Load Balancing in Large Scale Bayesian Inference

Daniel Wälchli
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
wadaniel@ethz.ch

Sergio M. Martin
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
martiser@ethz.ch

Athena Economides
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
eceva@ethz.ch

Lucas Amoudruz
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
amlucas@ethz.ch

George Arampatzis
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
Collegium Helveticum, 8092 Zurich, Switzerland.
arampatzis@ethz.ch

Xin Bian
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
bianx@ethz.ch

Petros Koumoutsakos[*]
Computational Science and Engineering Laboratory, ETH Zürich, CH-8092, Switzerland.
Collegium Helveticum, 8092 Zurich, Switzerland.
John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA.
petros@ethz.ch

ABSTRACT

We present a novel strategy to improve load balancing for large scale Bayesian inference problems. Load imbalance can be particularly destructive in generation based uncertainty quantification (UQ) methods since all compute nodes in a large-scale allocation have to synchronize after every generation and therefore remain in an idle state until the longest model evaluation finishes. Our strategy relies on the concurrent scheduling of independent Bayesian inference experiments while sharing a group of worker nodes, reducing the destructive effects of workload imbalance in population–based sampling methods.

To demonstrate the efficiency of our method, we infer parameters of a red blood cell (RBC) model. We perform a data-driven calibration of the RBC's membrane viscosity by applying hierarchical Bayesian inference methods. To this end, we employ a computational model to simulate the relaxation of an initially stretched RBC towards its equilibrium state. The results of this work advance upon the current state of the art towards realistic blood flow simulations by providing inferred parameters for the RBC membrane viscosity.

We show that our strategy achieves a notable reduction in imbalance and significantly improves effective node usage on 512 nodes of the CSCS *Piz Daint* supercomputer. Our results show that, by enabling multiple independent sampling experiments to run concurrently on a given allocation of supercomputer nodes, our method sustains a high computational efficiency on a large-scale supercomputing setting.

[*]Corresponding author.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel algorithms**; • **Mathematics of computing** → **Statistical software**; • **Applied computing** → *Life and medical sciences.*

KEYWORDS

Bayesian Inference, Uncertainty Quantification, Load Balancing, High Performance Computing, Erythrocyte Membrane Viscosity

ACM Reference Format:
Daniel Wälchli, Sergio M. Martin, Athena Economides, Lucas Amoudruz, George Arampatzis, Xin Bian, and Petros Koumoutsakos. 2020. Load Balancing in Large Scale Bayesian Inference. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '20), June 29-July 1, 2020, Geneva, Switzerland.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3394277.3401849

1 INTRODUCTION

Bayesian inference of parameters of physical models is an essential part of evidence-based modeling of physical systems. At the same time it requires extensive model evaluations as it relies on sampling of the parameter space through Monte Carlo methods. These methods imply the repeated evaluation of thousands (or even millions) of independent model evaluations (samples), in repeated phases (generations) to produce a faithful representation of their associated uncertainty. This type of inference has become more viable in the recent years thanks to the availability of large supercomputers. The samples can be easily evaluated across different nodes thus making such approaches highly suited to large scale computing architectures. However, sampling methods can be highly inefficient in the presence of a parameter-based load imbalance in the computational model.

Load imbalance can be particularly destructive in generation based UQ methods such as Bayesian Annealed Sequential Importance Sampling (BASIS) since all compute nodes in a large-scale allocation have to synchronize after every generation and therefore remain in an idle state until the longest model evaluation finishes. The effect is a drastic decrease in the overall usage of computational resources, even when the computational model is, itself, efficient. This inefficiency is a waste of node hours, detrimental to high-throughput, and increases energy waste, a primary concern towards the upcoming era of exascale computing [1].

We demonstrate the capabilities of our approach by inferring the membrane dissipation of a RBC model. The model is employed to simulate the relaxation of an initially stretched RBC towards its equilibrium shape. We analyze five independent data sets from experimental studies [2, 3] to perform the parameter inference in the context of hierarchical Bayesian inference as, for example, described in [4]. To the best of our knowledge, a Bayesian UQ for the membrane dissipation of the RBC has not been performed before. Previous computational studies of RBCs relied on distinct fits to experimental data yielding a range of plausible values for the membrane viscosity, $\eta_m = [0.05 - 0.8] \times 10^{-6}\,\mathrm{Pa\,s\,m}$ [2, 5]. The sources of the discrepancy between estimations of η_m are not the focus of this study. Instead, we concentrate on inferring a probability distribution for the membrane dissipation, given two independent sources of experimental data and the particular RBC computational model.

The software implementation of the RBC model is optimized for graphics processing unit (GPU) architectures. The runtime of this model is in direct correlation with the inferred parameter and thus shows an uneven distribution of work among samples. The resulting workload imbalance reduces the effectively used computational resources, keeping GPUs busy only 74% of the time.

This article presents a novel strategy to deal with the problem of load imbalance for population based methods. Our approach relies on the concept of *oversubscription* of computational resources, where more computational tasks than available workers are created. Through oversubscription, processors remain busy even when parts of the parallel application need to synchronize and wait for the exchange of data [6]. We extend this concept to BASIS, by enabling multiple independent sampling experiments to run concurrently under the same group of worker nodes, keeping nodes busy at almost all times.

We report the improvements obtained with our proposed method in terms of effective core usage, total node hours required, and power consumption. We demonstrate that different job scheduling methods, including the one presented in this paper, play a notable role in solving the problem of load imbalance in our RBC study. We furthermore demonstrate that our approach improves the node usage efficiency from 74% to 98%, and validate it on 512 nodes of the CSCS *Piz Daint* supercomputer.

2 STUDY CASE: RBC RELAXATION

The RBC membrane consists of a lipid bilayer anchored on a network of proteins called the membrane skeleton (MS), both of which contribute to highly non-linear elastic mechanics. In physiological conditions, RBCs are surrounded by plasma, a water-rich liquid which is essentially Newtonian. The RBC contains a solution of hemoglobin, which also behaves as a Newtonian fluid. Consequently, RBCs are highly deformable, allowing for complex dynamical transitions in response to external disturbances [7]. The dynamics of single cells in simple flow conditions lay the foundation for understanding more complex behaviors, existing in flows through capillaries, vascular networks, and the flow in microfluidic devices. Therefore, to understand the complex rheology of the whole blood, we have to model the dynamics of single RBCs accurately.

For half a century, there have been many ingenious developments on RBC membrane mechanics. For example, the model proposed by Lim et al. [8] represents a remarkable breakthrough, combining the complete area-difference elasticity (ADE) model of the lipid bilayer and a non-linear strain-hardening model of MS. This RBC model is termed as Lim-Wortis-Mukhopadhyay (LWM) model for brevity. With a single parameter for the bending reference state to reflect the surrounding solution status, such as PH value or other chemical agents, the model can generate equilibrium configurations for the whole stomatocyte-discocyte-echinocyte (SDE) sequence, which have been unanimously observed in experiments [9]. Although the original study of the LWM model was based on Monte Carlo sampling of the energy landscape, a recent numerical study has already put forth the bending force computation due to the lipid bilayer in the model [10]. Following this line, we further developed the force computation arising from the in-plane elasticity of the MS, which allows for simulations of the LWM model coupled with a dissipative particle dynamics (DPD) fluid solver in a dynamic context.

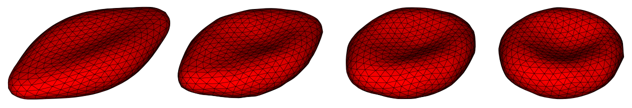
Figure 1: Illustration of the relaxation sequence of a RBC. *Left*: Initial elongated shape. *Middle*: Intermediate shapes during the relaxation process. *Right*: Relaxed biconcave shape.

Increasing evidence from experimental [2, 11], theoretical [12], and computational studies [13–15], indicate that a significant contribution on the total viscous dissipation arises from the lipid bilayer membrane, in addition to the commonly acknowledged contributions from the outer and inner viscous fluids of the RBCs. However, in many computational studies for RBCs, membrane viscosity is neglected [16, 17]. To this end, we study the pairwise dissipative interactions controlled by the dissipative parameter, γ_C, between neighboring vertices on the triangulated mesh, inspired by the pioneering work of a coarse-grained perspective [18, 19]. The complete mathematical model of the RBC is described in detail in Appendix B.

One of the unambiguous experiments to determine the RBC membrane viscosity is the relaxation of a stretched RBC to its equilibrium shape, as demonstrated in Fig. 1. First, the RBC is stretched by pulling two opposite sides of the cell with micro-beads [3] or micro-pipettes [2]. After the release of the external force, the RBC restores its original biconcave shape. The time of this relaxation process results from an interplay between the elastic energy and viscous dissipation of the RBC membrane. The relaxation time increases proportionally to the viscosity, and inversely to the elastic energy [2, 20]. To obtain a reliable estimation (Section 2.1) of the RBC membrane viscosity we systematically infer the membrane dissipation parameter γ_C, based on five experimental data set from [3] and [2].

2.1 Statistical Method

Our goal is to estimate the parameter γ_C and its uncertainty based on five experimental data sets. Due to the presence of heterogeneous data, we formulate the inference problem as a *hierarchical Bayesian inference problem* [21]. We assume that for each data set we have to infer a different parameter for the computational model ($\boldsymbol{\vartheta} := \gamma_C$). Additionally, we assume that each $\boldsymbol{\vartheta}_i$, $i = 1, \ldots, 5$, follows the same distribution that is controlled by a hyperparameter vector ($\boldsymbol{\psi}$). In Appendix A we describe in detail the Bayesian framework and the necessary statistical assumptions. The statistical setup for our study case can be found in Appendix B.4.

