Massively Parallel Likelihood-Free Parameter Inference for Biological Multi-Scale Systems

E. Alamoudi, J. Starruß, N. Bundgaard, R. Müller, F. Reck, F. Graw, L. Brusch, J. Hasenauer, Y. Schälte

published in

NIC Symposium 2022
M. Müller, Ch. Peter, A. Trautmann (Editors)

Forschungszentrum Jülich GmbH,
John von Neumann Institute for Computing (NIC),
Schriften des Forschungszentrums Jülich, NIC Series, Vol. 51,
ISBN 978-3-95806-646-5, pp. 355.
http://hdl.handle.net/2128/31840

Massively Parallel Likelihood-Free Parameter Inference for Biological Multi-Scale Systems

Emad Alamoudi[1], Jörn Starruß[2], Nils Bundgaard[3], Robert Müller[2], Felipe Reck[1], Frederik Graw[3,4], Lutz Brusch[2], Jan Hasenauer[1,5,6], and Yannik Schälte[1,5,6]

[1] Faculty of Mathematics and Natural Sciences, Rheinische Friedrich-Wilhelms-Universität Bonn, 53115 Bonn, Germany
E-mail: {emad.alamoudi, jan.hasenauer, yannik.schaelte}@uni-bonn.de

[2] Center of Information Services and High Performance Computing (ZIH), Technische Universität Dresden, 01062 Dresden, Germany

[3] BioQuant - Center for Quantitative Biology, Heidelberg University, 69120 Heidelberg, Germany

[4] Interdisciplinary Center for Scientific Computing, Heidelberg University, 69120 Heidelberg, Germany

[5] Institute of Computational Biology, Helmholtz Zentrum München, 85764 Neuherberg, Germany

[6] Center for Mathematics, Technische Universität München, 85748 Garching, Germany

Biological processes such as virus spread and tumour growth are highly complex. In order to accurately describe and understand such systems, increasingly complex models are being developed. In particular, multi-scale multi-cellular models are used to describe organisation and development of cell populations and tissues. Typically, such models depend on unknown parameters that need to be calibrated on experimental data, in order to test hypotheses and make predictions. Approximate Bayesian Computation (ABC) is a likelihood-free parameter inference method applicable to a wide range of models, where other methods fail. However, it is computationally demanding, such that computation time is often the limiting factor, especially for expensive multi-scale models. In this article, we demonstrate how the use of high-performance computing (HPC) infrastructure together with efficient parallelisation strategies enables ABC inference on challenging application problems. Further, we demonstrate how HPC enables the development and study of novel algorithms.

1 Introduction

Quantitative mechanistic models are important tools in many research areas to describe, study, and understand real-world systems[1,2]. Commonly, these models depend on unknown parameters. In order to test hypotheses, make predictions, or compare models, these parameters need to be calibrated on experimental data[3]. The Bayesian framework allows doing so by combining prior beliefs about parameters with the likelihood of data given parameters. However, especially for complex mechanistic models, evaluating the likelihood quickly becomes computationally infeasible[4,5]. Approximate Bayesian computation (ABC) is a widely applicable parameter estimation method that, in a nutshell, circumvents likelihood evaluation by simulating data from the model, and accepting corresponding parameters if a distance between simulated and observed data is sufficiently small[6].

A particular model type of interest are multi-scale multi-cellular models, which are used to describe interacting processes on a cellular or tissue level, such as viral infection

or tumour development. Such models provide a realistic description and are often easy to formulate as a computer program, taking an agent-based perspective. However, they can be highly complex, often combining different sub-model types to describe different sub-processes, and can be computationally expensive[4,7,8].

In this article, we first briefly summarise the underlying methodology. Then, we outline concrete implementations of the various algorithms via software packages developed by the authors, and outline their deployment and parallelisation on HPC infrastructure. Then, we give concrete biological applications demonstrating how bringing ABC methods to HPC infrastructure allows to tackle new problems in the life sciences, and further how it allows to develop and test novel algorithms, specifically on outlier-robust adaptive distances, and the use of inverse machine learning models to construct summary statistics in ABC.

