Forward flux sampling (FFS) as workflow application using the Science Experimental Grid Laboratory (SEGL)

Yevgen Dorozhko,[1] Yuriy Yudin,[1] Kai Kratzer,[2] Axel Arnold,[2] Colin W. Glass,[1] and Michael Resch[1]

[1] *Department Numerics and Libraries, High Performance Computing Center Stuttgart (HLRS), 70569 Stuttgart, DE.*
[2] *ICP, Institute for Computational Physics, University of Stuttgart, Allmandring 3, 70569 Stuttgart, DE.*

(Dated: March 12, 2013)

Rare events are difficult to study, because of the long waiting times between their occurrences. Therefore, a lot of sampling techniques have been developed recently. However, the implementation of these techniques together with a simulation tool and calculating on high performance computers turned out to be very difficult. In this article, we succeed in this challenge by exemplarily describing one of these samplinge techniques, namely Forward Flux Sampling (FFS) as a workflow application in the context of the Science Experimental Grid Laboratory (SEGL). SEGL handles the workflow and dataflow of the simulation, and matches the requirements for FFS by drawing a certain number of random datasets from a given dataspace. We demonstrate the usability of this work with an example simulation problem from physics. With this combination, FFS can easily be used together with any simulation software which can be described in the context of the SEGL framework, without the need of implementing the FFS method by the researcher himself, making rare event simulations easily available. Beyond that, trajectory fragments can be calculated in parallel and on high performance computing hardware by design.

I. INTRODUCTION

Rare events happen rapidly compared to the long waiting times between the events and with important consequences for the investigated system, e.g. lightening, vulcanic eruptions and telecommunication failures on the macroscale or nucleation, protein folding and translocation of DNA through nanopores on the microscale. Therefore, it is important to investigate these events. However, in brute force computer simulations, a lot of time is wasted simulating the long waiting times between these fluctuation-driven events, where nothing interesting happens [33]. To overcome this problem, a lot of rare event techniques have been developed for the particular simulation scenarios such as umbrella sampling [1, 2], Bennett-Chandler methods [2–4], transition path sampling [5–7], transition interface sampling [8–10], milestoning [11–13], nudged elastic band [14, 15], string methods [16, 17], weighted-ensemble methods [18], non-equilibrium umbrella sampling or forward flux sampling (or splitting)-type methods [19–28]. The aim of all these methods is the reduction of simulation time which is spent in the long waiting time between the events and concentrate on simulating the interesting domains in phase space.

However, implementing these rare event simulation methods together with the particular scientific simulation code can be very demanding. In this article, we show an elegant way of using such methods together with the scientific simulation code. Therefore, we focus on the forward flux sampling (FFS) approach [22] which we apply to an example problem from physics, namely the barrier crossing of a particle. In FFS, the progress of a reaction from the initial state to the final state is characterized by an order parameter. The region between these two states is subdivided by several interfaces, which are positioned at certain values of the order parameter. Then, the simulation is driven step-wise towards the final state by first firing trajectories from the initial state and storing configurations, if they reach the next interface. From these configurations, new trajectories are fired and so on, until the final state is reached. From the fraction of successful trajectories, the "success probabilities" can be determined, which, together with the flux of the trajectories which managed to leave the initial state in a certain simulation time, give the transition rate from the initial state to the final state. Beyond the transition rate, successful transition trajectories can be extracted from such a simulation, e.g. to investigate reaction pathways.

FFS can be used for both equilibrium and non-equilibrium systems, because it doesn't require *a priori* knowledge of the phase space density) [22]. Recently, several extensions for FFS have been developed, including different algorithms for the trajectory-firing procedure [20], computation of phase-space densities and transition rates [29], analysis of the efficiency [21, 22, 24, 25, 28], enhancing the placement of the interfaces [30] and the development of FFS-like methods for systems which are out of the stationary state [31, 32].