In order to sample the individual parameters ($\boldsymbol{\vartheta}_i$, $i = 1, \ldots, 5$) for each experiment, we use a sampling technique based on importance sampling [22]. In the first step of the algorithm we sample the distribution of the parameters $\boldsymbol{\vartheta}_i$ conditioned on the i-th data set as if they were independent. These distributions will serve as the importance sampling distribution. In the second phase, we sample the vector of hyperparameters conditioned on all the available data. Finally, we use the samples of the hyperparameters to obtain samples for each $\boldsymbol{\vartheta}_i$ conditioned on all data. The details of this approach are discussed in Appendix A.2.

We perform the sampling using BASIS [21], a reduced bias variant of the Transitional Markov Chain Monte Carlo (TMCMC) algorithm [23]. The choice of BASIS is supported by the fact that it is one of the most efficient Markov chain Monte Carlo (MCMC) algorithms in the context of Bayesian uncertainty quantification that is targeted to parallel computing architectures. Efficiency is a major requirement since every MCMC step is coupled with a computationally expensive RBC relaxation simulation.

BASIS is a population-based algorithm that successively samples from intermediate annealed distributions that eventually converge to the target distribution. On each *generation*, samples from the previous generation are being resampled. From each sample a small MCMC chain (*e.g.,* of length one) is generated in order to rejuvenate the sample-set. The algorithm is approximate and in the limit of infinitely many samples or intermediate steps is exact.

Although chains in BASIS are independent from each other and can run in parallel, this algorithm is particularly susceptible to the problem of load imbalance. We discuss this problem in the following section.

3 PROBLEM: LOAD IMBALANCE

In parallel algorithms, many processors work simultaneously to reach a common goal. During runtime, processors synchronize –wait for each other– at given points to, for example, exchange data required to continue the computation. Whenever one processor takes longer than the others to reach a given synchronization point, all other computational resources have to idle. Such idling time represents a waste of computational resources and delays the desired results. The non-uniform distribution of workload among computational resources, known as *Load imbalance*, is often the cause for such a problem in parallel algorithms [24].

3.1 Imbalance on generation-based algorithms

Population-based methods belong to the category of *embarrassingly parallel* [25] algorithms. These methods rely on the execution of many independent tasks that can be executed in parallel without synchronization. As a consequence, the performance of these methods is expected to scale perfectly with the number of computational resources. However, in *generation based* algorithms all sample evaluations of the same generation need to be synchronized before starting the next. Therefore, the runtime of one generation is dominated by the sample evaluation with the longest runtime.

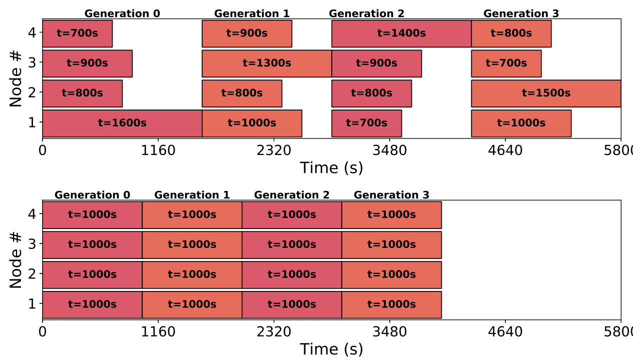

Figure 2: An illustrative example of two execution timelines of BASIS, each running four generations and four samples per generation. On top, an experiment with load imbalance. On the bottom, illustration of an experiment with perfect work balance. Even though the computational demands of both experiments are the same, the imbalanced experiment takes 1800 seconds longer to complete.

The impact of load imbalance in a generation based algorithm is exemplified by the timelines shown in Fig. 2. These timelines represent two experiments where the workload is unfairly distributed (top) and perfectly balanced (bottom). The model parameters which directly impact the runtime of a sample evaluation amplify the load

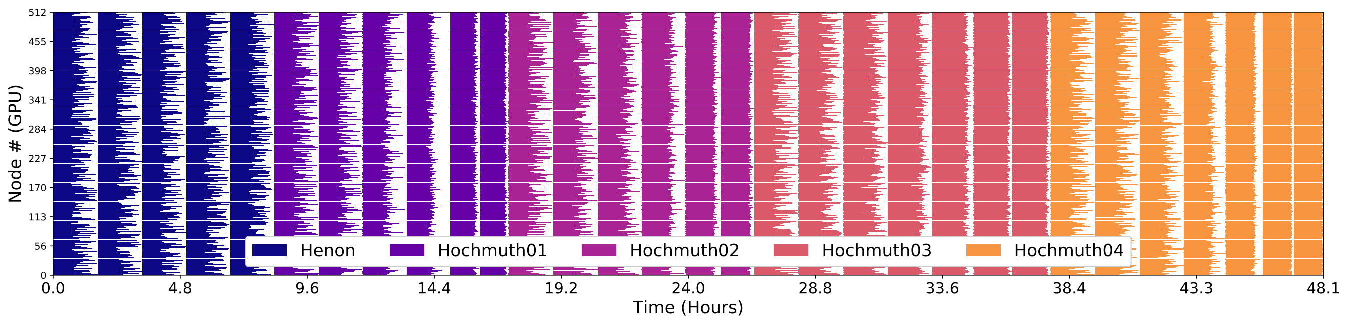

Figure 3: Core usage timeline of sequentially scheduled BASIS experiments. The horizontal axis represents the elapsed time (hours) from the start of the experiment. On the vertical axis, each line represents a different *Piz Daint* node. Whenever the line is colored, this means that Mirheo is actively using the node's GPU to evaluate a given sample. Different colors represent different experiments, starting from the left with Henon (darkest shade), and ending on the right with Hochmuth04 (lightest shade). Blank spaces represent times where the node is idle, waiting for the next sample to be processed.

imbalance in population and generation based sampling algorithms such as BASIS. As the algorithm explores the parameter-space, the parameter variance will cause a wide variety of model evaluation times. This problem occurs during the inference of the posterior distribution of the model parameter γ_C described in Section 2.

3.2 Imbalance in the RBC Model

In this work, we infer the posterior distribution of the membrane dissipation parameter γ_C of the RBC and the posterior of the hyperparameters in the hierarchical Bayesian setting. A preliminary analysis shows that the constraints described in Appendix B imply that the time step of the numerical time integration scheme decreases linearly with γ_C. As a consequence, increasing values of γ_C result in an increased number of simulation steps, and thus higher execution times.

After running BASIS on all data sets, we analyzed the individual model evaluations and we found a linear relationship between runtimes and the model parameter γ_C. This behavior aligns with our preliminary analysis and is responsible for most of the encountered load imbalance. Based on the linear relationship between model runtime and γ_C, we estimate the expected load imbalance ratio given by

$$\mathcal{L} = \frac{T_{max} - T_{avg}}{T_{avg}} , \tag{1}$$

where T_{max} is the maximum runtime, and T_{avg} is the average runtime of the sample evaluations. Assuming that we sample the membrane dissipation parameter $\gamma_C \sim \mathcal{U}(8000, 32000)$ (expressed hereafter in units of U_M/U_T, see Appendix B for details) from a uniform distribution we can estimate the load imbalance ratio as

$$\mathcal{L} = \frac{T(\gamma_{C\,max}) - T(\gamma_{C\,avg})}{T(\gamma_{C\,avg})} \approx 0.44 , \tag{2}$$

with $T(\gamma_{C\,avg}) = 1.16$h and expected maximal runtime $T(\gamma_{C\,max}) = 1.67$h as $\mathbb{E}[\gamma_C] = 20000$ and $\mathbb{E}[\max \gamma_C] \approx 32000$ (in simulation units). Based upon this calculation, we can express the inefficiency, in terms of effective node usage, by

$$\mathcal{E} = \frac{1}{1 + \mathcal{L}} = 69\% . \tag{3}$$

This estimation shows that computational resources will remain idle 31% of the time. Note that this is a conservative estimate since we expect the sampling to converge to the mass of the posterior distribution at later generations of BASIS and hence the variance within the sample population will decrease.

3.3 Preliminary runs

We performed a preliminary run of the proposed setup to verify that the load imbalance estimated in Section 3.2 is reflected in the experiments runtime. The results from this case will constitute a baseline for our improved method presented in Section 4.

3.3.1 Testbed. We performed all simulations on *Piz Daint*, a Cray supercomputer [26] located at the *Swiss National Supercomputing Centre* (CSCS) [27]. We used the GPU-based XC50 partition, comprising 5'704 compute nodes, each equipped with a single processor socket populated with a 2.6GHz 12-core Intel Xeon E5-2690v3 "Haswell" processor and 64GB of DRAM. Each node also contains an NVIDIA "Tesla" P100 GPU with 16GB of device memory.

The RBC model is implemented in Mirheo, a high-performance software for microfluidic simulations that outperforms current state-of-the-art packages [28]. Mirheo is written in C++/CUDA and targets NVIDIA GPUs with compute capability 3.5 and greater. We base our choice on the fact that Mirheo is optimized for the NVIDIA-Pascal architecture and is specifically tailored for the *Piz Daint* supercomputer platform. Mirheo has shown exceptionally low time to solution on several benchmark problems [28]. For our sampling engine, we use Korali [29], a high-performance framework for Bayesian uncertainty quantification and optimization of engineering models that allows running computational models on large-scale jobs.