2 Sequential Approximate Bayesian Computation

Consider a mechanistic model specified via its likelihood $\pi(y|\theta)$ of observing data $y \in \mathbb{R}^{n_y}$ given model parameters $\theta \in \mathbb{R}^{n_\theta}$. The parameters are assumed unknown and can comprise e.g. reaction rates or initial concentrations. Assuming we have observed experimental data $y_{\text{obs}} \in \mathbb{R}^{n_y}$, the goal of Bayesian inference is to obtain information about the posterior distribution

$$\pi(\theta|y_{\text{obs}}) \propto \pi(y_{\text{obs}}|\theta)\pi(\theta),$$

where $\pi(\theta)$ encodes beliefs prior to observing the data. Unlike many other methods, ABC is still applicable when evaluating the likelihood is computationally infeasible, and only requires that the model is generative, i.e. one can generate data $y \sim \pi(y|\theta)$ given parameters $\theta \sim \pi(\theta)$. This amounts to assuming that the model exists as an executable, potentially stochastic, forward simulation program. Then, at its core, ABC consists of three steps (Fig. 1)[10,5]: First sample parameters $\theta \sim \pi(\theta)$, second simulate data $y \sim \pi(y|\theta)$, and third accept (θ, y) if $d(y, y_{\text{obs}}) \leq \varepsilon$. Here, d is a distance measure comparing simulated and observed data, and $\varepsilon \geq 0$ is an acceptance threshold. This is repeated until sufficiently many, say N, particles have been accepted. Denoting $\pi_{\text{ABC},\varepsilon}(\theta|y_{\text{obs}})$ the distribution of

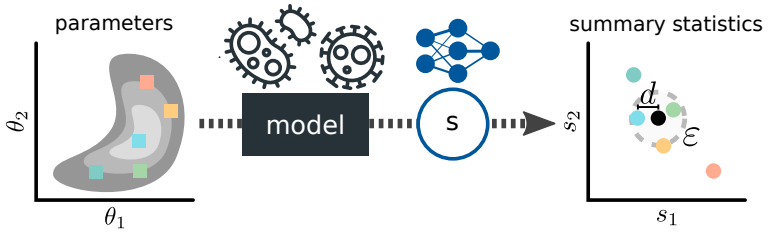

Figure 1. **Basic ABC algorithm.** Parameters $\theta \sim \pi(\theta)$ are sampled from the prior or a proposal distribution, and passed to a black-box model generating potentially stochastic simulated data according to the likelihood $y \sim \pi(y|\theta)$. These are optionally passed through a summary statistics function giving a low-dimensional representation $s(y)$. Summary statistics of simulated and observed data are compared via a distance metric d, and the underlying parameters accepted if the distance is below an acceptance threshold ε. This figure is taken from Schälte *et al.*, 2022[9].

accepted particles, it can be shown under mild conditions that $\pi_{\mathrm{ABC},\varepsilon}(\theta|y_{\mathrm{obs}}) \to \pi(\theta|y_{\mathrm{obs}})$ for $\varepsilon \to 0$ in an appropriate sense[11]. To tackle the curse of dimensionality, the comparison of simulations and data is often in terms of low-dimensional summary statistics $s(y)$ in place of y.

ABC is frequently combined with a sequential Monte Carlo (SMC) scheme[12,13]. In ABC-SMC, a series of particle populations $P_t = \{(\theta_i^t, y_i^t, w_i^t)\}_{i \leq N}$, $t = 1, \ldots, n_t$, are generated, with thresholds $\varepsilon_1 > \ldots > \varepsilon_{n_t}$. In generation t, importance sampling based on generation $t-1$ is used, with Radon-Nikodym derivatives w^t. It successively reduces the acceptance threshold and thus improves the posterior approximation, while maintaining high acceptance rates.

3 Implementation

3.1 Systematic Inference for Multi-Scale Multi-Cellular Models via FitMultiCell

We developed a scalable, modular ABC-SMC framework in the open-source Python toolbox pyABC [a][9]. Beyond the core ABC-SMC routine sketched above, pyABC provides various algorithms to adapt to the problem structure and automatically tune hyperparameters, such as transitions kernels, acceptance thresholds, and population sizes[14,15]. pyABC can be used with models written in arbitrary languages, with tight support of Python, R, and Julia. See Klinger *et al.*, 2018, and Schälte *et al.*, 2022, for details[16,9].