A. The direct FFS algorithm (DFFS)

With FFS, the transition rate k_{AB} from an initial state A to a final state B can be calculated with [8]

$$k_{AB} = \Phi_{\text{esc}} P_B. \tag{1}$$

Here, Φ_{esc} is the escape flux, a quantity to characterize the flux of trajectories leaving the initial state, and P_B is the conditional probability, that a trajectory which successfully left the initial state reaches the final state. Beyond the calculation of the transition rate in FFS, the

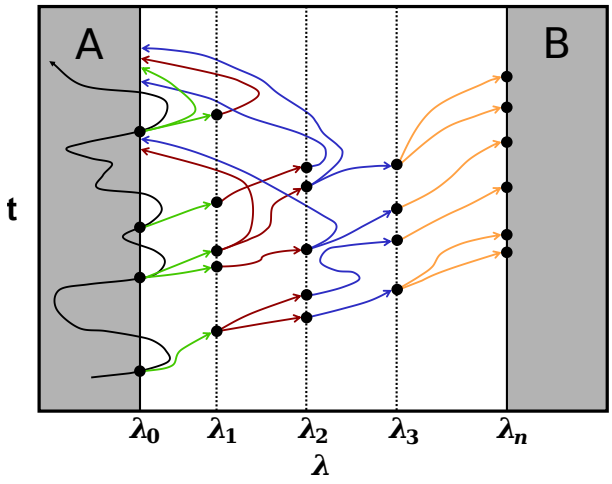

Figure 1: The DFFS algorithm, schematic. The region between the initial state A and the final state B is subdivided by a set of interfaces at specific values of the order parameter λ. The vertical axes denotes simulation time and the arrows depict the trajectory fragments between the interfaces.

transition trajectories can be investigated. To this aim, the region from the initial state A to the final state B is defined in terms of λ, the order parameter, such that the system is in A, if $\lambda < \lambda_A$ and in B, if $\lambda > \lambda_B$, respectively. The area between A and B ($\lambda_A < \lambda < \lambda_B$) is subdivided by a set of n interfaces at specific values of λ, with $\lambda_0 \equiv \lambda_A$, $\lambda_n \equiv \lambda_B$ and $\lambda_i < \lambda_{i+1}$ (Fig. 1). The probability P_B can be expressed as [8]

$$P_B = \prod_{i=0}^{n-1} p_i \qquad (2)$$

where p_i is the conditional probability between the interfaces λ_{i+1} and λ_i. Using FFS, these quantities can be computed in an efficient way. Hence, k_{AB} can be computed. Moreover, the transition trajectories are generated. Recently, several variants of FFS have been developed [20, 22, 28]. Here, we use the direct FFS algorithm (DFFS) [19, 22, 28].

The DFFS algorithm consists of two simulation tasks. In the first task, the escape flux Φ is computed by drawing a configuration of the basin A and starting the simulation while monitoring the crossing frequency on λ_0 of the trajectories. Every time, a crossing of the trajectory happens and the trajectory was in A before the crossing, the configuration on λ_0 is stored. This is continued until a set of N_0 configurations on λ_0 is collected. If the system reaches B during this task, a new state of basin A is drawn and the system is re-equilibrated. For further practical details see ref [22]. In the second task, the conditional probabilities p_i are computed for each interface transition step by step. Therefore, a configuration on the last interface λ_i is drawn at random, and the simulation is started using this configuration point. If the trajectory reaches the next interface λ_{i+1}, this counts as "success" and a new configuration on λ_{i+1} is stored. If the trajectory falls back to A, the number of launched runs is incremented but not the success counter and no new configuration is stored. The fraction of the successful runs divided by the number of the total runs gives then the probability p_i for each interface transition. This procedure is repeated until the final interface at the border of state B is reached. Then, the set of conditional probabilities p_i is complete. After the simulation, the reaction pathways can be reconstructed from the successful trajectories [20, 22].