3.3.2 Observations. As a preliminary run, we executed five BASIS sampling experiments, one per data set, using 512 nodes. These experiments use Korali as our sampling engine with 512 samples per generation. By running these experiments sequentially, using a 1:1 node per sample ratio, we maximize the experimental-wise parallelism, but also expose all possible load imbalance as shown in Section 3.2.

Fig. 3 shows the results of our preliminary run. The run took 48.1h[1] to complete on 512 nodes, requiring a total of 24.6k node hours. The figure shows the impact of load imbalance on our experiments. We observed a sustained efficiency of $\mathcal{E} = 73\%$ throughout the run, meaning that the nodes waste 27% of their running time. The bottom line in Fig. 6 shows the time evolution of efficiency $\mathcal{E}$ of the sequential execution.

This observation agrees with our conservative estimate of 30% idle time presented in Section 3.2. Of the 24.6k node hours consumed, only 18.1k node hours were effectively used for computation. To prevent this problem in future studies, we implemented a more effective way of work scheduling in Korali, as presented in the next section.

4 IMPROVING WORKLOAD DISTRIBUTION

To improve the distribution of work among nodes, we resorted to *processor oversubscription.* Oversubscription is a technique that has been proposed to improve load balancing and reduce the cost of communication in the context of parallel algorithms [30–32].

Oversubscription means to subdivide the required work into more subtasks than available processors. In this way, every processor must compute on average more than one task. At any given point, if a particular processor is overloaded, one of its tasks can be bestowed to an underloaded processor. In the context of sampling, each model evaluation (sample) within a given generation can be considered as a single indivisible task. Therefore, computing as many tasks as available nodes (512, in our case) represents the case where no oversubscription is applied.

The load imbalance that occurred during our preliminary run were introduced due to *sequential scheduling*. In this approach, the five experiments were executed independently, one after the other. This kind of scheduling gives no possibility for oversubscription, as only one sample is available to each node at a given time (see Fig. 3).

In Table 1, we observe that by scheduling experiments sequentially, we wasted 6398 node hours, out of the total 24,605 allocated hours for the job, yielding an efficiency of only 73.9%. The experiment ran for 48.05 hours and incurred an energy usage of 10.45GJ.

4.1 Alternative Scheduling Methods

The reduced efficiency of the sequential scheduling approach motivated us to find alternative methods to improve efficiency in these types of experiments. We used the metadata from Korali to obtain the runtimes for each sample and thus be able to simulate the behavior of other scheduling methods given the same conditions. We analyze two scheduling methods which we denote as *partial* and *dynamic* scheduling.

4.1.1 Partial Scheduling. One way to achieve oversubscription for population and generation based sampling methods is to reduce the number of nodes available to each experiment. When using a fixed node count, oversubscription is achieved by allowing each experiment to execute on a subset of nodes. The effect is that all five BASIS runs will be executing simultaneously, increasing the

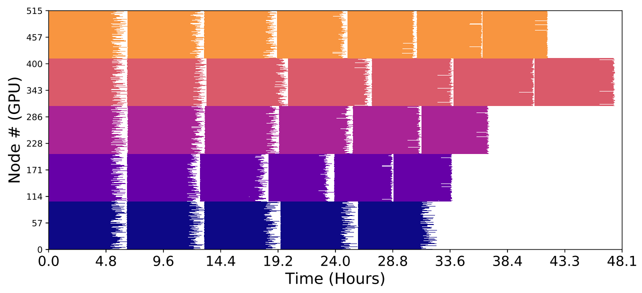

Figure 4: Core usage timeline of partially scheduled BASIS experiments in which each experiment run on its own 103 node subset.

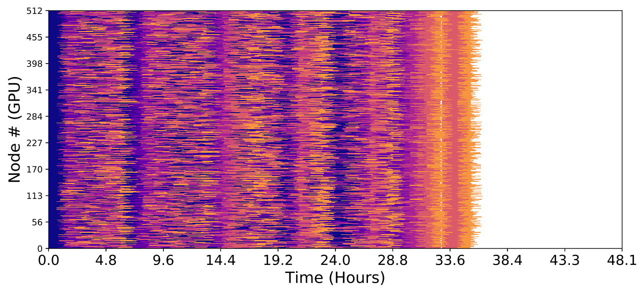

Figure 5: Core usage timeline of inter-experiment scheduled BASIS runs. Samples generated from all experiments can be scheduled for execution on any of the available 512 nodes.

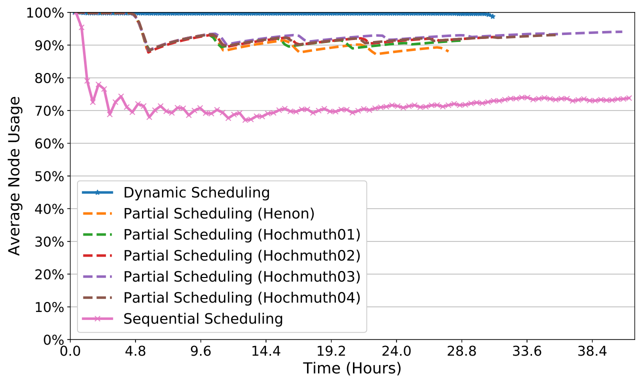

Figure 6: Node usage efficiency across time. A higher percentage indicates better node usage.

overall number of samples per node. For our study, we defined five subsets of nodes of 102.4 nodes each in average[2].

Fig. 4 shows the simulated execution of our partial scheduling experiment. We observed that, thanks to oversubscription, nodes can remain busy for longer before imbalance appears. The results in Table 1, indicate that this strategy resulted in improved efficiency (91.7% in average), wasting much fewer node hours (1628), and requiring less energy (8.42GJ). It did not improve much in terms of running time, since it required 47h to complete all experiments, due

[1]Since *Piz Daint* scheduler is limited to 24h runs, we were required to launch multiple jobs to obtain the final results. Since we used Korali's capability for suspending and resuming jobs on a per-generation basis, sample selection remained unaltered.

[2]In our experiments, we use 103 nodes for all subsets, for a total of 515 nodes to prevent outstanding samples to artificially introduce load imbalance due to a faulty division of work, where an outstanding sample remains for evaluation towards the end. By using an extra node, we avoid this problem and guarantee a fair comparison with our next method, where we use full node/experiment scheduling.

			Node Hours				
Experiment	**Node #**	**Time (h)**	**Used (Total)**	**Effective**	**Wasted**	**Efficiency**	**Energy Usage (GJ)**
Partial (Henon)	103	32.53	3350	2923	427	87.2%	1.42
Partial (Hochmuth01)	103	33.83	3484	3178	306	91.2%	1.48
Partial (Hochmuth02)	103	36.93	3803	3508	295	92.2%	1.62
Partial (Hochmuth03)	103	47.47	4889	4589	300	93.8%	2.07
Partial (Hochmuth04)	103	41.83	4308	4008	300	93.0%	1.82
Partial Scheduling (Total):			19836	18207	1628	**91.7%**	8.42
Sequential Scheduling	512	48.05	24605	18207	6398	**73.9%**	10.45
Dynamic Scheduling	512	36.33	18601	18207	394	**97.8%**	7.90

Table 1: Run statistics for the three scheduling strategies described in this paper. The data for sequential scheduling are taken from preliminary run on *Piz Daint*. The values for partial and dynamic scheduling are simulated and show how the experiment would have run with these scheduling strategies. The power consumption estimates are extrapolated from the actual measured power consumption (10.45GJ) on *Piz Daint*.

to sampling the posterior corresponding to data from *Hochmuth03* taking longer than the rest of the experiments.

4.1.2 Dynamic Scheduling. Another way to achieve oversubscription is to run all experiments simultaneously such that all samples can be scheduled among any of the available nodes. This approach achieves oversubscription, not by reducing the number of nodes, but by increasing the number of tasks. In this case, the increase in tasks comes from scheduling multiple independent experiments[3].

In our case, this approach requires that the scheduler implemented in Korali will be able to sample five experiments in parallel and distribute the workload among all 512 nodes. Fig. 5 shows a simulated execution of our dynamic scheduling experiment. We observed that almost all nodes can remain busy during the entirety of execution, only to lose efficiency towards the end. This efficiency loss comes from the fact that some experiments finish before others, reducing available oversubscription towards the end.

The results summarized in Table 1 indicate that this strategy yields in a superior efficiency (97.8% on average) than the others, wasting much fewer node hours (394), as well as requiring less energy (7.90GJ). Furthermore, it also reduced the runtime from 48 to 36h until completion.

4.2 Statistical Analysis of the Methods

There are two ways to implement the dynamic scheduling discussed in Section 4.1.2: (i) evaluating all samples from one experiment before moving to the next one (see Fig. 5), or (ii) alternate sample evaluations from the individual BASIS runs in a cyclic way without prioritizing a sampling problem. To find which strategy performs better, we conduct a bootstrap analysis on both dynamic scheduling strategies and calculate the mean, median, and standard deviation of the effective node usage $\mathcal{E}$. Through the bootstrap approach we are able to alleviate the randomness associated in the ordering of the samples. Table 2 summarizes our results, from which we conclude that $\mathcal{E}$ is insensitive with respect to strategies (i) or (ii). For a practical reason we favor the former strategy: Korali stores intermediate results after termination of a generation, we prefer to reduce the time taken between start and end of a generation and hence maximize the recovery rate in the case of an unforeseen interruption.

