European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

THE PARALLEL IMPLEMENTATION OF FORWARD-REVERSE ESTIMATOR

Evelyne Huber[*], **Daria Spivakovskaya**[†], **Hai X. Lin**[†] **and Arnold W. Heemink**[†]

[*]ETH Zrich-Swiss Federal Institute of Technology,
e-mail:hubere@student.ethz.ch
[†]Delft University of Technology,
Delft Institute of Applied Mathematics
Mekelweg 4, 2628 CD, The Netherlands
e-mail: D.Spivakovskaya@ewi.tudelft.nl, h.x.lin@ewi.tudelft.nl, A.W.Heemink@ewi.tudelft.nl

Key words: Forward-reverse estimator, Random walk model, Advection-diffusion equation, Parallel algorithm

Abstract. *For a lot of applications the Lagrangian approach may be a suitable alternative of the Eulerian approach. However, one of the biggest disadvantages of the Monte Carlo methods is their computational expensiveness. The traditional Monte Carlo methods can be improved by using more sophisticated methods, as,for instance, the forward-reverse estimator recently introduced by Milstein, Schoenmakers and Spokoiny [7]. Another method to speed up the Monte Carlo methods is to use parallel processing. In this paper the parallel version of the forward-reverse estimator is developed and analyzed. The parallelized estimator is applied to estimate the concentration of pollutant in Dutch coastal zone and the efficiency of the algorithm is discussed*

1 INTRODUCTION

Last several years the ship accidents caused the serious ecological catastrophes in the seaside of different countries. To prevent or, at least, reduce the possible damage due to accidents we need to develop efficient and accurate model for the simulation of the pollution spreading. The movement of the pollutant can be modeled by using a random walk model [4]. Here the trajectory of a particle of the pollutant is simulated with the help of the appropriate system of the stochastic differential equations. By averaging the positions of many particles the concentration of the pollutant can be found.

For a number of applications, it is not necessary to simulate the concentration in the whole domain of the problem. For these kind of problem the forward- reverse estimator can be applied. This estimator has recently been introduced by Milstein, Schoenmakers and Spokoiny [7] and is based on realizations of original forward system and also on realizations of reverse time system derived from original forward one. This approach allows to compute the concentration of the pollutant in certain region efficiently without solving the complete simulation problem.

Because of the independence of movement of particles random walk models can be easily parallelized. In this paper we considered the parallelization of the forward-reverse estimator based on the particle decomposition. Several approaches are proposed and their advantages, disadvantages and efficiency are discussed.

The parallel version of the forward-reverse method is applied for the simulation of the pollution spreading along the Dutch seaside. For this model we investigate the efficiency of the parallel algorithm. The results show that the efficiency of the parallelization is very high, especially for a large size of realizations.

2 THE TRANSITION DENSITY ESTIMATORS

2.1 The forward estimator (FE)

First of all, let us consider the stochastic differential equation in the Îto sense in general form:

$$\begin{array}{l} d\boldsymbol{X}(s) = \boldsymbol{a}(s,\boldsymbol{X})ds + \boldsymbol{\sigma}(s,\boldsymbol{X})d\boldsymbol{W}(s), \quad t \le s \\ \boldsymbol{X}(t) = \boldsymbol{x} \end{array} \tag{1}$$

where $\boldsymbol{X} = (X_1, \dots, X_d)^T$. $\boldsymbol{a} = (a_1, \dots, a_d)^T$ are d-dimensional vectors, $\boldsymbol{W} = (W_1, \dots, W_m)^T$ $(m \ge d)$ is a m-dimensional Wiener process and $\boldsymbol{\sigma} = \{\sigma_{ij}\}$ is a $d \times m$ matrix. We assume that the $d \times d$ matrix $\boldsymbol{b} = \boldsymbol{\sigma}\boldsymbol{\sigma}^T$ is of full rank for every $(s, \boldsymbol{z})$, $s \in [t, T]$, $\boldsymbol{z} \in \mathbb{R}^d$. We also assume that the functions $a_i(s, \boldsymbol{z})$ and $\sigma_{ij}(s, \boldsymbol{z})$ and their first derivatives are continuous and bounded. This particularly implies existence and uniqness of the solution $\boldsymbol{X}_{t,\boldsymbol{x}}(s) \in \mathbb{R}^d$, $\boldsymbol{X}(t) = \boldsymbol{x}$ of (1), smoothness of the transition density $p(t, \boldsymbol{x}, s, \boldsymbol{z})$ of the Markov process $\boldsymbol{X}$ and existence of all the moments $p(\cdot, \cdot, \cdot, \boldsymbol{y})$ [1, 3, 9].

In many situations we do need to know the exact solution of the equation (1), which is called strong solution, but the solution in weaker sense - the probability law of the process $\boldsymbol{X}$, so called weak solution. The weak solution is wider concept, than the strong solution in the sense that the strong solution is of always a weak solution, but the converse is not true. There are stochastic differential equations that have weak solutions, but have no strong ones (for instance, Tanaka equation). Further, speaking about the solution of stochastic differential equation, we will always mean the weak solution.

Suppose that our goal is to find the transition density function $p(t, \boldsymbol{x}; s, \boldsymbol{z})$ of the stochastic process $\boldsymbol{X}$. It may be found from the following partial differential equation, called Fokker-Planck equation (or Kolmogorov equation) $(t \le s)$

$$\begin{array}{l} \dfrac{\partial p}{\partial s} = \dfrac{1}{2} \sum\limits_{i,j=1}^{d} \dfrac{\partial^2}{\partial z_i \partial z_j}(b_{ij}p) - \sum\limits_{i=1}^{d} \dfrac{\partial}{\partial z_i}(a_i p), \\ p(t, \boldsymbol{x}, t, \boldsymbol{z}) = \delta(\boldsymbol{x} - \boldsymbol{z}) \end{array} \tag{2}$$

Another way is to use the standard method of non-parametric statistics called kernel estimator [10, 16].