B. The Science Experimental Grid Laboratory (SEGL)

The Science Experimental Grid Laboratory (SEGL) [?] is a scientific workflow management system, designed with the aim of efficiently supporting complex, data-intensive, multi-dimensional computational experiments across distributed high-performance computing resources. Thus, by specifying the high-level workflow, SEGL can efficiently integrate many different resources and applications into a single computational experiment.

To describe the workflow, the "Grid Concurrent Language" (GriCoL) is used. GriCoL is based on a two-level model for the description of workflows - the control flow level and the data flow level. The control flow level offers the possibility to describe the control flow of the set of tasks within the scope of the workflow. To describe the logic within the control flow level, different types of blocks are offered: solver, transition, fork/merge (parallel branching), cycle, conditional transition and nested experiment. For each such block in the control flow, the user has to describe the underlying data flow. The data flow level description contains detailed information on the computational jobs that will be started on the HPC resource. Also this level includes the input and output data sets, variables that describe different parallel data sets, parameter values for the different data sets and expressions to select and group the data sets. Well-known middleware systems such as Globus Toolkit [?], gLite [?] or Unicore [?] provide a typical task and service oriented approach with rather high latency. Arguably, these middleware systems do not well support dynamical and intensive multi-job scheduling and real-time optimization. Therefore, SEGL features a dedicated middleware layer. In this layer dynamic clustering is implemented, with the help of which it is possible to implement dynamic performance optimization for simulation experiments. Now, we describe the implementation of Forward Flux Sampling into SEGL to make use of all these advantages which SEGL provides.

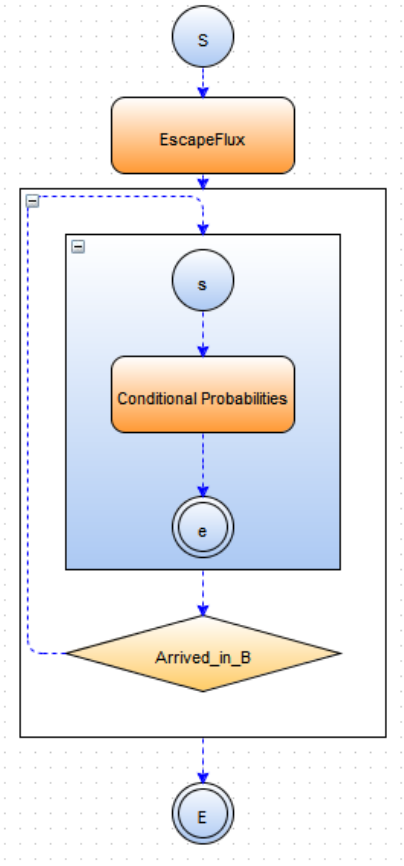

Figure 2: Control flow of FFS implemented in SEGL. The circles denote start (s,S) and end (e,E) points of the particular experiments. Thereby, the lower-case letters correspond to a so-called sub-experiment. The main calculation is performed in the calculation modules (orange-colored rectangles) like described in Sec. II A and Sec. II B. At the end of a calculation stage, the "Arrived in B" module decides, whether to proceed with the next calculation stage or to end the simulation (see Sec. II C).

II. IMPLEMENTATION OF FFS IN SEGL

In this section, we describe how the two-stage DFFS algorithm (see sec. I A) was implemented using SEGL. Thereby, the main setup of the DFFS algorithm, e.g. the positions of the interfaces, is stored in a configuration file.

A. First stage: The escape flux module

In this module, a random configuration in the initial state A of the system is drawn. Therefore, the initial parameterset for FFS is read from a configuration file (Fig. 3). Then, the calculation is performed while monitoring the calculation time which is necessary to calculate the desired number N of configurations on the border of

