very useful, since the aim is basically to avoid the computation of *rare events*.

Choice of N_i As explained in [1], care has to be taken to choose the number of splits N_i appropriately; a too small number will generate few hits of the next threshold and this will cause the paths to "die-out" after a certain number of stages. Conversely, a too large splitting factor N_i will cause too many hits of the next threshold and will cause the number of paths per stage to "snowball". This will result in a large total simulation time, and the highest fraction of it spent on the last stages, which is not efficient.
We tried two different strategies to deal with this choice:

1. To avoid the "snowball" effect, we heuristically tried to choose high values of splits in the first stages and smaller values in the last stages. This should both help to avoid the simulation to "die-out" in the first stages, where we have generally less walks, and balance the total number of walks per stage, that could grow exponentially in the latter stages.

2. A more appropriate choice of N_i could be done with the help of a pilot run, where we can roughly estimate p_i and then use the following formula to compute the splits:

$$N_i = \left\lceil \frac{1}{p_i} \right\rceil \qquad \text{"balanced growth"}$$

   This choice should help the total number of walks to increase in a balanced way. However, we saw that this choice could lead to too small N_i when the probabilities p_i are too high, hence to a bad approximation of the probability. To overcome this issue, we can multiply by a small factor (2 or 3 usually) the obtained values of the splits.

Variance and error estimate To introduce the formula for the variance, we first have to introduce some more notation.
Firstly, we will call "roots" the N_0 walks generated in the first stage C_0; this name comes from the fact that for each path in the first stage, a tree of walks that share the same initial "root" will be generated. This tree can be very small, for instance if the starting path does not reach the second stage, but obviously can also be very large in the case of several hits of the next stages by the offspring of the first stage path.
The second thing that we need to define is Y_j for $j = 0, ..., N_0$; that is the number of walks that reached the well and that have been generated by the $j-th$ root.
If we denote by $\hat{P}$ the estimator for the probability given by plugging the estimations of p_i into 10, its variance is given by:

$$\mathrm{Var}[\hat{P}] = \frac{\mathrm{Var}\left[\sum_{j=1}^{N_0} Y_j\right]}{\prod_{i=0}^{N_{\text{stages}}} N_i^2} \tag{11}$$

Since in 5.1 we assumed that the "roots" are independent, the random variables Y_j will have the same variance. Thus the formula 11 can be simplified, becoming:

$$\mathrm{Var}[\hat{P}] = \frac{\mathrm{Var}[Y_1]}{N_0 \prod_{i=1}^{N_{\text{stages}}} N_i^2} \tag{12}$$

Finally, we can estimate the variance of Y_1 using the standard variance estimator $S_{Y_1}^2 = \sum(Y_j - \bar{Y})^2/(N_0 - 1)$ and plugging it into 12, to obtain the final variance estimator.

5.3 Results

The results of this method are quite impressive, since we were able to obtain quite satisfactory results in cases of very small probabilities. In Table 1 prediction of the splitting method for starting points $(3, 3)$ and $(3.5, 3.5)$ are reported.

X_0	**PDE prob**	**Nr. stages**	**Estimated prob**	**95% CI**
$(3, 3)$	1.199e-13	10	8.896e-14	+- 3.515e-14
$(3.5, 3.5)$	1.605e-16	12	2.620e-16	+- 1.624e-16

Table 1: Results of splitting method with $N_i = [20, 26, 20, 28, 32, 34, 42, 44, 44, 48]$ (2 * balanced growth) and $[54, 34, 31, 29, 25, 25, 25, 25, 25, 25, 25, 25]$ respectively, using $\Delta t = 10^{-3}$.

This method has been used in very low probabilities scenarios, where it is almost impossible to perform a comparison with the standard MC technique. As you can see from Table 1, the probabilities that have been estimated are very low and we were not able to reproduce satisfactory results with such small probabilities with a naive approach, because of the high computational costs.

References

[1] Marnix Joseph Johann Garvels. The splitting method in rare event simulation. *Doctoral thesis, University of Twente*, 2000.

[2] Michael B. Giles. Multilevel monte carlo methods. *Acta Numerica - Cambridge University Press*, 2018.

[3] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python.* Springer, 2017.

[4] Sandro Salsa. *Partial Differential Equations in Action. From Modelling to Theory.* Springer, 3rd edition, 2016.

Appendices

A Derivation of the adaptive time step

To propose an adaptive time step formula, we will first find an upper bound for d_R, that is the distance in the radial direction covered by the particle in Δt. From this distance, we will derive an adapted value for Δt that will depend on r, i.e the radial position of the particle wrt the center of the well (0,0), in order to cover small radial distances d_R when the pollutant approaches the well.

To begin with, let us calculate the distance covered by a particle in one time step d:

$$\begin{aligned} d^2 &= (\Delta X)^2 + (\Delta Y)^2 = (X_{k+1}^2 - X_k^2) + (Y_{k+1}^2 - Y_k^2) \\ &= (u_1\Delta t + \sigma\sqrt{\Delta t}Z_k)^2 + (u_2\Delta t + \sigma\sqrt{\Delta t}Z'_k)^2 \\ &= (u_1^2+u_2^2)\Delta t^2 + \sigma^2\Delta t(Z_k^2+Z_k'^2) + 2\sigma\sqrt{\Delta t^3}(Z_k u_1 + Z'_k u_2) \end{aligned} \tag{13}$$