$$\widehat{p}(t,\boldsymbol{x},T,\boldsymbol{y}) = \frac{1}{N\delta^d}\sum_{n=1}^{N} K\left(\frac{\boldsymbol{X}_{t,\boldsymbol{x}}^{(n)}(T)-\boldsymbol{y}}{\delta}\right), \tag{3}$$

where δ is a positive real number called bandwidth, K is a kernel function and $\left\{\boldsymbol{X}_{t,\boldsymbol{x}}^{(n)}\right\}$, $n = 1,\ldots,N$ are independent realizations of the solution $\boldsymbol{X}$ of stochastic differential equation (1) with initial condition $\boldsymbol{X}(t) = \boldsymbol{x}$. In general, the sample $\left\{\boldsymbol{X}_{t,\boldsymbol{x}}^{(n)}\Big(T)\right\}$, $n = 1,\ldots,N$ is not known and to use the estimator (3) we need to use one of the approximation scheme, for instance the Euler scheme [5, 8]

$$\begin{aligned}&\overline{\boldsymbol{X}}_{k+1}^{\Delta t} = \overline{\boldsymbol{X}}_{k}^{\Delta t} + \boldsymbol{a}(t,\overline{\boldsymbol{X}}_{k}^{\Delta t})\Delta t + \boldsymbol{\sigma}(t,\overline{\boldsymbol{X}}_{k}^{\Delta t})\Delta\boldsymbol{W}_k,\\ &\overline{\boldsymbol{X}}_{0}^{\Delta t} = \boldsymbol{0},\end{aligned} \tag{4}$$

where $k = 0,\ldots,L-1$, $\overline{\boldsymbol{X}}_{k}^{\Delta t} := \overline{\boldsymbol{X}}^{\Delta t}(t_k)$ is the numerical approximation of the position $\boldsymbol{X}(t_k)$, $t_k = k\Delta t$, $\Delta t = (T-t)/L$ is the time step of numerical integration and $\Delta\boldsymbol{W}_k$ are mutually independent Gaussian variable with zero mean and covariance matrix $\Delta t\boldsymbol{I}_m$.

Applying the approximation (4), the estimator (3) may be transformed as

$$\widehat{p}(t,\boldsymbol{x},T,\boldsymbol{y}) = \frac{1}{N\delta^d}\sum_{n=1}^{N} K\left(\frac{\overline{\boldsymbol{X}}_{t,\boldsymbol{x}}^{(n)}(T)-\boldsymbol{y}}{\delta}\right). \tag{5}$$

Because this estimator is based on the forward system we will call it as Forward Estimator (FE). The error of this estimator can be split into two parts: the error due to the kernel estimator (3) and the error due to the approximation of the stochastic process $\boldsymbol{X}$. The first error depends on the choice of the kernel function and the bandwidth. The most popular kernel functions are Gaussian function

$$K(\boldsymbol{x}) = (2\pi)^{-d/2} exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}\right) \tag{6}$$

and Epanechnikov symmetric multivariate kernel

$$K(\boldsymbol{x}) = \frac{1}{2}\nu_d^{-1}(d+2)(1-\boldsymbol{x}^T\boldsymbol{x})1_{\boldsymbol{x}^T\boldsymbol{x}\leq 1} \tag{7}$$

where $\nu_d = 2\pi^{d/2}/\{d\Gamma(d/2)\}$ is the volume of the unit d-dimensional sphere. However, the c the accuracy of the kernel estimator depends most of all on the choice of bandwidth δ. It is well known ([10, 16]) that the optimal bandwidth is given by $\delta_{opt} \sim N^{-\frac{1}{d+4}}$ provides the error due to kernel estimator of the order $\epsilon(p) \sim N^{-\frac{2}{d+4}}$. As a result, for high dimensional systems this method became inefficient. In statistical literature this fact is often referred as "the curse of dimensionality".

2.2 The reverse estimator (RE)

In the previous section we described the estimator based on forward system. Another approach is based on the so called reverse time diffusion and has been introduced by [15]. If the forward estimator (5) is connected with the Fokker-Planck equation (2), the reverse estimator is connected with the Kolmogorov backward equation

$$\begin{aligned}&\frac{\partial p}{\partial t} = -\frac{1}{2}\sum_{i,j=1}^{d} b_{ij}\frac{\partial^2 p}{\partial x_i \partial x_j} - \sum_{i=1}^{d} a_i \frac{\partial p}{\partial x_i},\\ &p(T, \boldsymbol{x}, T, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y})\end{aligned} \tag{8}$$

The main idea of the reverse time approach is that the forward Kolmogorov equation can be considered as a backward Kolmogorov equation (8) associated with the stochastic process $(\boldsymbol{Y}, \mathcal{Y})$ in $\mathbb{R}^{d+1}$, hence in a space of one dimension higher.

First, we introduce a reverse time variable $\tilde{s} = T + t - s$, $t \leq \tilde{s} \leq T$ and define the functions

$$\begin{aligned}\tilde{a}_i(\tilde{s}, \tilde{\boldsymbol{y}}) &= a_i(T + t - \tilde{s}, \tilde{\boldsymbol{y}}),\\ \tilde{b}_{ij}(\tilde{s}, \tilde{\boldsymbol{y}}) &= b_{ij}(T + t - \tilde{s}, \tilde{\boldsymbol{y}}).\end{aligned} \tag{9}$$

For a stochastic process $\boldsymbol{Y}_{t,\boldsymbol{y}}(\tilde{s}) \in \mathbb{R}^d$ and a scalar process $\mathcal{Y}_{t,\boldsymbol{y}}(\tilde{s})$ we then consider the system of stochastic differential equations

$$\begin{aligned}&d\boldsymbol{Y} = \boldsymbol{\alpha}(\tilde{s}, \boldsymbol{Y})d\tilde{s} + \tilde{\boldsymbol{\sigma}}(\tilde{s}, \boldsymbol{Y})d\tilde{\boldsymbol{B}}(\tilde{s}),\\ &d\mathcal{Y} = c(\tilde{s}, \boldsymbol{Y})\mathcal{Y}d\boldsymbol{s},\\ &\boldsymbol{Y}(t) = \boldsymbol{y}, \quad \mathcal{Y}(t) = 1\end{aligned} \tag{10}$$

with $\tilde{\boldsymbol{B}}$ being an m-dimensional standard Wiener process and

$$\begin{aligned}\alpha_i(\tilde{s}, \tilde{\boldsymbol{y}}) &= \sum_{j=1}^{d} \frac{\partial \tilde{b}_{ij}}{\partial \tilde{y}_j} - \tilde{a}_i,\\ c(\tilde{s}, \tilde{\boldsymbol{y}}) &= \frac{1}{2}\sum_{i,1=1}^{d} \frac{\partial^2 \tilde{b}_{ij}}{\partial \tilde{y}_i \partial \tilde{y}_j} - \sum_{i=1}^{d} \frac{\partial \tilde{a}_i}{\partial \tilde{y}_i}\end{aligned} \tag{11}$$

It can be proved that the transition density function $p(t, \boldsymbol{x}, T, \boldsymbol{y})$ of the forward process $\boldsymbol{X}$ can be represented with the help of the reverse-time system (9) [6, 7]. By applying the kernel estimator, it is possible to construct the density estimator based only on the reverse time system

$$\widehat{p}(t, \boldsymbol{x}, T, \boldsymbol{y}) = \frac{1}{M\delta^d}\sum_{m=1}^{M} K\left(\frac{\boldsymbol{x} - \overline{\boldsymbol{Y}}_{t,\boldsymbol{y}}^{(m)}}{\delta}\right)\overline{\mathcal{Y}}_{t,\boldsymbol{y}}^{(m)}(T), \tag{12}$$

where $(\overline{\boldsymbol{Y}}_{t,\boldsymbol{y}}^{(m)}, \overline{\mathcal{Y}}_{t,\boldsymbol{y}}^{(m)})$, $m = 1, \ldots, M$ is an independent identically distributed (i.i.d.) sample of numerical solutions of system (10). The Reverse Estimator (RE) (12) can be applied in a number of application, for instance for the risk analysis [13].