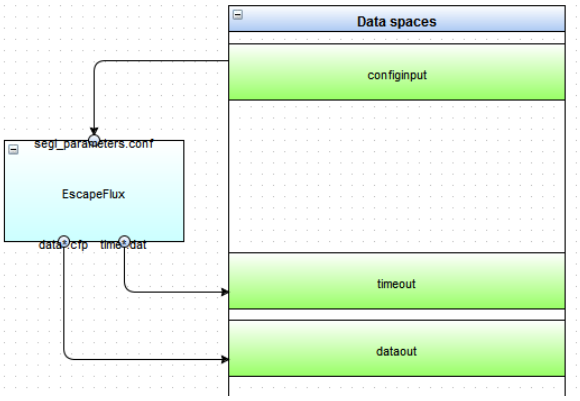

Figure 3: Stage 1: Data flow of the escape flux module. This module uses as input parameters only the configuration file with the general FFS setup. The escape flux module is expected to calculate the configuration points on the first interface while monitoring the calculation time, which was necessary to collect them. These informations are stored as output-files, which are collected by SEGL and stored in the particular dataspaces.

the initial state A which corresponds to the first interface λ_0. The calculation time and the configuration points are then written to datafiles which are collected and stored in the corresponding dataspaces. In our case, our simulation tool has the information, how many configurations are collected, therefore no control structure is needed. If this is not the case, a simple control structure can be added, which queries the number of configurations on λ_0 and decides how to proceed. The outcome of this calculation module is the first quantity of the transition rate, namely the escape flux

$$\Phi_{\text{esc}} = \frac{\text{number of configurations on } \lambda_0}{\text{simulation time}}, \qquad (3)$$

which is a measure, how good our system is in crossing the border of the initial state A. With this information, the first calculation stage is ready and we proceed with the next calculation stage.

B. Second stage: The conditional probabilities module

In this module, a random configuration from the last interface λ_i is drawn, and the calculation is continued using this configuration point as starting point. The simulation is aborted, if either the next interface λ_{i+1} or the initial state λ_0 is reached. Therefore, the simulation has as input the standard configuration file, the index of the current interface and the filename of the random chosen configuration snapshot (see Fig. 4). If the run was successful, a new snapshot on interface λ_{i+1} is stored. This results in a collection of successful snapshots, which are collected and stored in the dataspace. Each cycle of this

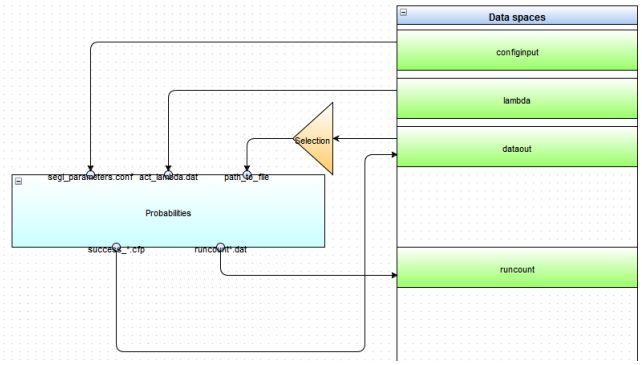

Figure 4: Stage 2: Data flow of the probabilities module. The input parameters are the general set-up of the FFS simulation, the current interface index i to calculate on, and a random chosen configuration point from the dataspace, drawn by the "Selection" filter. If the run was successful by reaching the next interface λ_{i+1}, a new configuration point on λ_{i+1} is produced and stored in the dataspace, ready for beeing drawn in the next iteration. If the number of trial runs is not fixed (e.g. if no successful points are created, the number of trial runs could be increased on-the-fly), the number of trial runs launched is stored in the dataspace "runcount" for later postprocessing.