We are interested in d_R, the projection of d on the radius. We will not calculate it but we will simply consider that d is an upper bound for d_R. Thanks to this, we will be able to choose a Δt that depends on r such that:

$$d_R^2 \sim (R-r)^2 \tag{14}$$

However, this is not trivial since we have bounded d_R with d and the latter one is a random variable. We are going to consider the expected value and the variance of d^2.

$$\mathbb{E}[d^2] = \mathbb{E}[(u_1^2+u_2^2)\Delta t^2 + \sigma^2\Delta t(Z_k^2+Z_k'^2) + 2\sigma\sqrt{\Delta t^3}(Z_k u_1 + Z'_k u_2)] \tag{15}$$

i) $\mathbb{E}[(u_1^2+u_2^2)\Delta t^2]$ is simply equal to $(u_1^2+u_2^2)\Delta t^2$.
ii) $\mathbb{E}[\sigma^2\Delta t(Z_k^2+Z_k'^2)] = 2\sigma^2\Delta t$ as $Z_k^2+Z_k'^2$ is a χ_2^2 distribution, and $\mathbb{E}[\chi_k^2] = k$.
iii) $\mathbb{E}[2\sigma\sqrt{\Delta t^3}(Z_k u_1 + Z'_k u_2)] = 0$ as Z_k, Z'_k are Normal distributions with mean = 0.

So we have:

$$\mathbb{E}[d^2] = (u_1^2+u_2^2)\Delta t^2 + 2\sigma^2\Delta t \tag{16}$$

The calculation of the variance of d^2 is more complex because we have a correlation between some terms:

$$\begin{aligned} \mathrm{Var}[d^2] =& \mathrm{Var}[(u_1^2+u_2^2)\Delta t^2 + \sigma^2\Delta t(Z_k^2+Z_k'^2) + 2\sigma\sqrt{\Delta t^3}(Z_k u_1 + Z'_k u_2)] \\ =& \mathrm{Var}[\sigma^2\Delta t(Z_k^2+Z_k'^2) + 2\sigma\sqrt{\Delta t^3}(Z_k u_1 + Z'_k u_2)] \\ =& \sigma^4\Delta t^2\mathrm{Var}[Z_k^2+Z_k'^2] + 4\sigma^2\Delta t^3(u_1^2\mathrm{Var}[Z_k] + u_2^2\mathrm{Var}[Z'_k]) \\ &+ 4\sigma^3\Delta t^{5/2}\mathrm{Cov}(Z_k^2+Z_k'^2, Z_k u_1 + Z'_k u_2) \end{aligned} \tag{17}$$

by properties of the Variance and independence of Z_k and Z'_k; moreover:
i) $\mathrm{Var}[Z_k^2+Z_k'^2] = 4$ is the variance of a χ_2^2 distribution, and $\mathrm{Var}[\chi_k^2] = 2k$.
ii) $\mathrm{Var}[Z_k] = \mathrm{Var}[Z'_k] = 1$ as Z_k and Z'_k are $\mathcal{N}(0,1)$.
iii) $\mathrm{Cov}(Z_k^2+Z_k'^2, Z_k u_1 + Z'_k u_2)$ is harder to solve and we will not discuss it here. However, if we assume $\Delta t << 1$ we can approximate:

$$\mathrm{Var}[d^2] = 4\sigma^4\Delta t^2 + \mathcal{O}(\Delta t^{5/2}) \tag{18}$$

Now we are ready to propose an equation for the relation 14. Since d is an upper bound for d_R, we can choose Δt such that in expectation d behaves well, bounding the sum of its expected value and its variance

in the following way:

$$\begin{aligned} \mathbb{E}[d^2] + \mathrm{Var}[d^2] &< (R-r)^2 \\ (u_1^2 + u_2^2)\Delta t^2 + 2\sigma^2 \Delta t + 4\sigma^4 \Delta t^2 &< (R-r)^2 \\ (u_1^2 + u_2^2 + 4\sigma^4)\Delta t^2 + 2\sigma^2 \Delta t - (R-r)^2 &< 0 \end{aligned} \tag{19}$$

This is a second order equation on Δt^2 which has roots:

$$\Delta t_{1,2} = \frac{-2\sigma^2 \pm \sqrt{(2\sigma^2)^2 + 4(u_1^2 + u_2^2 + 4\sigma^4)(R-r)^2}}{2(u_1^2 + u_2^2 + 4\sigma^4)} \tag{20}$$

Where we keep only the positive solution and simplify to get:

$$\Delta t_{\mathcal{O}2} = \frac{-\sigma^2 + \sqrt{\sigma^4 + (u_1^2 + u_2^2 + 4\sigma^4)(R-r)^2}}{u_1^2 + u_2^2 + 4\sigma^4} \tag{21}$$

Equation 21 gives an upper bound for the time step. This time step is always positive, it is 0 if $R = r$ and increases with R. Also, u_1 and u_2 depend on r.

However, it can be costly to compute 21 at every step of the random walk, hence we also propose a simpler term approximating the variance and the expected value of d omitting the terms of order $\mathcal{O}(\Delta t^2)$:

$$\mathrm{Var}[d^2] = \mathcal{O}(\Delta t^2) \quad \text{and} \quad \mathbb{E}[d^2] = 2\sigma^2 \Delta t + \mathcal{O}(\Delta t^2) \tag{22}$$

With the above approximation, from equation 19 we get:

$$\begin{aligned} 2\sigma^2 \Delta t &< (R-r)^2 \\ \Delta t_{\mathcal{O}1} &= \frac{(R-r)^2}{2\sigma^2} \end{aligned} \tag{23}$$