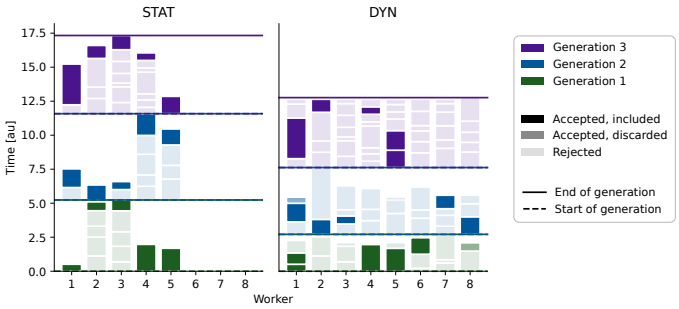

Figure 2. **Static (STAT) and dynamic (DYN) scheduling.** For 3 sequential populations of $N = 5$ particles on $W = 8$ workers. The colour shades from dark to light indicate whether a sample satisfies the acceptance criterion and is included in the population, satisfies the acceptance criterion but is discarded from the population due to late start, or does not satisfy the acceptance criterion and is rejected. For details, see Klinger *et al.*, 2018[16].

The forward model $\pi(y|\theta)$ is to (py)ABC a black-box generating data from parameters. A particular application are multi-scale and multi-cellular models of cell populations and tissues. The open-source C++ toolbox Morpheus [b] is a graphical user interface (GUI) based modelling and simulation environment for such models. It allows to couple various modelling formalisms, including ordinary, partial, and stochastic differential equations,

[a] `https://github.com/icb-dcm/pyabc`
[b] `https://morpheus.gitlab.io`

and cellular Potts models, to describe interactions of cells with each other and intercellular substances. See Starruß *et al.*, 2014, for details[8].

To enable systematic parameter inference for multi-scale multi-cellular models, we connected the tools pyABC and Morpheus in the platform FitMultiCell [c], using a standardised interface of parameter definitions and simulation outputs, as well as a problem formulation based on the PEtab standard[17].

3.2 Parallelisation

Each ABC-SMC population consists of conditionally independent particles, which allows to parallelise the sampling. pyABC currently supports two strategies of parallelisation for both multi-processing and multi-machine distributed execution: Static (STAT) and dynamic (DYN) scheduling (Fig. 2). STAT, which is implemented in many ABC tools, generates only as many sampling tasks and processes as needed, N. However, this does not always use all available cores, especially leaving cores idle while waiting for a few to finish. This is especially a problem on common HPC infrastructure, where resources cannot be easily dynamically re-assigned, leading to inefficient resource usage. In comparison, DYN continues sampling on all available cores until sufficiently many particles have been accepted, afterwards correcting for execution time bias. Thus, it uses all available cores at almost all times, except shortly at the end of generations. DYN has been shown to reduce the overall wall-time substantially over STAT on distributed environments[16]. In all numerical experiments in this work, we used DYN for parallelisation.

3.3 Automated Resource Allocation on HPC Systems

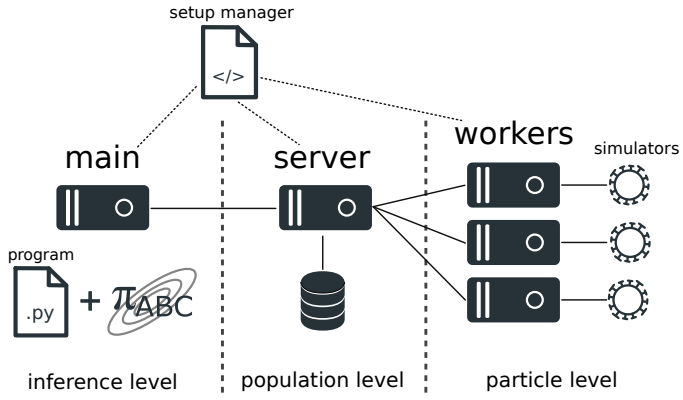

Figure 3. **Distributed ABC-SMC architecture.** The main inference process executes the actual ABC routine via pyABC and an execution program. It transmits tasks to generate particle populations to the central server, from which a number of workers read. The workers then perform the actual model simulations single- or multi-core, e.g. via Morpheus. A central script manages the architecture setup.