2.3 The forward-reverse density estimator (FRE)

In this section we discuss a new estimator based on both the forward system (1) and the reverse time system (10). This estimator is called the forward-reverse estimator and has been recently introduced by [7]. By taking advantage of the forward and reverse-time systems via the Chapman-Kolmogorov equation with respect to an intermediate time t^* one can obtain the following estimator

$$\widehat{p}(t,\boldsymbol{x},T,\boldsymbol{y}) = \frac{1}{MN\delta^d}\sum_{n=1}^{N}\sum_{m=1}^{M} K\left(\frac{\overline{\boldsymbol{X}}^{(n)}_{t,\boldsymbol{x}}(t^*) - \overline{\boldsymbol{Y}}^{(m)}_{t^*,\boldsymbol{y}}(T)}{\delta}\right)\overline{\mathcal{Y}}^{(m)}_{t^*,\boldsymbol{y}}(T). \tag{13}$$

In this equation t^* is an arbitrary point from the time interval $[t, T]$. If the $t^* = T$ the forward-reverse estimator collapses to the pure forward estimator (5) and if we take $t^* = t$ we will obtain the reverse estimator (12). The Forward-Reverse Estimator (FRE) is more efficient as compared with the forward or reverse estimators. Under the assumption that $N = M$, $t < t^* < T$ and the kernel function K is a second order kernel, it can be shown that for $d \leq 4$ and the bandwidth $\delta \sim N^{-1/d}\log^{1/d} N$ the forward-reverse estimator has the root-N accuracy $\epsilon(p) \sim N^{-1/2}$. For $d > 4$ the root-N accuracy is lost, but the order of accuracy of the FRE $\epsilon(p) \sim N^{\frac{4}{d+4}}$ (for the bandwidth of order $\delta \sim N^{-\frac{2}{4+d}}$ is still a square of the FE accuracy order. The proof can be found in [7].

3 THE PARALLEL ALGORITHMS FOR DENSITY ESTIMATORS

Because of all realizations of particle's paths are independent the random walk model can be easily parallelized. It makes the particle modelling very attractive for a lot of application. In this section we consider the algorithms of the parallelization for the forward and for the forward-reverse estimators.

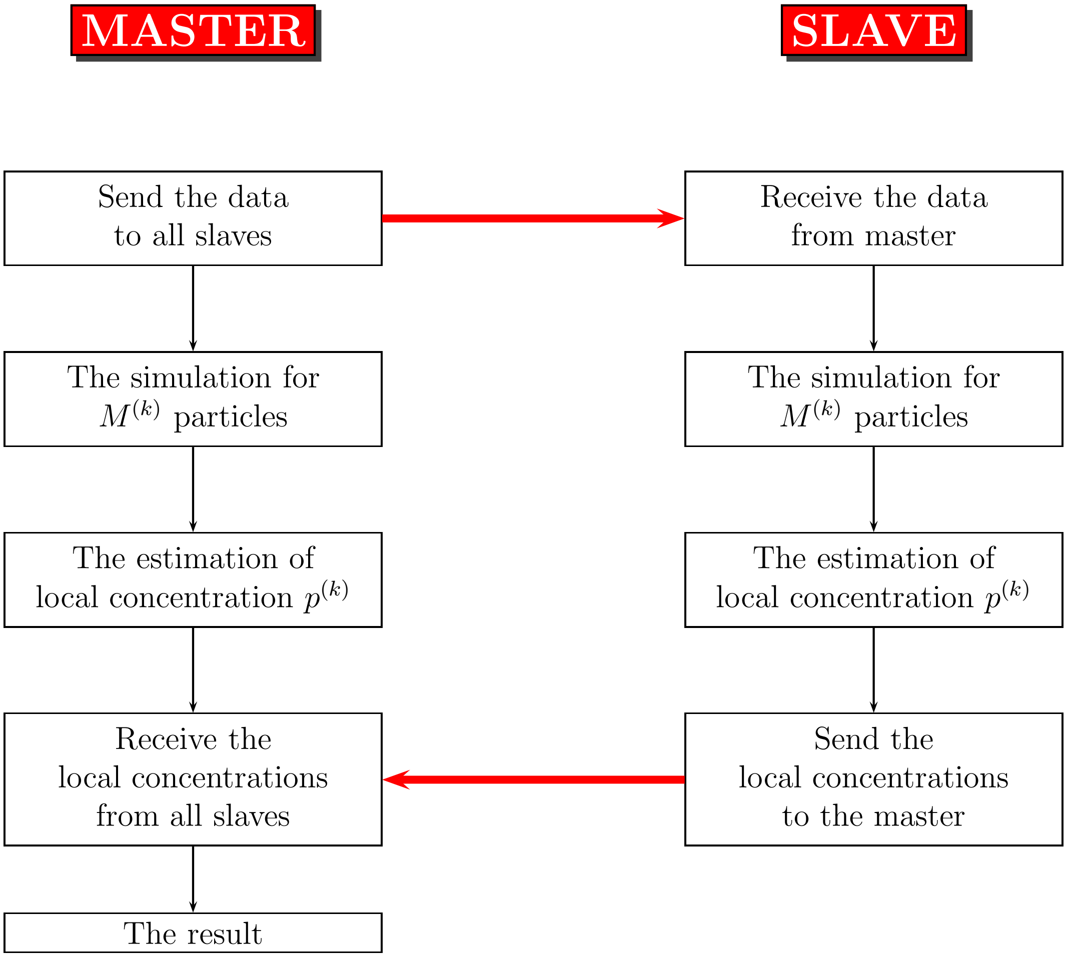
3.1 The parallelization of FE/RE estimators