[3]As opposed to dividing a sample into finer-grained pieces, the way oversubscription is typically attained in the context of parallel algorithms.

Additionally, we analyze the behavior of partial scheduling, discussed in Section 4.1.1, with varying number of nodes. Although reducing the number of nodes $\mathcal{E}$ increases the expected effective node usage, we observe that with $N = 16$ nodes we can achieve the same expected node usage ($\approx$ 98%) as with dynamic scheduling (see Table 2). This results shows that we can reduce the time to solution by a factor of 32 while running with almost no resource waste.

	Dyn. (i)	Dyn. (ii)	Part.	Part.	Part.	Part.
Node #	512	512	16	32	128	512
$\mathbb{E}[\mathcal{E}]$	0.98	0.98	0.98	0.97	0.93	0.863
Median[$\mathcal{E}$]	0.98	0.98	0.98	0.97	0.93	0.863
SD[$\mathcal{E}$]	0.001	0.001	0.001	0.002	0.002	0.003

Table 2: Bootstrap analysis of the effective node usage $\mathcal{E}$ based on 100 simulated BASIS runs with different work distribution strategies (Dynamic (i) & (ii) and Partial Scheduling) and nodes (N).

4.3 Experimental Validation

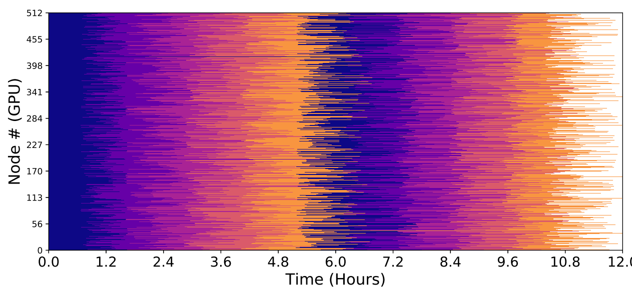

Figure 7: Scheduling under a multiple-experiment configuration on Korali. This 12 hour timeline shows how Korali scheduled all five BASIS experiments simultaneously for a limited two-generation run on 512 *Piz Daint* job.

To validate the dynamic scheduling approach in an actual run on *Piz Daint*, we extended Korali capability to allow for scheduling of independent experiments though one Korali instance. Therefore we re-ran our five BASIS experiments on 512 *Piz Daint* nodes and re-evaluated the behavior of Korali and the achieved load balance. To avoid excessive node hour usage, we limit all five

BASIS experiments to evaluate 2 generations, as opposed to run until completion.

The timeline in Fig. 7 shows that Korali yields a similar interlacing pattern to that from our simulated load balancing strategy. Korali achieves close to 100% efficiency during steady-state, before the tail-end of the experiment.

5 RESULTS OF RBC INFERENCE

5.1 Surrogate model based on BASIS output

In order to improve the accuracy of the sampling algorithm, we use the simulation output of the BASIS samples from all generations to generate a surrogate model for the RBC relaxation. The surrogate is an analytical function following an exponential decay [2], where the decay constant is set through a least-squares fit to the simulation outputs. We use these surrogates to conduct a new series of sampling experiments with an increased sample size (10^4) whilst avoiding the use of computationally additional model evaluations. Using the output of BASIS to construct a surrogate shows to be a reasonable approach, as BASIS samples the parameter space uniformly at start and then converges to the mass of the posterior distribution.

5.2 Sampling results

	maximum a posteriori (MAP)					
Parameter	HO_1	HO_2	HO_3	HO_4	HE	$\mathcal{HB}$
$\gamma_C \times 10^{-3}$	14.676	18.896	23.576	17.912	19.513	24.848
$\sigma_1 \times 10^2$	3.846	6.733	3.822	1.888	-	7.230
$\sigma_2 \times 10^3$	-	-	-	-	3.977	2.389×10^{-4}

Table 3: MAP parameter values of the Bayesian inference for each data set. We denote by MAP_x the MAP parameter values corresponding to the inference performed on data set x, and by $\mathrm{MAP}_{\mathcal{HB}}$ the MAP from the hierarchical inference.

The MAP values for the parameters are summarized in Table 3. We observe that each data set has a different set of MAP values, practically showing that each data set is best fitted by a different set of parameters. These differences can be due to inherent differences in the mechanical parameters of RBCs, or experimental measurement errors causing variations among the data sets.

A general posterior distribution $p(\boldsymbol{\vartheta}^{\text{new}}|\boldsymbol{d})$, obtained from the hierarchical Bayesian inference, is shown in Fig. 8. The marginal posterior distribution for γ_C follows a unimodal distribution, with a MAP value at $\gamma_C = 24.848 \times 10^3$.

This value of γ_C corresponds to a membrane surface viscosity of $\eta_m = 0.63 \times 10^{-6}$ Pa s m, which is within the range estimated in the literature, $\eta_m = [0.05 - 0.8] \times 10^{-6}$ Pa s m [2, 5] (details for transformation between SI and simulation units are given in Appendix B).

To our knowledge this is the first time that the dissipation coefficient has been inferred, for a membrane dissipation model that employs friction only along the lines of centers of the membrane vertices. Previous studies [33] have calibrated the membrane dissipation based on twisting torque cytometry experiments and cross-validated the relaxation timescale with the relaxation data of [2, 3], however using both central and tangential dissipation terms, and thus not conserving angular momentum. Future simulations employing this model can use the MAP value of $\gamma_C = 24.848 \times 10^3$ from the results of the hierachical inference, or use the full posterior distribution to obtain predictions with uncertainty.

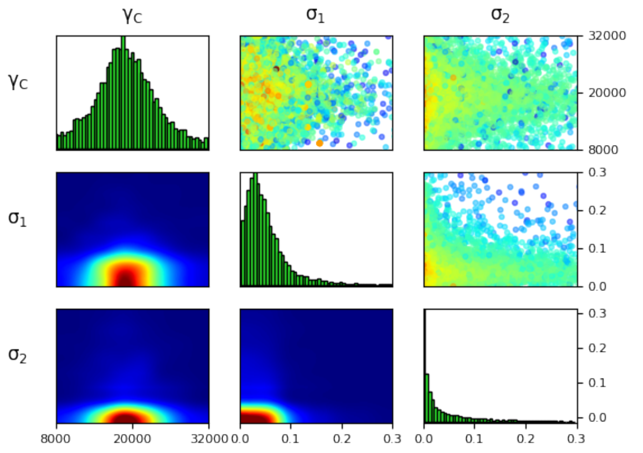

Figure 8: Results of the hierarchical Bayesian inference for $p(\boldsymbol{\vartheta}^{\text{new}}|\boldsymbol{d})$. The diagonals represent the marginal distributions of the parameters, below the diagonal are 2D projections of the posterior distributions $p(\boldsymbol{\vartheta}^{\text{new}}|\boldsymbol{d}, \mathcal{M}_{HB})$, and above the diagonal are samples from the posterior distribution.

6 RELATED WORK

The application of uncertainty quantification methods on large scale computational models and massively parallel computer architectures entails a number of challenges that have scarcely been addressed before. We have found, very few frameworks besides Korali that are suitable for the validation of parameter against experimental data through a large-scale Hierarchical Bayesian analysis as the one presented in this work.

The Π4U [34] offers a framework for the execution of MPI-based parallel applications at a large scale for Bayesian Inference and optimization methods. Π4U employs the tasking library *TORC* [35], which uses *job-stealing* to deal with load imbalance. By enabling idle worker nodes to gather pending samples from another busy worker's queue, it actively responds to the load imbalance originated from an unfair initial distribution of work queues [36]. This source of workload imbalance, however, is not present in Korali since its scheduling engine distributes one sample at a time in real-time, guaranteeing a continuous fair distribution across all workers. Furthermore, *TORC*'s load-balancing method does not address the source of work imbalance discussed in this paper since this imbalance arises from the sampling method rather than the initial distribution of work. Moreover, since Π4U does not offer support for hierarchical inference where many concurrent experiments require sampling, it is unlikely that experiments running under Π4U allow the implementation of the solutions discussed here.

The use of processor oversubscription has been described previously to deal with load imbalance and the cost of communication in parallel algorithms. The *Charm++* programming model [37] and the *AMPI* interface [31] use *virtualization*, where tasks can be migrated among systems by a distributed runtime system, to actively balance the workload of distributed algorithms. The authors of *MATE* [32] describe the use of oversubscription to distribute uneven numbers of tasks among workers to enforce balancing with a priori information [6] on *Cart3D*, a high-fidelity inviscid analysis package for conceptual and aerodynamic design [38]. We will explore the use of integrating a priori information into the scheduling strategy to improve the strategies presented in this paper for cases where there are few available samples to compensate for imbalances in the model.

7 CONCLUSION

State-of-art blood flow simulations have demonstrated the potential of computational studies to contribute in the development of novel diagnostics and treatments for life-threatening diseases, by being able to simulate full microfluidic devices [39] and large microvascular networks [40]. We seek to advance these studies with more detailed and physically precise simulations. In this context, the contributions of our work are, therefore, twofold:

First, the inferred values of RBC membrane dissipation in this work will enable us to perform realistic studies of RBC dynamics, having now a complete model with validated strain [41] and calibrated dissipation models. Armed with this information, we will seek to improve on previously two-dimensional simulations [42], for the optimization of a deformability-based cell sorting device, by correctly capturing the relaxation timescale of RBCs.