[c] e.g. `https://gitlab.com/fitmulticell/fit`

As part of the FitMultiCell effort, we wrote an automatised job management, resource allocation, and execution system for SLURM workload managers, which are in use on many computer clusters. It maintains three types of processes (Fig. 3): a server process that acts as a central broker to distribute work across workers, a number of worker processes that wait for tasks to perform, and an inference process to execute the actual ABC routine. The processes are linked in a hierarchical way uncommon in classical distributed execution applications. The server process runs per default a Redis [d] server that builds a central in-memory database acting as a message broker. Once the server has been launched, its address is dynamically retrieved and passed along with a specified port to all worker processes, enabling their unidirectional communication with the server via Pub/Sub. Simultaneously with starting the workers, the inference process is started, which executes the ABC routine. In each SMC generation, the inference process sends a request to simulate particles to the server, and processes the results once the server has gathered N successful answers from the workers.

Number and resources of worker processes, port, execution time, and target program can be specified via a central shell script [e]. Resources can be shared among the three process types to ensure full exploitation. As (py)ABC is agnostic of model specifics, it is straightforward to replace Morpheus by other simulation programs.

4 Case Studies

4.1 Application Examples

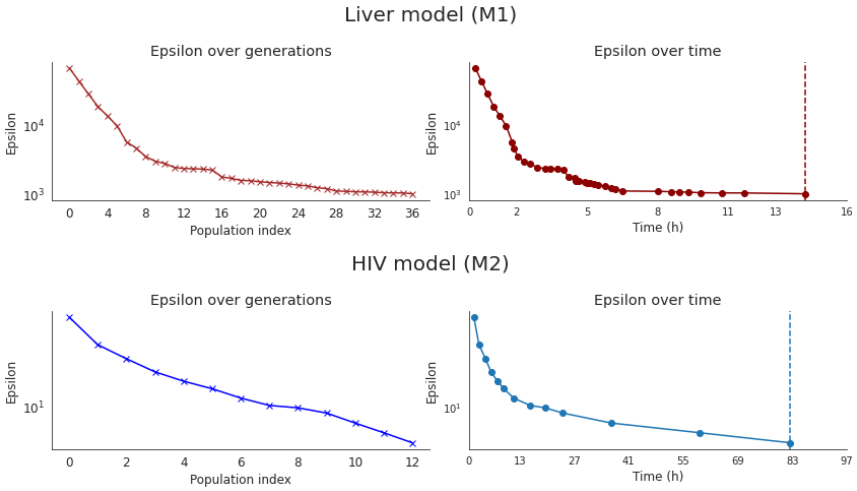

Figure 4. **Acceptance thresholds.** Over generations (left), and over time (right), for models M1+2.

[d] https://redis.io
[e] https://gitlab.com/fitmulticell/Distribution_manager

To assess and validate the FitMultiCell pipeline, we considered two models of different biological processes: Model M1 describes the process of mouse liver regeneration[18]. On different sections of the liver tissue, the mean of yes-associated protein (YAP), which drives the regeneration process, was quantified in 10 zones at different time points. These data were used as observation data for fitting. Model M2 describes the motility of uninfected and HIV-infected lymphocytes within complex tissue environments via a stochastic cellular Potts model (CPM) accounting for biophysical properties of individual cell types to analyse the spread of viruses between cells[19].