It is very simple to parallelize the FE/RE estimators (5) and (12). As all particles are independent and do not interact during the simulation, therefore, each processor can simulate the movements of subgroup of particles without communications. However, communications are needed to locate particles on the different processors in the beginning and to gather the results from each process. Equation (5) can be rewritten as follows

$$\begin{aligned}\widehat{p}(t,\boldsymbol{x},T,\boldsymbol{y}) = {} & \frac{1}{N\delta^d}\sum_{n=1}^{N} K\left(\frac{\overline{\boldsymbol{X}}^{(n)}_{t,\boldsymbol{x}}(T) - \boldsymbol{y}}{\delta}\right) = \\ & \frac{1}{N\delta^d}\sum_{k=1}^{Nproc}\left(\sum_{i=1}^{M^{(k)}} K\left(\frac{\overline{\boldsymbol{X}}^{(n,k)}_{t,\boldsymbol{x}}(T) - \boldsymbol{y}}{\delta}\right)\right) = \sum_{n=1}^{Nproc} p^{(k)},\end{aligned} \tag{14}$$

where $Nproc$ is the number of processors, $M^{(k)}$ is a number of particles handled by kth processor, $\sum_{k=1}^{Nproc} M^{(k)} = N$. Each processor gets approximately the same amount of particles, so $M^{(k)} \sim \dfrac{N}{Nproc}$. Finally, $p^{(k)} = \sum_{i=1}^{M^{(k)}} K\left(\dfrac{\overline{\boldsymbol{X}}_{t,\boldsymbol{x}}^{(n,k)}(T) - \boldsymbol{y}}{\delta}\right)$ is the contribution of the kth processor to the sum $\widehat{p}$.

The following algorithm can be used for the parallelization of the FE/RE estimators. We need only two communications and the load is equally distributed. That's why the efficiency of this algorithm is close to 1.


MASTER
SLAVE
Send the data to all slaves
Receive the data from master
The simulation for $M^{(k)}$ particles
The simulation for $M^{(k)}$ particles
The estimation of local concentration $p^{(k)}$
The estimation of local concentration $p^{(k)}$
Receive the local concentrations from all slaves
Send the local concentrations to the master
The result

3.2 The parallelization of the FRE estimator

The FRE can not be parallelized so easy as it has been done for the FE/RE. It is clear that it should be used the particle decomposition.

For the forward-reverse estimator (13) we assume that $M = N$, so we need to simulate the same number of the forward and reverse particles. Therefore, if we evaluate the double sum (13) directly, we need $\mathcal{O}(N^2)$ elementary computations. It makes the forward-reverse estimator too computational expensive. However, it is possible to speedup the calculation, if the kernel function $K(\boldsymbol{x})$ has a compact support, as, for instance, Epanechnikov function (7). If we want to use the Gaussian function (6), we need first to modify it

$$K_\delta(\boldsymbol{x}) = \begin{cases} K(\boldsymbol{x}) \text{ as in (6)} & \sqrt{\boldsymbol{x}^T\boldsymbol{x}} \leq 6\delta \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

This modification does not affect the final result, because for the values $\boldsymbol{x}$, such that $\sqrt{\boldsymbol{x}^T\boldsymbol{x}} > 6\delta$ the value of Gaussian kernel is very small and can be neglected. As a result, in formula (13) we need to consider only pairs $\left(\overline{\boldsymbol{X}}^{(n)}_{t,\boldsymbol{x}}(t^*), \overline{\boldsymbol{Y}}^{(m)}_{t^*,\boldsymbol{y}}(T)\right)$ for which $\left|\overline{\boldsymbol{X}}^{(n)}_{t,\boldsymbol{x}}(t^*) - \overline{\boldsymbol{Y}}^{(m)}_{t^*,\boldsymbol{y}}(T))\right| \leq \alpha\delta$, for some α.

The algorithm proceeds as follows. For a given forward sample $\left\{\boldsymbol{X}^{(n)}\right\}$, $n = 1, \ldots, N$ we define the bounding hyper-rectangle, i.e. the smallest hyper-rectangle in which all realizations lie. This hyper-rectangle is divided in d-dimensional hypercubes with the length of a side equal to $\alpha\delta$. The size of each cell in the obtained grid can be enlarged to make sure that our grid consists of an integer number of cells. Each realization of forward process $\boldsymbol{X}^{(n)}$ is stored in cell is which enclosed it. For each realization of reverse process $\boldsymbol{Y}^{(m)}$ we consider only sample pairs $(\boldsymbol{X}^{(n)}, \boldsymbol{Y}^{(k)})$ where $\boldsymbol{X}^{(n)}$ and $\boldsymbol{Y}^{(k)}$ are in the same grid cell or in neighboring cells. Figure 1 illustrates the example of a 2-dimensional grid.

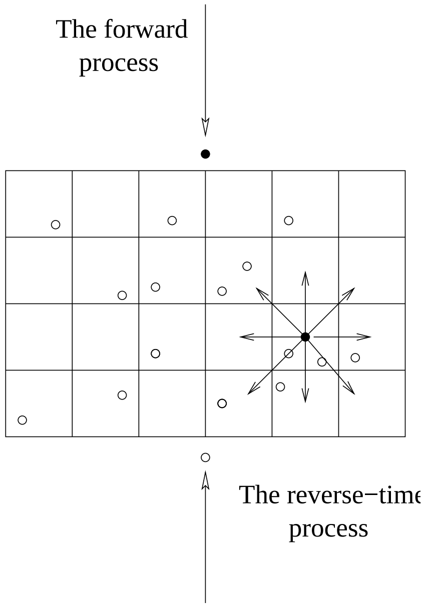


Figure 1: The 2-dimensional grid

Of course, some cells in the grid contains no particle and it has no sense to store them. In case of high-dimensional system the number of empty cells is quite large and saving of such a grid requires a lot of memory. To reduce the memory needed for the grid hash-tables can be used. The further discussion about using of hash-tables for the forward-reverse estimator can be found in [2].

Next we consider the parallelization of the calculations of the forward-reverse estimator. Three different approaches to particle decomposition were considered and implemented.

1. Each processor gets the same amount of particles assigned for the forward as well for the reverse simulation and simulates the random walk of the forward and the reverse particles. Minimal and maximal positions of the forward simulation particles are evaluated for all processors and the corresponding grid is constructed and broadcasted to all processors. The positions of the reverse particles are then broadcasted to all processors and the kernel is estimated locally. In the end, the resulting local kernel estimation is brought together to make the overall result.

2. Each processor is either computing forward or reverse particles. Minimal and maximal positions of forward simulation particles are exchanged among forward processors. The corresponding grid is calculated and sent to all forward processors. Reverse particle positions are broadcasted and the kernel is estimated locally. Finally, the resulting local kernels are brought together.