Second, parametric investigations, inference and optimization problems may all suffer from load imbalance due to parameter-dependent model runtimes. In order to address this problem, we presented and implemented a method to improve effective node usage of such studies, reducing the overall cost in computational and physical time.

Our next research goals target algorithm dependent scheduling strategies within Korali. The state of running algorithms could be used to prioritize model evaluations from inference/optimization problems that approach the termination criteria slower than others. Through that, early termination of problems can be avoided, maintaining flexibility for oversubscription for a longer time and hence further increase effective node usage. These improvements will enable the efficient simulation of multiple large systems, necessary to transit from simulations of simplified systems, to optimization studies of full microfluidic devices, large vasculature networks, and drug transport in patient-specific vasculatures.

ACKNOWLEDGMENTS

We acknowledge support by the European Research Council (ERC Advanced Grant 341117) and computational resources granted by the Swiss National Supercomputing Center (CSCS) under project ID s929.

Appendices

A STATISTICAL METHODS

A.1 Bayesian Inference

We are interested in estimating the posterior probability distribution of a parameter (or a vector of parameters) conditioned on a set of observations. The *computational parameters* $\boldsymbol{\vartheta}_c \in \mathbb{R}^{N_{\vartheta_c}}$ parametrize a computer simulation with $F(\boldsymbol{x}; \boldsymbol{\vartheta}_c) \in \mathbb{R}$ the simulation output with input variables $\boldsymbol{x} \in \mathbb{R}^{N_x}$.

We observe the vectors $\boldsymbol{d} = (d_1, \dots, d_N)$, e.g., from experiments. We assume that the vector of observations $\boldsymbol{y} \in \mathbb{R}^N$ is a random variable and $\boldsymbol{d}$ is a realization of $\boldsymbol{y}$. In addition, we assume that $\boldsymbol{y}$ is linked to the model through the error equation,

$$\boldsymbol{y} = F(\boldsymbol{x}; \boldsymbol{\vartheta}_c) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \Sigma) , \tag{4}$$

where $\Sigma = \Sigma(\boldsymbol{\vartheta}_s)$ is, in general, a function of the *statistical parameters* $\boldsymbol{\vartheta}_s$. Here, we assume that $\Sigma(\boldsymbol{\vartheta}_s) = \sigma_n^2 \mathbf{I}_N$ with $\boldsymbol{\vartheta}_s = \sigma_n$ and $\mathbf{I}_N$ the identity matrix in $\mathbf{R}^{N \times N}$. The density of $\boldsymbol{y}$ is given by

$$p(\boldsymbol{y} \,|\, \boldsymbol{\vartheta}) = \mathcal{N}(\boldsymbol{y} \,|\, \mu(\boldsymbol{\vartheta}_c), \Sigma(\boldsymbol{\vartheta}_s)) , \tag{5}$$

with $\mu(\boldsymbol{\vartheta}_c) = \big(F(\boldsymbol{x}_1, \boldsymbol{\vartheta}_c), \dots, F(\boldsymbol{x}_N, \boldsymbol{\vartheta}_c)\big)$ and $\boldsymbol{x}_i$ are the different input variables linked to the the observation $\boldsymbol{d}_i$. Here, $\boldsymbol{\vartheta} = (\boldsymbol{\vartheta}_c, \boldsymbol{\vartheta}_s) \in \Omega \subseteq \mathbb{R}^{N_\vartheta}$ is the augmented parameter vector including computational and statistical parameters.

For the estimation of the probability of $\boldsymbol{\vartheta}$ conditioned on the observations $\boldsymbol{d}$ we apply Bayes' theorem,

$$p(\boldsymbol{\vartheta} \,|\, \boldsymbol{d}, \mathcal{M}) = \frac{p(\boldsymbol{d} \,|\, \boldsymbol{\vartheta}, \mathcal{M}) \, p(\boldsymbol{\vartheta} \,|\, \mathcal{M})}{p(\boldsymbol{d} \,|\, \mathcal{M})} , \tag{6}$$

where $p(\boldsymbol{d} \,|\, \boldsymbol{\vartheta}, \mathcal{M})$ is the likelihood function, $p(\boldsymbol{\vartheta} \,|\, \mathcal{M})$ is the prior probability distribution, and $p(\boldsymbol{d} \,|\, \mathcal{M})$ is the model evidence. All other model assumptions are denoted by the variable $\mathcal{M}$. The model evidence can be obtained by marginalizing the likelihood function over the parameters $\boldsymbol{\vartheta}$, i.e.

$$p(\boldsymbol{d} \,|\, \mathcal{M}) = \int_\Omega p(\boldsymbol{d} \,|\, \boldsymbol{\vartheta}, \mathcal{M}) \, \mathrm{d}\boldsymbol{\vartheta} . \tag{7}$$

The Bayesian framework is a useful tool for model comparison in the advent of limited experimental data.

A.2 Hierarchical Bayesian Inference

The hierarchical Bayesian framework is an extension to the single-staged Bayesian framework (Appendix A.1). In a two-staged hierarchical model we first group data in smaller data sets and infer the posterior distributions of the model parameters conditioned on each data set. In the second stage, we combine the posterior distributions and then we infer the hyper-parameters describing the posterior distributions.

For example, each data set corresponds to the observed output of a single experiment from a series of experiments. By inferring the posteriors on an experiment basis, we can capture systematic errors found in the experimental setups. In this setup we denote the collection of data sets as $\vec{\boldsymbol{d}} = \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_{N_d}\}$, where $\boldsymbol{d}_i = \{\boldsymbol{d}_{i,1}, \dots, \boldsymbol{d}_{i,N_i}\}$ is the observed data of to the i-th experiment and $\boldsymbol{x}_{ij}$ is the input vector corresponding to $\boldsymbol{d}_{ij}$. Finally we can obtain better informed individual parameter probabilities, since information from all data sets can be used in the posterior distribution of the individual $\boldsymbol{\vartheta}_i$. The hierarchical Bayesian framework and the concurrent sampling process employed in this work will be described in the following.

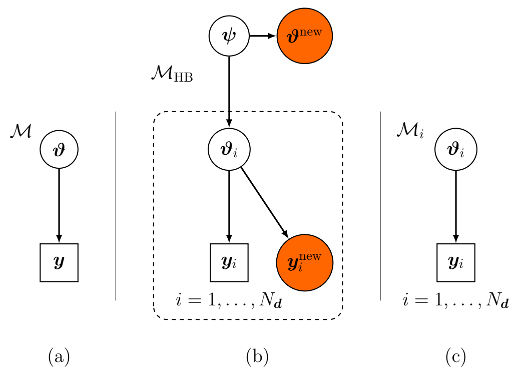

Figure 9: DAGs representing three statistical models: (a) One model parameter $\boldsymbol{\vartheta}$ explaining all observations across all experiments, (b) hierarchical Bayesian model, observations from individual experiments are associated with a parameter $\boldsymbol{\vartheta}_i$, $\boldsymbol{\vartheta}_i$ belong to a common distribution that is described by a vector of hyperparameters $\boldsymbol{\psi}$, (c) individual experiments are described by individual parameter $\boldsymbol{\vartheta}_i$

The joint probability distribution of data $\vec{\boldsymbol{d}}$, model parameters $\vec{\boldsymbol{\vartheta}}$ and hyper-parameters $\boldsymbol{\psi}$ is given by

$$p(\boldsymbol{d}, \vec{\boldsymbol{\vartheta}}, \boldsymbol{\psi}) = \prod_{i=1}^{N} p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i)\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{\psi}) p(\boldsymbol{\psi}) = p(\vec{\boldsymbol{d}} \,|\, \vec{\boldsymbol{\vartheta}})\, p(\vec{\boldsymbol{\vartheta}} \,|\, \boldsymbol{\psi})\,. \quad (8)$$

Our goal is to draw samples from the distribution $p(\vec{\boldsymbol{\vartheta}}, \boldsymbol{\psi} \,|\, \vec{\boldsymbol{d}})$. By applying Bayes' theorem we can write

$$p(\vec{\boldsymbol{\vartheta}}, \boldsymbol{\psi} \,|\, \vec{\boldsymbol{d}}) = \frac{p(\vec{\boldsymbol{d}} \,|\, \vec{\boldsymbol{\vartheta}}, \boldsymbol{\psi})\, p(\vec{\boldsymbol{\vartheta}}, \boldsymbol{\psi})}{p(\vec{\boldsymbol{d}})} = \frac{p(\vec{\boldsymbol{d}} \,|\, \vec{\boldsymbol{\vartheta}})\, p(\vec{\boldsymbol{\vartheta}} \,|\, \boldsymbol{\psi})\, p(\boldsymbol{\psi})}{p(\vec{\boldsymbol{d}})}\,, \quad (9)$$

where we write the likelihood as $p(\vec{\boldsymbol{d}} \,|\, \vec{\boldsymbol{\vartheta}}) = \prod_{i=1}^{N} p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i)$. Sampling the target distribution in Eq. (9) usually becomes inefficient as the number of parameters of the target probability grow with the number of experiments $N_{\boldsymbol{d}}$.