The FitMultiCell pipeline was used on model M1 to fit 14 reaction rate parameters. The fit was performed on the JUWELS supercomputer at the Jülich Supercomputing Centre, using up to 84 nodes with 48 cores each, of type 2x Intel Xeon Platinum 8168 CPU, 2.7 GHz. We used $N = 1,000$ particles per generation and $4,032$ workers. The fitting process for this model was run for $n_t = 36$ generations, with a total of $3,803,776$ simulations. The average time for a single forward simulation of this model was about $53.42\,s$, the whole analysis took around $14\,h$ wall-time. Over time, the epsilon acceptance threshold decreased by orders of magnitude and converged gradually (Fig. 4 red). The ABC approximate posterior distribution evolved over the generations, yielding tight posteriors for some of the 14 parameters, while others could not be as precisely inferred (Fig. 5 red, using kernel density estimates).

As an additional application to test the platform, we used model M2, fitting the model's 13 reaction rate parameters. Here, we ran the analysis on the BioQuant Cluster. Each node in the cluster is equipped with IBM xSeries 2 x Quad Intel Xeon E5530 with 20 GB Memory. This demonstrates that the developed platform can be used on different HPC environments. We employed a population size of $N = 256$ using 256 cores for the worker processes. As termination criterion, a minimum discrepancy of $\varepsilon_{\text{min}} = 5$ was used, which

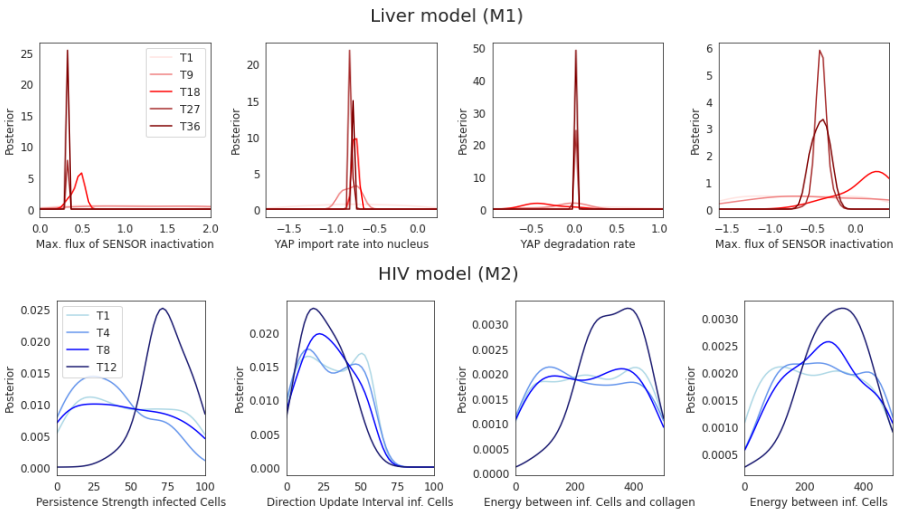

Figure 5. **Parameter fits.** Kernel density estimates in different generations of the iterative fitting process for models M1+2. Shown is only a subset of 4 of the models' 14 resp. 13 parameters each.

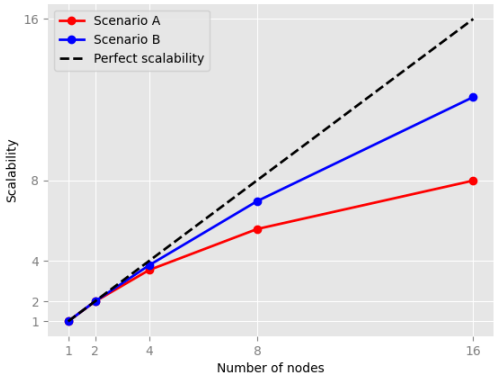

Figure 6. **Scaling behaviour.** Of the FitMultiCell pipeline on JUWELS, for model M3.

was met after $n_t = 12$ generations, resulting in a total of $60,215$ simulations. The simulation of this model is computationally more expensive, taking on average about $1280.25\,s$, the whole analysis required around $83\,h$ wall-time. Similar to before, the epsilon thresholds of different generations, as well as ABC posterior approximations are shown in Fig. 4 and Fig. 5 (blue).

Note that the analyses performed here serve as a first proof-of-concept. They demonstrate that using HPC parallelisation allows us to consider models whose inference would be far too time-consuming otherwise. To assess validity and robustness of obtained posterior approximations, further studies would be necessary.