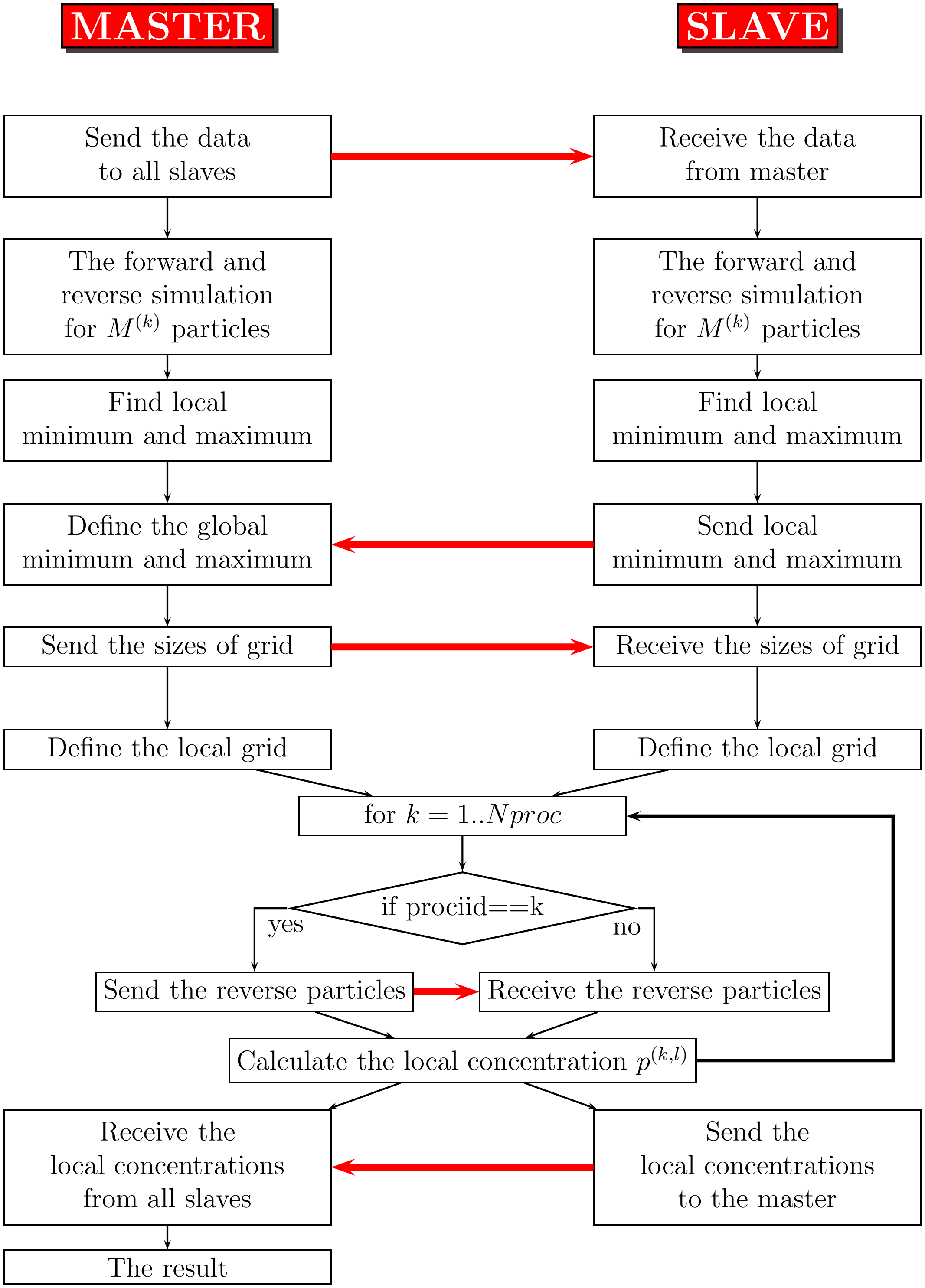
3. One of the processors is assigned to simulate the reverse particles while all the other processors run the forward simulation. The reverse processor keeps simulating while the forward processors exchange grid information.Then the position of the reverse particles are broadcasted and followed by the estimation of the kernel. The more particles are used for the simulation the more runs can be made to balance the work load.

Comparing the three different approaches, the first one seems to have the fewest drawbacks. In the second approach the problem that slows down the calculation is that the reverse processors are lazy while the forward processors estimate the kernel which means that half of the processors are idle during the kernel estimation. In the third approach it is hard to determine how many runs must be made to balance the work load equally. This changes with the number of particles and must be adaptive. The first version seems to be the simplest and all processors are running the same program and is therefore expected to perform the best.

As it has been done for the FE/RE estimator the forward reverse estimator (13) can be rewritten as

$$\begin{aligned}\widehat{p}(t,\boldsymbol{x},T,\boldsymbol{y}) = {} & \frac{1}{N^2\delta^d}\sum_{k=1}^{Nproc}\sum_{l=1}^{Nproc}\left(\sum_{n=1}^{M^{(l)}}\sum_{m=1}^{M^{(k)}} K\left(\frac{\overline{\boldsymbol{X}}_{t,\boldsymbol{x}}^{(n)}(t^*)-\overline{\boldsymbol{Y}}_{t^*,\boldsymbol{y}}^{(m)}(T)}{\delta}\right)\overline{\mathcal{Y}}_{t^*,\boldsymbol{y}}^{(m)}(T)\right) = \\ & \sum_{k=1}^{Nproc}\sum_{l=1}^{Nproc} p^{(k,l)},\end{aligned} \tag{16}$$

where $p^{(k,l)} = \dfrac{1}{N^2\delta^d}\displaystyle\sum_{n=1}^{M^{(l)}}\sum_{m=1}^{M^{(k)}} K\left(\frac{\overline{\boldsymbol{X}}_{t,\boldsymbol{x}}^{(n)}(t^*)-\overline{\boldsymbol{Y}}_{t^*,\boldsymbol{y}}^{(m)}(T)}{\delta}\right)\overline{\mathcal{Y}}_{t^*,\boldsymbol{y}}^{(m)}(T)$


MASTER
SLAVE
Send the data to all slaves
Receive the data from master
The forward and reverse simulation for $M^{(k)}$ particles
The forward and reverse simulation for $M^{(k)}$ particles
Find local minimum and maximum
Find local minimum and maximum
Define the global minimum and maximum
Send local minimum and maximum
Send the sizes of grid
Receive the sizes of grid
Define the local grid
Define the local grid
for $k = 1..Nproc$
if prociid==k
yes
no
Send the reverse particles
Receive the reverse particles
Calculate the local concentration $p^{(k,l)}$
Receive the local concentrations from all slaves
Send the local concentrations to the master
The result

In the above described algorithm all the processors are running the same program which makes it particularly easy to implement and the load is equally distributed. The particles are distributed equally on all the processors. Forward as well as reverse simulation is run on each processor. The number of processors and particles can therefore be changed independently without any modifications in the code. It's the simplest approach, but through its simplicity it performs the best. Also the computation on each processor takes about the same amount of time as they are all running the same piece of code. Only when the position of the forward and reverse simulation particles is known, communication between the processors starts. In case of a long simulation this approach performs especially well. A grid is computed relying on the forward simulation positions. This grid is sent to all processors. Now the position of the reverse simulation is introduced into the grid. This happens by broadcasting the reverse particles. The estimation can now be computed on each processor locally. Then the results are summed together into a global kernel estimation.