In order to obtain samples from the posterior distribution of the hyperparameters we have to proceed as follows,

$$p(\boldsymbol{\psi} \,|\, \vec{\boldsymbol{d}}) = \frac{p(\vec{\boldsymbol{d}} \,|\, \boldsymbol{\psi})\, p(\boldsymbol{\psi})}{p(\vec{\boldsymbol{d}})}\,, \quad (10)$$

where

$$\begin{aligned}
p(\vec{\boldsymbol{d}} \,|\, \boldsymbol{\psi}) &= \prod_{i=1}^{N} p(\boldsymbol{d}_i \,|\, \boldsymbol{\psi}) \\
&= \prod_{i=1}^{N} \int p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i)\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{\psi})\, \mathrm{d}\boldsymbol{\vartheta}_i \\
&= \prod_{i=1}^{N} \int \frac{p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i)\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{\psi})}{p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{d}_i, \mathcal{M}_i)}\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{d}_i, \mathcal{M}_i)\, \mathrm{d}\boldsymbol{\vartheta}_i \qquad (11) \\
&= \prod_{i=1}^{N} \int \frac{p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i)\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{\psi})\, p(\boldsymbol{d}_i \,|\, \mathcal{M}_i)}{p(\boldsymbol{d}_i \,|\, \boldsymbol{\vartheta}_i, \mathcal{M}_i)\, p(\boldsymbol{\vartheta}_i \,|\, \mathcal{M}_i)}\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{d}_i, \mathcal{M}_i)\, \mathrm{d}\boldsymbol{\vartheta}_i \\
&= \prod_{i=1}^{N} \int \frac{p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{\psi})\, p(\boldsymbol{d}_i \,|\, \mathcal{M}_i)}{p(\boldsymbol{\vartheta}_i \,|\, \mathcal{M}_i)}\, p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{d}_i, \mathcal{M}_i)\, \mathrm{d}\boldsymbol{\vartheta}_i\,,
\end{aligned}$$

which can be approximated by

$$p(\vec{\boldsymbol{d}} \,|\, \boldsymbol{\psi}) \approx \prod_{i=1}^{N} \frac{p(\boldsymbol{d}_i \,|\, \mathcal{M}_i)}{N_s} \sum_{k=1}^{N_s} \frac{p(\boldsymbol{\vartheta}_i^{(k)} \,|\, \boldsymbol{\psi})}{p(\boldsymbol{\vartheta}_i^{(k)} \,|\, \mathcal{M}_i)}\,, \quad (12)$$

using samples drawn from the posterior distributions of the model parameter for each of the experiments $\boldsymbol{\vartheta}_i^{(k)} \sim p_{\boldsymbol{\vartheta}_i}(\cdot \,|\, \boldsymbol{d}_i, \mathcal{M}_i)$.

B INFERENCE OF RBC MEMBRANE DISSIPATION

B.1 Experimental data

The experimental data used for the Bayesian inference correspond to relaxation of RBCs, after elongation with either micro-pipettes or optical tweezers. We consider five relaxation data sets, four obtained from the studies of Hochmuth et al. [2] and one from Henon et al. [3], describing the length over width (L/W) of a RBC and the length (in pixels) over time respectively. We transform the measurements of Henon to L/L_0 such that they can be directly compared to the simulations.

B.2 Computational model

We model one RBC as a visco-elastic membrane enclosing a viscous solvent (hemoglobin). The membrane is discretized into N_e edges and N_v vertices, with positions $\mathbf{r}_i$ and velocities $\mathbf{v}_i$, $i = 1, 2, \ldots, N_v$. The connectivity is fixed over time, and the particle positions and velocities evolve in time according to Newton's law of motions.

The incompressibility of the lipid bilayer and the hemoglobin is modeled through area and volume energy constraint terms[33],

$$E^{co} = k_A \frac{(A - A^{eq})^2}{2A^{eq}} + k_V \frac{(V - V^{eq})^2}{2V^{eq}}, \quad (13)$$

where k_A and k_V are the computational area and volume moduli and A^{eq} and V^{eq} are the area and volume of the RBC at equilibrium, respectively. The elastic forces are derived from the bending energy potential E^{be} and shear energy potential E^{sh}. We employ the discretization for the bending energy as [10, 43]:

$$E^{be} = 2\kappa_b \sum_{i=1}^{N_v} \left(\frac{M_i}{A_i} - H_0 \right)^2 A_i + \frac{\alpha \kappa_b \pi}{2AD^2} \left(2D \sum_{i=1}^{N_v} M_i - \Delta A_0 \right)^2 , \tag{14}$$

where κ_b and $\alpha\kappa_b$ are local and non-local bending muduli, and H_0 and ΔA_0 are their reference states, respectively. Furthermore, D is half the lipid-bilayer thickness and A_i is the Barycentric area associated to vertex i and

$$M_i = \frac{1}{4} \sum_{\langle i,j \rangle} l_{ij} \theta_{ij}. \tag{15}$$

The above summation goes over the direct neighbors of vertex i, $l_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the length of the edge and θ_{ij} is the angle between the normals of the two triangles adjacent to the edge ij. We employ the strain energy proposed in Lim et al. [8, 41],

$$E^{sh} = \frac{k_a}{2} \sum_{i=1}^{N_t} A_i^{sf} \left(\alpha_i^2 + a_3 \alpha_i^3 + a_4 \alpha_i^4 \right) + \mu \sum_{i=1}^{N_t} A_i^{sf} \left(\beta_i + b_1 \alpha_i \beta_i + b_2 \beta_i^2 \right), \tag{16}$$

where A_i^{sf} is the area of triangle i in the associated stress-free shape, α_i and β_i are the triangle area and shear strain invariants of triangle i with respect to the MS stress-free shape, a_3, a_4, b_1 and b_2 are constants, k_a is the dilatation energy coefficient and μ is the shear energy coefficient. The elastic forces acting on each vertex is thus given by $\mathbf{F}_i^{el} = -\nabla_i E^{el} = -\nabla_i \left(E^{co} + E^{be} + E^{sh} \right)$, where ∇_i is the gradient with respect to $\mathbf{r}_i$.

The membrane dissipation is modeled as proposed in [33], albeit excluding the non-radial component such that the angular momentum is conserved and viscous effects remain in-plane,

$$\mathbf{F}_i^{visc} = -\gamma_C \sum_{\langle i,j \rangle} \mathbf{v}_{ij} \cdot \hat{\mathbf{r}}_{ij} \hat{\mathbf{r}}_{ij}, \tag{17}$$

where γ_C is the friction coefficient, $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$, $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$ and $\hat{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/|\mathbf{r}_{ij}|$. The membrane surface viscosity can then be approximated as $\eta_m \approx \gamma_C \sqrt{3}/4$ [33]. The model includes an additional fluctuation term to satisfy the fluctuation-dissipation theorem,

$$\mathbf{F}_i^{fluct} = \sqrt{2\gamma_C k_B T} \sum_{\langle i,j \rangle} \xi_{ij} \hat{\mathbf{r}}_{ij}, \tag{18}$$

where $k_B T$ is the temperature in energy units and ξ_{ij} are independent Gaussian random variables with zero mean and unit variance, chosen independently for each pair of particles and at each timestep.

The fluids inside and outside of the membrane are both modeled with the DPD method [18, 44], with the dissipative kernel elevated to the power $k < 1$ in order to increase the solvent viscosity. The flow-structure Interactions (FSI) are modeled through viscous friction between the mesh vertices and the suspended DPD particles [33, 45–47]. Furthermore, the solvent particles are bounced back to satisfy no-through and no-slip boundary conditions. The time integration is performed using a modified velocity-verlet algorithm with $\lambda = 1/2$ [44]. The timestep is set through CFL-type conditions for the viscous, sonic and acceleration timescales, following Morris et al. [48].

B.3 Simulation setup

The relevant dimensionless quantities for the RBC relaxation are the characteristic relaxation time $t_c = \eta_m/\mu$ [2, 3], the Föppl von Kármán number FvK $= \mu R^2/k_b$, the relative strength between bending and thermal energies k_b/k_BT, and the ratio between the linear elastic moduli of the strain energy k_α/μ, where η_m denotes the membrane surface viscosity and $R = \left(A/(4\pi)\right)^{1/2}$ denotes the effective size of the RBC. The RBC is immersed in a solution of viscosity η_o, and encloses a liquid with viscosity η_i, with $\lambda = \eta_i/\eta_o$ defining the viscosity ratio. The relaxation experiments [2, 3] are performed at room temperature with $\eta_o \approx 1$mPa s. Experimental studies for the viscosity of hemoglobin at room temperature [49, 50] show that $\eta_i = 10$mPa s. This leads to a viscosity ratio $\lambda = 10$. The RBC mechanical properties are set as defined in Table 2.2 in [41], leading to FvK $= 139.26$, $k_b/k_BT = 48.94$ and $k_\alpha/\mu = 2$. The MS stress-free shape is an oblate-like configuration with a reduced volume $v = 0.95$, generated by a quasi-vesicle minimization process [8, 41].