4.2 Performance Assessment

To assess the speed-up achieved using the massive parallelisation facilitated by the FitMultiCell pipeline, we evaluated parallel scaling on JUWELS standard compute nodes using a model that describes the spread of HCV virus (M3)[20]. We performed the scaling study on two different scenarios of M3, running three sequential populations of size $N = 1,000$ (Scenario A) or $N = 10,000$ (Scenario B) particles. For Scenario A, we observed for 8 nodes (384 cores) a parallel efficiency of 69% and for 16 nodes (768 cores) a parallel efficiency of 49%. For Scenario B, the parallel efficiency was higher, namely 87% and 75% (Fig. 6).

In typical ABC-SMC applications, 15 to 40 sequential populations are run. With each population, usually the acceptance rate decreases, i.e. more simulations have to be performed. This implies an increase of the simulations-to-cores ratio, and in particular the idle time at the end of each population has proportionally less of an impact. This is similar to what can be observed in the scaling study for the larger population size. As we here only ran three sequential populations, we thus expect in an application setting a further improved overall parallel efficiency in later iterations. This indicates that the wall-time of our ABC pipeline scales well, nearly inversely proportionally, with the number of cores.

5 Algorithm Development: Robustness and Informativeness

While the core principle of ABC is simple, its practical performance relies on a number of factors. Two essential problems that we faced are robustness especially to outliers, and assessment of data informativeness.

Outliers, resulting from errors in the data generation process not captured by the model, can severely impact the analysis and lead to biased or uncertain results[22,21]. We showed that popular ABC distance measures can unfortunately be highly sensitive to outliers. This holds especially for a popular distance measure that adjusts in each ABC-SMC generation to the problem structure by weighting different data points or summary statistics based on samples in previous generations, aiming to automatically level the impact of heterogeneous data on the analysis by weights accounting for scale differences[23]. We extended this idea to outlier-corrupted data, by the use of robust norms, and having weights also account for discrepancy of simulations and data in order to identify data points that cannot be explained by the model[21]. This allowed to identify and down-weight outliers, see Schälte *et al.*, 2021, for details[21].

The second problem is that data points are often not homogeneously informative of parameters, e.g. when some only depend on background noise, or different data points are informative of different parts of the model. A popular line of approaches uses inverse regression models, e.g. linear or neural network models, of parameters on data, to extract low-dimensional informative summary statistics[24–26]. We showed first that such approaches benefit on heterogeneous data scales from the combination with scale-normalised distance measures as above[23]. Second, we developed an approach that uses inverse regression models not to define low-dimensional summary statistics, which can possibly lead to a loss of information, but to define additional sensitivity weights quantifying informativeness

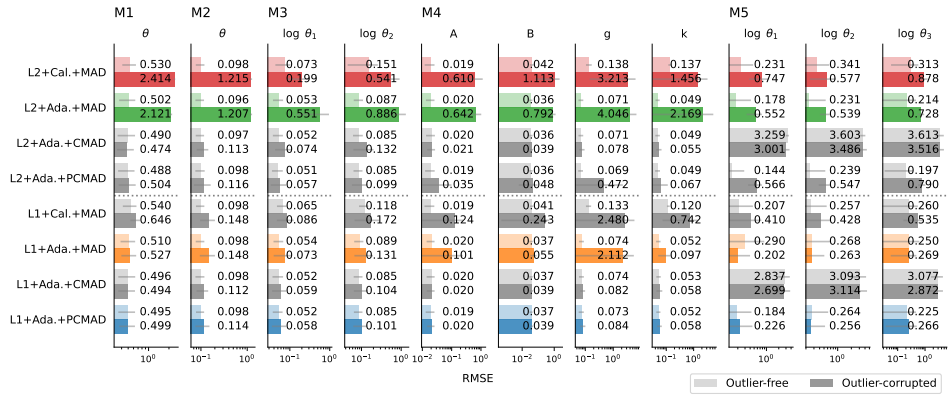

Figure 7. **Distance robustness study.** Mean root mean square errors (RMSE) for the parameters of 5 test models (columns) obtained for 8 distance functions (rows), using L2 or L1 distances, calibrated only in the first ("Cal.") or every ("Ada.") generation, and using MAD, CMAD, and PCMAD for distance weight calculation. Each RMSE is averaged over 20 data sets, grey lines indicate standard deviations. For each distance, the upper, lighter bar is based on outlier-free data, while the lower, darker bar is based on outlier-corrupted data. Distances of interest are coloured, alternative distance combinations are shown in grey for reference. This figure is taken from Schälte *et al.*, 2021, see there for details[21].