4 APPLICATION

4.1 Random walk model for the transport processes in shallow water

The forward and forward-reverse estimators may be used for a number of applications, for instance, for the environmental models [2, 4, 12, 14]. The parallel algorithm for forward-reverse estimator has been applied to estimate the concentration of the pollutant in Dutch coastal zone. In this section we introduce the random walk model for tidally-averaged flow.

Consider now a three dimensional space with points $\boldsymbol{x} = (x_1, x_2)$ and define the vector function $\boldsymbol{a}$ and $\boldsymbol{\sigma}$ according to:

$$\begin{aligned} \boldsymbol{a}(s,x) &= \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{pmatrix} \dfrac{D}{H}\dfrac{\partial H}{\partial x_1} \\ \dfrac{D}{H}\dfrac{\partial H}{\partial x_2} \end{pmatrix} + \begin{pmatrix} \dfrac{\partial D}{\partial x_1} \\ \dfrac{\partial D}{\partial x_2} \end{pmatrix} \\ \boldsymbol{\sigma}(s,\boldsymbol{x}) &= \begin{pmatrix} \sqrt{2D} & 0 \\ 0 & \sqrt{2D} \end{pmatrix} \end{aligned} \tag{17}$$

where $u = (u_1, u_2)$ is the water velocity vector, D is the dispersion coefficient and H is the water depth. The Fokker-Planck equation (2) in this case becomes (see [4]):

$$\frac{\partial p}{\partial s} = -\frac{\partial\left((u_1 + \frac{\partial D}{\partial y_1} + \frac{D\partial H}{H\partial y_1})p\right)}{\partial y_1} - \frac{\partial\left((u_2 + \frac{\partial D}{\partial y_2} + \frac{D\partial H}{H\partial y_2})p\right)}{\partial y_2} + \frac{\partial^2 (Dp)}{\partial (y_1)^2} + \frac{\partial^2 (Dp)}{\partial (y_2)^2}. \tag{18}$$

Substituting $p = HC$ in this equation and rearranging terms yields

$$\frac{\partial (HC)}{\partial s} = -\frac{\partial (u_1 HC)}{\partial y_1} - \frac{\partial (u_2 HC)}{\partial y_2} + \frac{\partial}{\partial y_1}\left(HD\frac{\partial C}{\partial y_1}\right) + \frac{\partial}{\partial y_2}\left(HD\frac{\partial C}{\partial y_2}\right) \tag{19}$$

where C is the particle concentration. This equation is the well-known vertically-integrated advection diffusion equation [4, 11]

4.2 The performance

To evaluate the performance the program was run on up to 8 CPUs. AMD Athlon(TM) XP 2600+, CPU Mhz 1916.547, cache size 512KB Figures 2,3 and 4 show the speed up and efficiency for 1000, 50000 and 100000 particles respectively.

To investigate the behavior of the parallelization with more processors the parallelization was also run on the DAS2 cluster. As it can be seen from figures 5,6 and 7 the results run on DAS2 are consistent with the previous ones. With 1000 particles the parallelization of the simulation is not very effective and efficiency decreases quickly as the number of processors increases.

Because there are no communications between processes during the forward and reverse simulations, we are interested also in the speedup of pure kernel estimator. Figures 5,6 and 7 illustrates the speedup of for the kernel simulation. When increasing the number of particles to 50000 a linear speedup can be observed. With a number of particles on the parallelization shows to be very effective. Better results are achieved when increasing the number of particles to a million. There is even superlinear speedup which happens due to some cache effects.

5 CONCLUSIONS

The forward reverse estimator can be used to make the standard Monte Carlo methods more efficient. However, this method still requires a lot of CPU time. To further speed up the estimator parallel algorithms can be applied. In this paper the parallelizations of the forward reverse estimator are considered and analyzed. The results of applying this parallel algorithm to estimate the concentration of the pollutant in shallow water shows that the efficiency of the program is close to the highest possible. This makes the forward-reverse estimator very attractive for real life applications.

REFERENCES

[1] L. Arnold. *Stochastic differential equations*, Willey, (1974).

[2] E. van der Berg, A.W. Heemink, H.X. Lin and J.G.M. Schoenmakers. Probability density estimation in stochastic environmental models using reverse representations. *Stochastic Environmental Research and Risk Assessment*, **20**, nr. 1/2, 126-134, (2005).

[3] A.W. Jazwinski. *Stochastic differential processes and filtering theory*, Academic Press, New York, (1970).

[4] A.W. Heemink. Stochastic modeling of dispersion in shallow water. *Stochastic Hydrol. and Hydraul.*, **4**, 161-174, (1990).

[5] P.E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*, Springer, (1992).