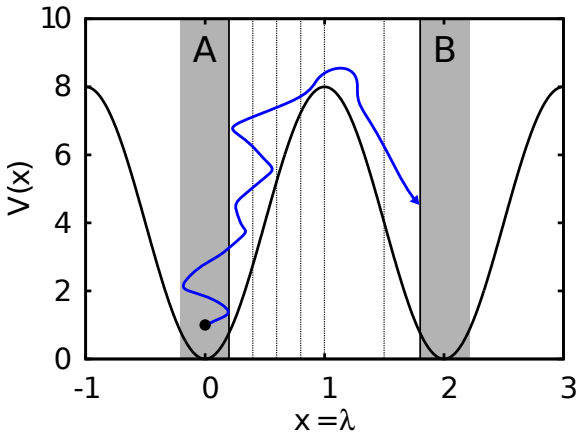

Figure 5: Barrier crossing of a single particle, initially set up in region A. The potential $V(x)$ is shown exemplarily for a height of $h = 8k_BT$. If the barrier is large compared to the energy of the particle, crossings become very rare. The reaction coordinate λ is the x-coordinate of the particle in this case. For FFS, the way from A to B is partitioned by several interfaces (dashed lines). Thereby, more interfaces must be placed in bottleneck regions, here the ascent of the potential.

subexperiment module results in a contribution to the overall transition rate k_{AB}: The probability to reach the next interface λ_{i+1} coming from the previous one, λ_i, is given by

$$p_i = P(\lambda_{i+1}|\lambda_i) = \frac{\text{number of successful trials}}{\text{number of total trials}}. \quad (4)$$

These quantities are calculated during the postprocessing, see sec. II D.

C. The "arrived in B" decision module

In this module, we check, if further simulations are necessary or if the simulation is finished. If the calculation on interface λ_i is complete, the index counter i for the current interface staging is incremented and the simulation is started again with the new configuration set. If the last interface was the border of the final state $\lambda_B = \lambda_n$, the simulation is ended and the simulation data can be extraced for postprocessing.

D. Postprocessing

After the run, the collected simulation data can be examined. The total transition rate k_{AB} from the initial state A to the final state B is given by Eq. (1). To extract the escape flux from the first calculation stage, the number of configuration points on λ_0 is divided by the simulation time stored in the "time" dataspace of the escape flux calculation module. The probabilities p_i of the second stages are extraced from the number of successful configuration snapshots in each interface dataspace divided by the number of total runs launched.

From the collected datasets, the reaction pathways (transition trajectories) can be extracted by backtracing the configuration snapshots. Therefore, the user can choose appropriate names for the datafiles, or store the information inside the datafile. For backtracing it is sufficient to store the name of the last snapshot, where the trajectory started, in the datafile. Now, we will show how this implementation works in practice with an example from physics.

III. EXAMPLE: SINGLE PARTICLE BARRIER CROSSING

We use FFS in combination with SEGL to push a single particle moving in one dimension over an energy barrier (see Fig. 5). If the barrier is large compared to the energy of the particle, this problem can be classified as a rare event. The potential in our case is defined by

$$V(x) = (h/2)\left[1 - \cos(\pi x)\right] \quad (5)$$

for x in the range $[-1, 3]$ and has two minima, at $x = 0$ and $x = 2$ (see Fig. 5). We set the maximal height of the potential barrier, at $x = 1$, to $h = 10k_BT$, where k_B is the Boltzmann factor and T the temperature. Therefore, crossings of the particle are very unlikely. The particle, which is initially placed in the region $-0.2 < x < 0.2$ which corresponds to the phase space region A in FFS, undergoes underdamped Langevin dynamics, which re-

Index i	0	1	2	3	4	5	6	7	8	9	10
λ_i	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	2.0

Table I: Interface positions of the FFS simulation. The main part of the interfaces is placed at the steep part of the potential coming from the initial state A. If the barrier is overcome, the probability to reach the final state B, beginning with $\lambda >= 2.0$ is very high. Therefore, less interfaces are needed for $1.0 < \lambda < 2.0$.

i	number of trials	successful	p_i
1	1000	324	0.324
2	1000	245	0.245
3	1000	253	0.253
4	1000	199	0.199
5	1000	227	0.227
6	1000	267	0.267
7	1000	385	0.385
8	1000	603	0.603
9	1000	864	0.864
10	1000	917	0.917

Table II: Number of trials per interface, successful trials with configuration points on λ_i and conditional transition probabilities to these interfaces p_i.