Despite all the advantages of the DPD model [16, 39] for approximating the Navier-Stokes equations, the simulation of low Reynolds number flows, where viscous effects dominate over inertial, implies the usage of large solvent viscosities, requiring smaller time steps and leading to unrealistically long simulation run-times. A remedy to this problem is the realization (and confirmation from preliminary simulations) that the exact values of all viscosities do not affect the relaxation of the RBC, as long as the dimensionless ratios mentioned above retain their values, and the Reynolds number is kept well below 1. Therefore, a common practice is to scale the viscosities (η_o, η_i, η_m) by the same factor $f_{sc} < 1$, allowing for larger simulation time steps and thus reasonable simulation runtimes. We use $f_{sc} = 0.005$. As the physical objective of this study is the calibration of the dissipation model to capture the timescale of the RBC relaxation, it is important that we preserve the timescale in the simulations. We thus scale the elastic and bending moduli $(\mu, k_\alpha, k_b, k_BT)$ by the same factor f_{sc}, such that the values of all aforementioned dimensionless quantities are preserved as well.

The basic mass, length and energy scales in the simulation are set as $m = 1$, $r_c = 1$, $k_BT = 0.02$, where m is the mass of a DPD particle, r_c the DPD cutoff radius and k_BT the thermal energy. Through these, we define the following conversion factors between the physical and simulation $(\_s)$ units: A length unit, $U_L = R/R_s = 8.344 \times 10^{-7}$m, a mass unit, $U_M = U_L^3 \rho/\rho_s = 5.810 \times 10^{-17}$kg, and a time unit, $U_T = (U_L^2 U_M (k_BT_s)/(k_BT))^{1/2} = 1.989 \times 10^{-4}$s, where the density of water at room temperature is assumed to be $\rho = 1000$kg/m^3, and $\rho_s = mn_d$. We use a number density $n_d = 10$, $R = 4$ and $N_v = 642$ mesh vertices. The total membrane mass is $M_m = \rho_s A_s$, assuming a two-dimensional membrane surface, and the mass of each membrane particle is $m_m = M_m/N_v$. The domain size is set to $L_x = 5.5R$, $L_y = 3.5R$ and $L_z = 1.2R$, based on a preliminary study for the effect of periodic boundary conditions on the relaxation curve of the RBC. Finally, the DPD dissipative kernel power is set to $k = 0.125$ and the conservative DPD parameter to $\alpha_{ij} r_c/k_BT = 50$, a value obtained from our preliminary studies to ensure that the DPD fluid remains in the liquid phase, similarly to other studies [51, 52].

B.4 Statistical Setup

In this work we consider $N_d = 5$ experiments found in the literature [2, 3]. Each experiment contains time series measurements of RBC extensions during relaxations. The computational parameter $\boldsymbol{\vartheta}_{c,i} = \gamma_{C,i}$ for $i = 1, \ldots, 5$. The statistical parameter σ_n in the error equation Eq. (4) is $\sigma_{n,i} = \sigma_{1,i}$ for $i = 1, \ldots, 4$ and $\sigma_{n,i} = \sigma_{2,1}$ for $i = 5$. This choice reflects the fact that the datasets 1-4 come from [2] and dataset 5 from [3]. The data from [2] show the extensions in the form of the ratio of length (L) over width (W), whereas the data from [3] show the extensions in the form of the ratio of length (L) over the initial length (L_0). The two data sources measure the RBC extension in different units, (L/W and L/L_0), therefore in order to be able to write an additive error equation, $\sigma_{1,i}$, $i = 1, \ldots, 4$ must be expressed in units of L/W, and $\sigma_{2,1}$ must be expressed in units of L/L_0.

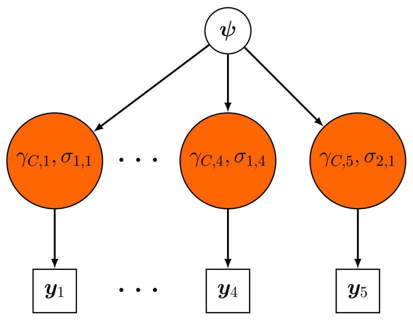
$$\begin{tikzcd}
 & & \psi \arrow[dll] \arrow[d] \arrow[drr] & & \\
\gamma_{C,1}, \sigma_{1,1} \arrow[d] & \cdots & \gamma_{C,4}, \sigma_{1,4} \arrow[d] & & \gamma_{C,5}, \sigma_{2,1} \arrow[d] \\
\boxed{y_1} & \cdots & \boxed{y_4} & & \boxed{y_5}
\end{tikzcd}$$

Figure 10: DAG for the relaxation parameter inference.

For the first step of the hierarchical inference, the posterior distribution of the $p(\boldsymbol{\vartheta}_i \,|\, \boldsymbol{d}_i, \mathcal{M}_i)$ is given by Eq. (6) with priors $p(\gamma_{C,i}) \sim \mathcal{U}(8000, 32000)$ and $p(\sigma_{n,i}) \sim \mathcal{U}(0.0, 1.0)$ for $i = 1, \ldots, 5$. The bounds for $\gamma_{C,i}$ are taken wide enough, such that the RBC model covers all experimental data.

Further, we assume that the relaxation parameter conditioned on the hyperparameters follows a normal distribution: $p(\gamma_{C,i} \,|\, \boldsymbol{\psi}_1, \boldsymbol{\psi}_2) = \mathcal{N}(\gamma_{C,i} \,|\, \boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$. The statistical parameters $\sigma_{n,i}$ follow a gamma distribution: $p(\sigma_{1,i} \,|\, \boldsymbol{\psi}_3, \boldsymbol{\psi}_4) = \Gamma(\sigma_{1,i} \,|\, \boldsymbol{\psi}_3, \boldsymbol{\psi}_4)$ for $i = 1, \ldots, 4$ and $p(\sigma_{2,1} \,|\, \boldsymbol{\psi}_5, \boldsymbol{\psi}_6) = \Gamma(\sigma_{2,1} \,|\, \boldsymbol{\psi}_5, \boldsymbol{\psi}_6)$. For the prior distributions of the hyperparameters we assume $\boldsymbol{\psi}_1 \sim \mathcal{U}(8000, 32000)$, $\boldsymbol{\psi}_2 \sim \mathcal{U}(0, 16000)$, $\boldsymbol{\psi}_3 \sim \mathcal{U}(0, 10)$, $\boldsymbol{\psi}_4 \sim \mathcal{U}(0, 0.5)$, $\boldsymbol{\psi}_5 \sim \mathcal{U}(0, 5)$, and $\boldsymbol{\psi}_6 \sim \mathcal{U}(0, 1)$. The posterior distribution of the hyperparameters is shown in Fig. 11.

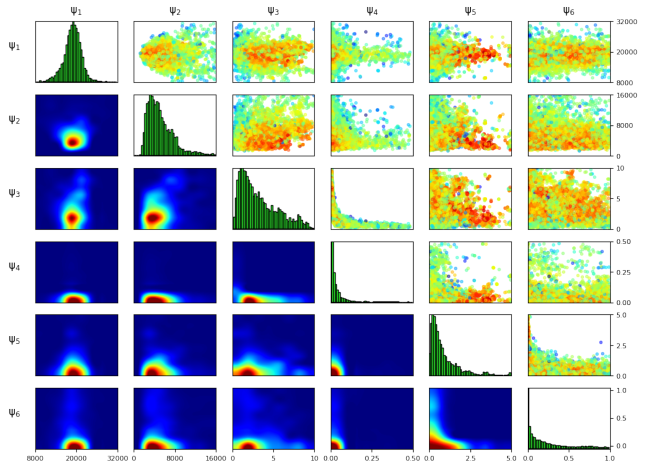

Figure 11: Bayesian inference for the hyperparameters. The diagonal represents the marginal distribution for each inferred quantity, below the diagonal are 2D projections of the posterior distribution and above the diagonal are samples from the posterior distribution, colored by the likelihood (red/blue denote high/low values respectively).

REFERENCES

[1] Steve Ashby, Pete Beckman, Jackie Chen, Phil Colella, Bill Collins, Dona Crawford, Jack Dongarra, Doug Kothe, Rusty Lusk, Paul Messina, et al. The opportunities and challenges of exascale computing. *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pages 1–77, 2010.

[2] Robert M Hochmuth, PR Worthy, and Evan A Evans. Red cell extensional recovery and the determination of membrane viscosity. *Biophysical journal*, 26(1):101–114, 1979.

[3] Sylvie Hénon, Guillaume Lenormand, Alain Richert, and François Gallet. A new determination of the shear modulus of the human erythrocyte membrane using optical tweezers. *Biophysical Journal*, 76(2):1145–1151, 1999.

[4] Hal S. Stern David B. Dunson Aki Vehtari Andrew Gelman, John B. Carlin. *Bayesian Data Analysis*. Taylor and Francis, 3 edition, 2013.

[5] R. Tran-Son-Tay, S. P. Sutera, and P. R. Rao. Determination of red blood cell membrane viscosity from rheoscopic observations of tank-treading motion. *Biophysical Journal*, 46(1):65–72, 1984.

[6] Sergio M. Martin and Scott B. Baden. Mate, a unified model for communication-tolerant scientific applications. In Mary Hall and Hari Sundar, editors, *Languages and Compilers for Parallel Computing*, pages 120–137, Cham, 2019. Springer International Publishing.

[7] C. G. Caro, T. J. Pedley, R. C. Schroter, W. A. Seed, and K. H. Parker. *The mechanics of the circulation*. Cambridge University Press, Cambridge, second edition, 2011.

[8] Gerald H. W. Lim, Michael Wortis, and Ranjan Mukhopadhyay. Stomatocyte–discocyte–echinocyte sequence of the human red blood cell: Evidence for the bilayer– couple hypothesis from membrane mechanics. *PNAS*, 99(26):16766–16769, 2002.