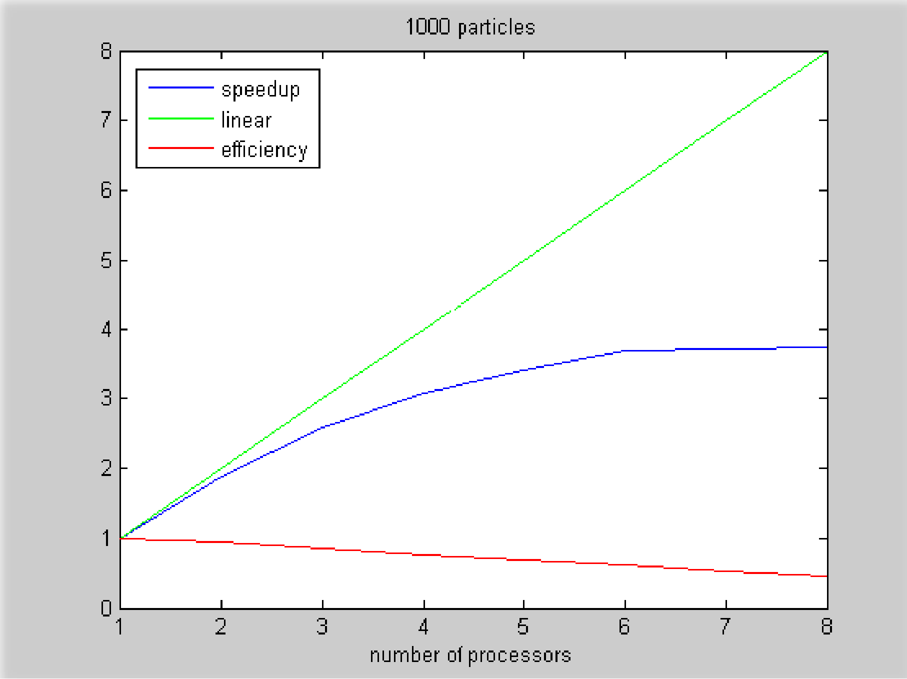


Figure 2: The speed up and efficiency for 1000 particles

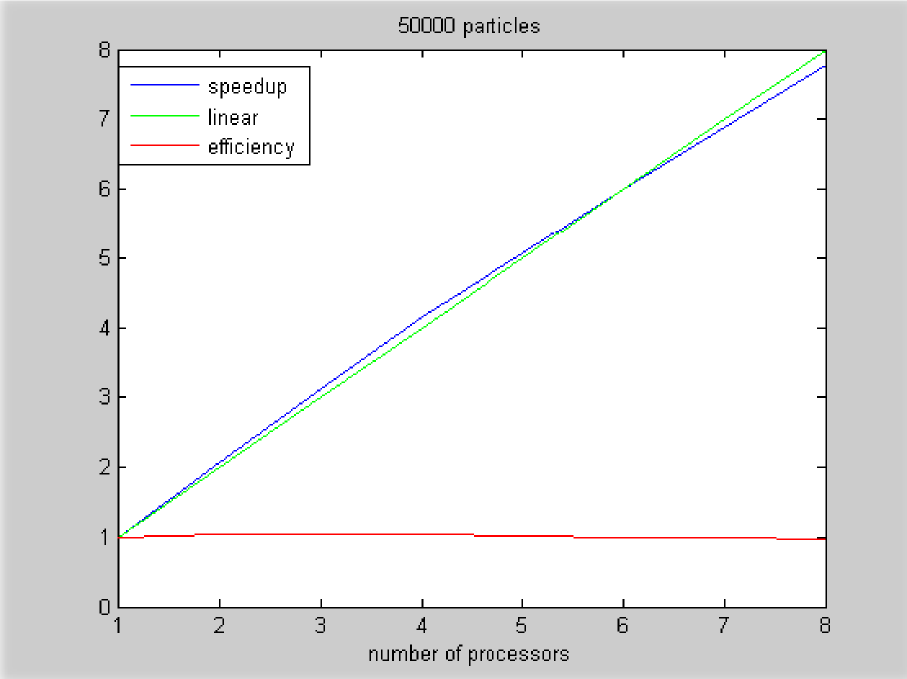


Figure 3: The speed up and efficiency for 50000 particles

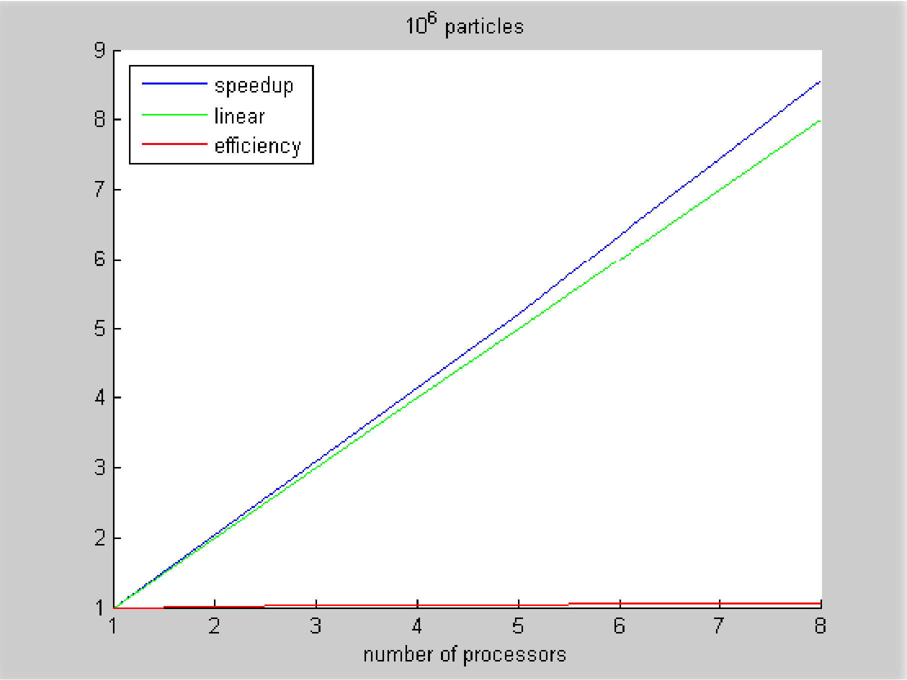


Figure 4: The speed up and efficiency for 10^6 particles

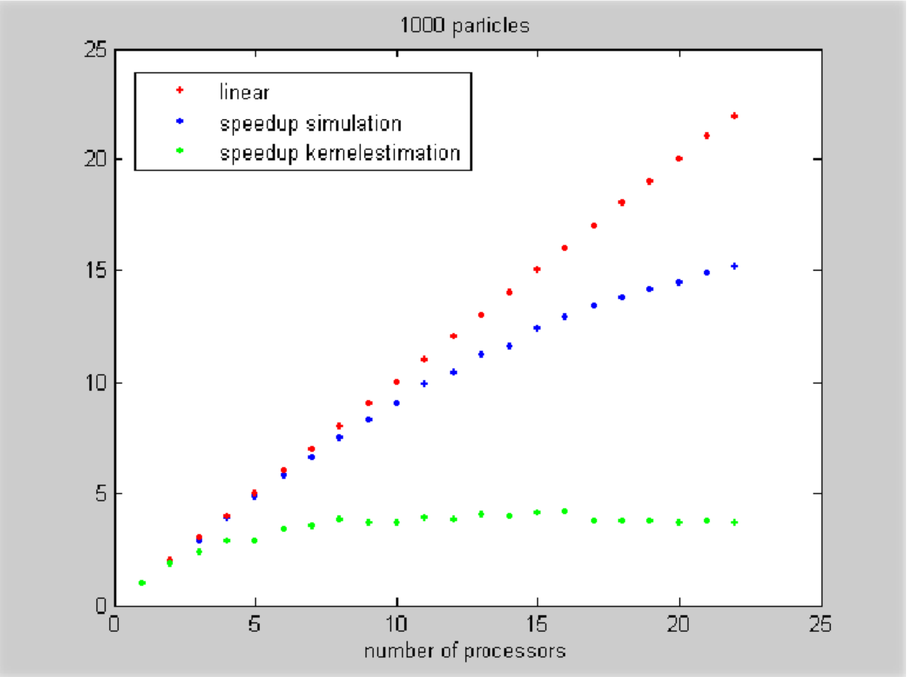


Figure 5: The speed up and efficiency run on DAS2 for 1000 particles

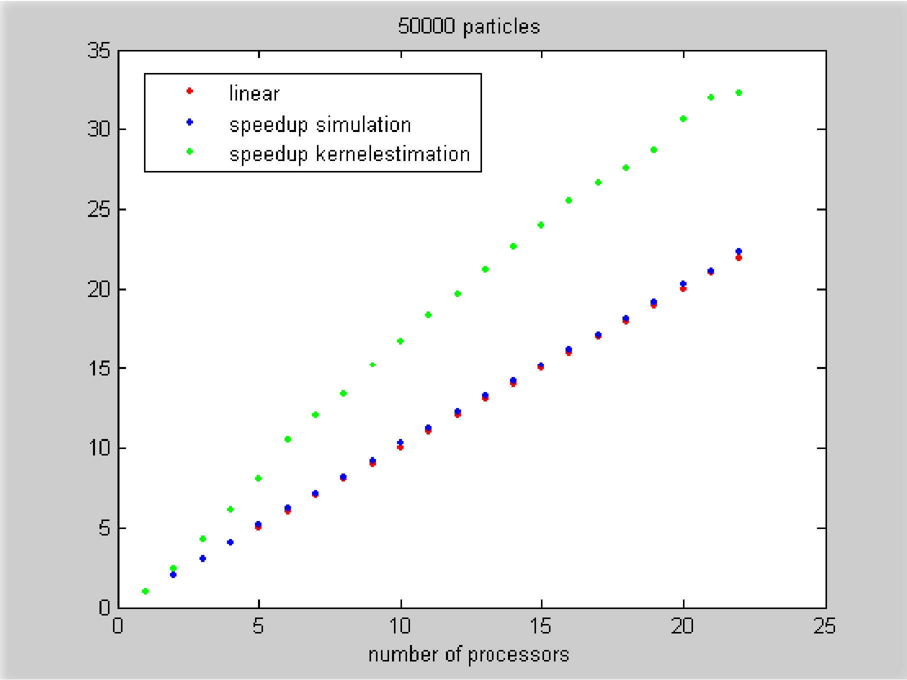


Figure 6: The speed up and efficiency run on DAS2 for 50000 particles

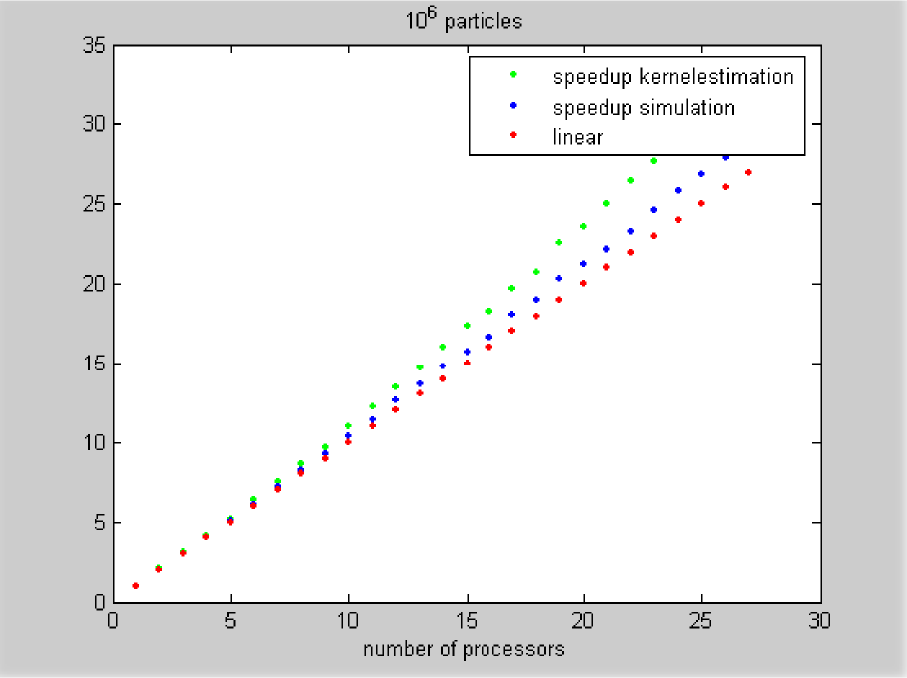


Figure 7: The speed up and efficiency run on DAS2 for 10^6 particles

[6] O.A. Kurbanmuradov, U. Rannik, K. Sabelfeld and T. Vesala. Direct and adjoint Monte Carlo for the footprint problem. *Monte Carlo Methods and Applications*, **5**, nr. 2, 85-111, (1999).

[7] G.N. Milstein, J.G.M. Schoenmakers and V. Spokoiny. Transition density estimation for stochastic differential equations via forward-reverse representations. *Bernoulli*, **10**, nr. 2, 281-312, (2004).

[8] G.N. Milstein and M.V. Tretyakov. *Stochastic numerics for mathematical physics*, Springer, (2004).

[9] B. Oksendal. *Stochastic differential equations*, Springer, (1985).

[10] B.W. Silverman. *Density estimation for statistics and data analysis*, Champan and Hall, (1986).

[11] E.R. Holley. Diffusion and dispersion. In *Environmental Hydralics* (eds. V.P. Singh and W.H. Hager), Kluwer, (1996).

[12] D. Spivakovskaya, A.W. Heemink , G.N. Milstein and J.G.M. Schoenmakers. Simulation of the transport of particles in coastal waters using forward and reverse time diffusion. *Advances in Water Resources*, **28**, 927-938, (2005).

[13] D. Spivakovskaya, A.W. Heemink and J.G.M. Schoenmakers. The risk analysis in coastal zone using reverse-time diffusion. Accepted to the publication in *The proceedings of Computational Methods in Water Resources 2006 International Conference.*

[14] J.W. Stijnen. *Numerical methods for stochastic environmental models*, Ph.D. Dissertation, Delft University of Technology, (2002).

[15] D.J. Thomson. Criteria for the selection of stochastic models of particles trajectories in turbulent flow. *J. Fluid Mech.*, **180**, 529-556, (1987).

[16] M.P. Wand and M.C. Jones. *Kernel smoothing*, Champan and Hall, (1995).