sults in the following equation of motion:

$$m\ddot{x} = -\nabla V(x) - \gamma m\dot{x} + R(t)\sqrt{2m\gamma k_B T}, \qquad (6)$$

where m is the mass of the particle, $\ddot{x}$ is the acceleration, $-\nabla V(x)$ is the force on the particle derived from the potential $V(x)$, $\dot{x}$ is the velocity, and γ is the friction coefficient. The Langevin thermostat is a stochastic thermostat, which models the fluctuation behavior of the particle given by the finite temperature T, and results in different trajectories which are necessary for FFS by the Gaussian-distributed random term $R(t)$[34]. In our case, we set $m = 1$, $\gamma = 1$ and $k_B T = 1$. Moreover, we use the verlocity verlet as integration scheme with a timestep of $dt = 0.001$. Our reaction coordinate λ is taken to be the position x of the particle and the borders of the initial and final states are defined by $\lambda_A = 0.2$ and $\lambda_B = 2.0$. Most of the interfaces are located at the steep part of the potential when coming from A, because in FFS, more interfaces should be placed in bottleneck regions [21, 30]. For the locations of the interfaces, see table I.

The result of the FFS calculation for the escape flux is $\Phi_{\text{esc}} = 0.424\tau^{-1}$, with the measuring unit of simulation time, τ. An overview over the conditional transition probabilities p_i per interface is given in table II.

The conditional probabilities are multiplied (see Eq. (2)), which gives us the probability $P_B = 4.455 \times 10^{-5}$. Hence, the transition rate for the event from A to B is

$$k_{AB} = 1.89 \times 10^{-5}\tau^{-1}$$

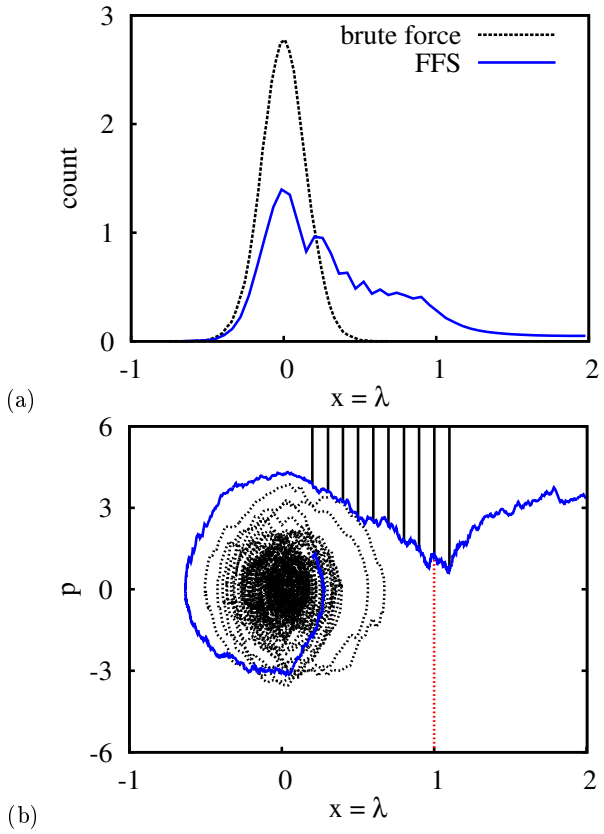

Figure 6: (a) Histogram of the distribution of the reaction coordinate $\lambda = x$. The solid line shows the histogram of the FFS simulation, the dashed line from a brute force simulation with the same number of simulation steps. The brute force simulation stays in the potential minima domain A, whereas the order parameter of the FFS simulation is distributed up to the border of basin B at $\lambda_B = 2.0$. (b) Phase space trajectory of a part of the brute force simulation run (dashed line) and a successful FFS trajectory (solid line). Thereby, p is the momentum of the particle. The solid vertical lines depict the position of the FFS interfaces, the dashed vertical line is at the position of the energy barrier, where the trajectory of the FFS run also has a minimum.