[9] M. Bessis. *Living blood cells and their ultrastructure*. Springer Berlin Heidelberg, 1973.

[10] Xin Bian, Sergey Litvinov, and Petros Koumoutsakos. Bending models of lipid bilayer membranes: Spontaneous curvature and area-difference elasticity. *Comput. Methods Appl. Mech. Engrg.*, 359(112758), 2020.

[11] R. M. Hochmuth and R. E. Waugh. Erythrocyte membrane elasticity and viscosity. *Ann. Rev. Physiol.*, 49:209–219, 1987.

[12] U. Seifert and S. A. Langer. Viscous modes of fluid bilayer membranes. *EPL (Europhysics Letters)*, 23(1):71, 1993.

[13] M Dao, C T Lim, and S Suresh. Mechanics of the human red blood cell deformed by optical tweezers. *Journal of the Mechanics and Physics of Solids*, 51:2259–2280, 2003.

[14] Thomas Klöppel and Wolfgang A. Wall. A novel two-layer, coupled finite element approach for modeling the nonlinear elastic and viscoelastic behavior of human erythrocytes. *Biomechanics and Modeling in Mechanobiology*, 10(4):445–459, 2011.

[15] Zhangli Peng, Adel Mashayekh, and Qiang Zhu. Erythrocyte responses in low-shear-rate flows: Effects of non-biconcave stress-free state in the cytoskeleton. *Journal of Fluid Mechanics*, 742:96–118, 2014.

[16] Jonathan B. Freund. Numerical Simulation of Flowing Blood Cells. *Annual Review of Fluid Mechanics*, 46(1):67–95, 2014.

[17] Thomas M. Fischer and Rafal Korzeniewski. Threshold shear stress for the transition between tumbling and tank-treading of red blood cells in shear flow: dependence on the viscosity of the suspending medium. *Journal of Fluid Mechanics*, 736:351–365, dec 2013.

[18] P Espanol and P Warren. Statistical Mechanics of Dissipative Particle Dynamics. *Europhysics Letters*, 30(May):191–196, 1995.

[19] D. A. Fedosov, B. Caswell, and G. E. Karniadakis. A multiscale red blood cell model with accurate mechanics, rheology, and dynamics. *Biophys. J.*, 98:2215–2225, 2010.

[20] EA Evans and RM Hochmuth. Membrane viscoelasticity. *Biophysical Journal*, 16(1):1–11, 1976.

[21] S. Wu, P. Angelikopoulos, C. Papadimitriou, and P. Koumoutsakos. Bayesian Annealed Sequential Importance Sampling (BASIS): an unbiased version of Transitional Markov Chain Monte Carlo. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 4(1):011008, 2018.

[22] Stephen Wu, Panagiotis Angelikopoulos, James L. Beck, and Petros Koumoutsakos. Hierarchical Stochastic Model in Bayesian Inference for Engineering Applications: Theoretical Implications and Efficient Approximation. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 5(1):011006, 2019.

[23] Jianye Ching and Yi-chu Chen. Transitional Markov Chain Monte Carlo Method for Bayesian Model Updating, Model Class Selection, and Model Averaging. *Journal of Engineering Mechanics*, 133(7):816–832, 2007.

[24] Robert K Brunner and Laxmikant V Kalé. Handling application-induced load imbalance using parallel objects. *Parallel and Distributed Computing for Symbolic and Irregular Applications*, pages 167–181, 2000.

[25] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of …, 2006.

[26] CSCS piz daint. https://www.cscs.ch/computers/piz-daint/. Accessed: 30-10-2019.

[27] CSCS swiss national supercomputing centre. https://www.cscs.ch/. Accessed: 30-10-2019.

[28] Dmitry Alexeev, Lucas Amoudruz, Sergey Litvinov, and Petros Koumoutsakos. Mirheo: High-performance mesoscale simulations for microfluidics, 2020.

[29] The korali framework. https://www.cse-lab.ethz.ch/korali/.

[30] T. Nguyen, P. Cicotti, E. Bylaska, D. Quinlan, and S. B. Baden. Bamboo – Translating MPI applications to a latency-tolerant, data-driven form. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–11, Nov 2012.

[31] Seonmyeong Bak, Harshitha Menon, Sam White, Matthias Diener, and Laxmikant V. Kalé. Multi-Level Load Balancing with an Integrated Runtime Approach. In *18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2018, Washington, DC, USA, May 1-4, 2018*, pages 31–40, 2018.

[32] Sergio M. Martin. *MATE, a Unified Model for Communication-Tolerant Scientific Applications*. PhD thesis, University of California, San Diego, 2018.

[33] Dmitry A. Fedosov, Bruce Caswell, and George Em Karniadakis. A Multiscale Red Blood Cell Model with Accurate Mechanics, Rheology, and Dynamics. *Biophysical Journal*, 98(10):2215–2225, 2010.

[34] P.E. Hadjidoukas, P. Angelikopoulos, C. Papadimitriou, and P. Koumoutsakos. Π4U: A high performance computing framework for Bayesian uncertainty quantification of complex models. *Journal of Computational Physics*, 284:1–21, 2015.

[35] Panagiotis E. Hadjidoukas, Evaggelos Lappas, and Vassilios V. Dimakopoulos. A runtime library for platform-independent task parallelism. *Proceedings - 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2012*, pages 229–236, 2012.

[36] Panagiotis E. Hadjidoukas, Panagiotis Angelikopoulos, Lina Kulakova, Costas Papadimitriou, and Petros Koumoutsakos. Exploiting task-based parallelism in bayesian uncertainty quantification. In Jesper Larsson Träff, Sascha Hunold, and Francesco Versaci, editors, *Euro-Par 2015: Parallel Processing*, pages 532–544, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[37] Laxmikant V. Kale and Sanjeev Krishnan. CHARM++: A Portable Concurrent Object Oriented System Based on C++. In *Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '93, pages 91–108, New York, NY, USA, 1993. ACM.

[38] M. Aftosmis, M. Berger, and G. Adomavicius. A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. In *AIAA'00*, 2000.

[39] Diego Rossinelli, George Karniadakis, Massimiliano Fatica, Igor Pivkin, Petros Koumoutsakos, Yu-hang Tang, Kirill Lykov, Dmitry Alexeev, Massimo Bernaschi, Panagiotis Hadjidoukas, Mauro Bisson, Wayne Joubert, and Christian Conti. The In-Silico Lab-on-a-Chip: Petascale and High-Throughput Simulations of Microfluidics at Cell Resolution. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15*, pages 1–12, 2015.

[40] Peter Balogh and Prosenjit Bagchi. Analysis of red blood cell partitioning at bifurcations in simulated microvascular networks. *Physics of Fluids*, 30(5), 2018.

[41] Gerald Lim H. W., Michael Wortis, and Ranjan Mukhopadhyay. Red Blood Cell Shapes and Shape Transformations: Newtonian Mechanics of a Composite Membrane. *Soft Matter: Lipid Bilayers and Red Blood Cells*, 4:83–254, 2008.

[42] Gokberk Kabacaoglu and George Biros. Optimal design of deterministic lateral displacement device for viscosity-contrast-based cell sorting. *Physical Review Fluids*, 3(12):124201, 2018.

[43] Frank Jülicher. The Morphology of Vesicles of Higher Topological Genus: Conformal Degeneracy and Conformal Modes. *Journal de Physique II*, 6(12):1797–1824, 1996.

[44] Robert D. Groot and Patrick B. Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *The Journal of Chemical Physics*, 107(11):4423, 1997.

[45] Dmitry A. Fedosov, Matti Peltomäki, and Gerhard Gompper. Deformation and dynamics of red blood cells in flow through cylindrical microchannels. *Soft Matter*, 10(24):4258–4267, 2014.

[46] Luca Lanotte, Johannes Mauer, Simon Mendez, Dmitry A Fedosov, Jean-Marc Fromental, Viviana Claveria, Franck Nicoud, Gerhard Gompper, and Manouk Abkarian. Red cells' dynamic morphologies govern blood shear thinning under microcirculatory flow conditions. *Proceedings of the National Academy of Sciences*, 113(47):13289–13294, 2016.

[47] Johannes Mauer, Simon Mendez, Luca Lanotte, Franck Nicoud, Manouk Abkarian, Gerhard Gompper, and Dmitry A. Fedosov. Flow-Induced Transitions of Red Blood Cell Shapes under Shear. *Phys. Rev. Lett.*, 121(11):118103–1–6, 2018.

[48] Joseph P Morris, Patrick J Fox, and Yi Zhu. Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, 136:214–226, 1997.

[49] R. M. Hochmuth, K. L. Buxbaum, and E. A. Evans. Temperature dependence of the viscoelastic recovery of red cell membrane. *Biophysical Journal*, 29(1):177–182, 1980.

[50] Giles R. Cokelet and Herbert J. Meiselman. Rheological Comparison of Hemoglobin Solutions and Erythrocyte Suspensions. *Science*, 162(3850):275–277, 1968.

[51] Jae Mo Kim and Ronald J. Phillips. Dissipative particle dynamics simulation of flow around spheres and cylinders at finite Reynolds numbers. *Chemical Engineering Science*, 59(20):4155–4168, 2004.

[52] Zunmin Zhang, Ewan Henry, Gerhard Gompper, and Dmitry A. Fedosov. Behavior of rigid and deformable particles in deterministic lateral displacement devices with different post shapes. *Journal of Chemical Physics*, 143(24), 2015.