This value of k_{AB} means, that we have to wait on average $5.30 \times 10^4\tau$ to observe a single crossing event at the used barrier height of $h = 10k_B T$. To compare the FFS results against a conventional simulation run, a brute-force simulation was performed with the same number of integration steps like the FFS simulation needed in total, including unsuccessful runs. The distribution of the order parameter $x = \lambda$ is shown in fig. 6(a). As one can clearly see, the brute-force simulating does not manage to cross the barrier as often as in the FFS simulation with the same number of simulation steps. In contrast, the order parameter of the FFS run is distributed in the whole region between A and B. Fig. 6(b) shows exemplarily a section of the trajectory of the particle for the brute-force simulation and a successful trajectory for the FFS-simulation in phase space. If one is interested in the

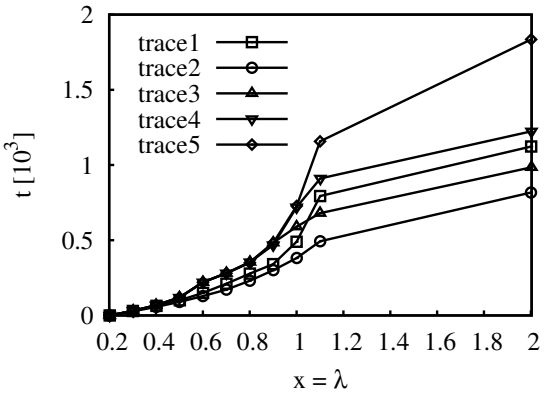

Figure 7: Sampling tree diagram of 5 successful trajectories, simulation time t in calculation steps against order parameter λ in terms of x. The interfaces are located at the datapoint positions. It is clearly visible, that the trajectories need a different amount of calculation steps to arrive at the final state $\lambda_B = 2$ because of the stochastic Langevin thermostat of the simulation. The successful runs can be backtraced to investigate the reaction pathways of the system.

reaction pathways, all the successful trajectories can be backtraced. This is exemplarily shown for 5 trajectories in fig. 7. In principle, if all the information which is necessary to reproduce a trajectory is stored in the configuration points, the whole successful simulation runs can be reproduced and different reaction pathways can be studied.

IV. DISCUSSION

We demonstrated the usage of Forward Flux Sampling (FFS) together with the Science Experimental Grid Laboratory (SEGL). The great benefit of this scheme is, that FFS must not be implemented in the researcher's simulation code. In principle, this scheme can be used with any simulation which can be described by the SEGL framework. Therefore, appropriate calculation modules for calculating the escape flux and the conditional probabilities must be created. Control flow, data flow, parallelization of the trajectory fragments and queuing the jobs on high performance hardware are handled by SEGL.

An issue of the parallelization of FFS is, that the collection of datapoints on the last interface must be complete before advancing to the next interface. This can lead to situations, where no new jobs can be started, before the last run has completed, which is undefined, because of the unknown simulation time (trajectory fragments must end either on the next interface or in A and have no fixed number of simulation steps). To overcome this issue, other flavors of FFS have been developed, like the branched growth FFS, but there, the control structure is more difficult to implement in SEGL. However, this parallel implementation should be in any case faster than the conventional serial implementation.

In principle, beyond the implementation of FFS, other sampling methods can be described using SEGL, which work in a similar way. The main thing which must be implemented is the way, how the next configuration is drawn from the present dataset. E.g. in the Stochastic Process Rare Event Sampling (S-PRES) method, the next configurations are drawn according to an associated weight. Therefore, this weight criterion should be somehow queriable in the control flow. The great advantage of this method would be the fixed simulation time of the runs, and that this method is also applicable for non-quasistatic processes.

Acknowledgments

A.A. and K.K. were funded by the cluster of excellence "SimTech", University of Stuttgart.

[1] G. M. Torrie and J. P. Valleau, Chem. Phys. Lett. **28**, 578 (1974).

[2] D. Frenkel and B. Smit, *Understanding Molecular Simulation. From Algorithms to Applications* (Academic Press, Boston, 2002), 2nd ed.

[3] D. Chandler, J. Chem. Phys. **68**, 2959 (1978).

[4] C. H. Bennett, in *Algorithms for Chemical Computations, ACS Symposium, Series No.46*, edited by R.Christofferson (American Chemical Society, Washington, D.C., 1977).

[5] C. Dellago, P. G. Bolhuis, F. S. Csajka, and D. Chandler, J. Chem. Phys. **108**, 1964 (1998).

[6] C. Dellago, P. G. Bolhuis, and P. L. Geissler, Adv. Chem. Phys. **123**, 1 (2002).

[7] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler, Annu. Rev. Phys. Chem. **53**, 291 (2002).

[8] T. S. van Erp, D. Moroni, and P. G. Bolhuis, J. Chem. Phys. **118**, 7762 (2003).

[9] T. S. van Erp and P. G. Bolhuis, J. Comp. Phys. **205**, 157 (2005).

[10] T. S. van Erp, Comp. Phys. Commun. **179**, 34 (2008).

[11] A. K. Faradjian and R. Elber, J. Chem. Phys. **120**, 10880 (2004).

[12] A. M. A. West, R. Elber, and D. Shalloway, J. Chem. Phys. **126**, 145104 (2007).

[13] E. Vanden-Eijnden, M. Venturoli, G. Ciccotti, and R. Elber, J. Chem. Phys. **129**, 174102 (2008).

[14] G. Henkelman, B. P. Uberuaga, and H. Jonsson, J. Chem. Phys. **113**, 9901 (2000).

[15] G. Henkelman and H. Jonsson, J. Chem. Phys. **113**, 9978 (2000).

[16] W. E, W. Ren, and E. Vanden-Eijnden, Phys. Rev. B **66**, 052301 (2002).

[17] W. E, W. Ren, and E. Vanden-Eijnden, J. Phys. Chem. B **109**, 6688 (2005).

[18] G. A. Huber and S. Kim, Biophys. J. **70**, 97 (1996).

[19] R. J. Allen, P. B. Warren, and P. R. ten Wolde, Phys. Rev. Lett. **94**, 018104 (2005).

[20] R. J. Allen, D. Frenkel, and P. R. ten Wolde, J. Chem. Phys. **124**, 024102 (2006).

[21] R. J. Allen, D. Frenkel, and P. R. ten Wolde, J. Chem. Phys. **124**, 194111 (2006).

[22] R. J. Allen, C. Valeriani, and P. R. ten Wolde, Journal of Physics: Condensed Matter **21**, 463102 (2009).

[23] C. Valeriani, R. J. Allen, M. J. Morelli, D. Frenkel, and P. R. ten Wolde, J. Chem. Phys. **127**, 114109 (2007).

[24] E. E. Borrero and F. A. Escobedo, J. Phys. Chem. B **113**, 6434 (2009).

[25] E. E. Borrero and F. A. Escobedo, J. Chem. Phys. **127**, 164101 (2007).

[26] A. Dickson and A. R. Dinner, Annu. Rev. Phys. Chem. **61**, 441 (2010).

[27] A. Warmflash, P. Bhimalapuram, and A. Dinner, J. Chem. Phys. **127**, 154112 (2007).

[28] F. A. Escobedo, E. E. Borrero, and J. C. Araque, Journal of Physics: Condensed Matter **21(33)**, 333101 (2009).

[29] C. Valeriani, Ph.D. thesis, FOM AMOLF (2007).

[30] K. Kratzer, A. Arnold, and R. J. Allen, submitted (2012).

[31] J. T. Berryman and T. Schilling, J. Chem. Phys. **133**, 244101 (2010).

[32] N. B. Becker, R. J. Allen, and P. R. ten Wolde, J. Chem. Phys. **136**, 174118 (2012).

[33] The long waiting times between the rare events are also a problem in experiments, not only in computer simulations.

[34] The random term in the Langevin thermostat must have zero mean and must be delta-correlated