of each data point on parameters. Third, we addressed problems of established regression-based approaches with non-identifiability, and presented a solution using regression target augmentation. We observed substantially superior performance of the novel approaches, see Schälte and Hasenauer, 2022, for details[27].

In both studies, we performed extensive method evaluations and comparisons on sets of test and application problems, using up to 20 data replicates (see exemplarily Fig. 7 on robust distances). This was to obtain average and deviation statistics, as results can vary substantially for specific runs and data types. The studies were performed on JUWELS, using up to 384 cores per analysis, and running various analyses in parallel. Given an overall wall-time for all analyses of still several days, a restriction to non-HPC setups would have substantially slowed down method development and robust evaluation.

See Schälte *et al.*, 2021, and Schälte and Hasenauer, 2022, for details on the methods and results[21,27].

6 Conclusion

In this article, we have shown that the use of massive parallelisation enables systematic likelihood-free parameter estimation for biological multi-scale models, and aids the development of novel algorithms. While we have obtained some promising results, the deployment of analysis pipelines on HPC infrastructure could still be improved. For example, existing parallelisation approaches do not make use of all available resources at all times yet, as also dynamic scheduling leaves cores idle at the end of generations, which could e.g. be used to start the next generation already. Further, common HPC infrastructures limit the wall-time of jobs, which can be problematic for longer-running analyses. While pyABC allows to continue previous runs, the amount of manual intervention and information loss could be reduced by automatic checkpointing and job continuation.

We anticipate that approaches as presented here will facilitate the development and analysis of holistic models and gain novel and deeper insights into the organisation of cellular systems.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) (FitMultiCell/031L0159 and EMUNE/031L0293) and the German Research Foundation (DFG) under Germany's Excellence Strategy (EXC 2047 390873048 and EXC 2151 390685813). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (`www.gauss-centre.eu`) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

1. N. A. Gershenfeld and N. Gershenfeld, *The nature of mathematical modeling*, Cambridge university press, 1999.

2. H. Kitano, *Systems Biology: A Brief Overview*, Science, **295**, no. 5560, 1662–1664, Mar. 2002.
3. A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, 2005.
4. N. Jagiella, D. Rickert, F. J. Theis, and J. Hasenauer, *Parallelization and High-Performance Computing Enables Automated Statistical Inference of Multi-scale Models*, Cell Systems, **4**, no. 2, 194–206, 02 2017.
5. S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly, *Inferring coalescence times from DNA sequence data*, Genetics, **145**, no. 2, 505–518, 1997.
6. S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of approximate Bayesian computation*, Chapman and Hall/CRC, 2018.
7. J. Hasenauer, N. Jagiella, S. Hross, and F. J. Theis, *Data-Driven Modelling of Biological Multi-Scale Processes*, Journal of Coupled Systems and Multiscale Dynamics, **3**, no. 2, 101–121, Sept. 2015.
8. J. Starruß, W. de Back, L. Brusch, and A. Deutsch, *Morpheus: A user-friendly modeling environment for multiscale and multicellular systems biology*, Bioinformatics, **30**, no. 9, 1331–1332, Jan. 2014.
9. Y. Schälte, E. Klinger, E. Alamoudi, and J. Hasenauer, *pyABC: Efficient and robust easy-to-use approximate Bayesian computation*, arXiv preprint arXiv:2203.13043, 2022.
10. J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman, *Population growth of human Y chromosomes: a study of Y chromosome microsatellites.*, Molecular biology and evolution, **16**, no. 12, 1791–1798, 1999.
11. S. Barber, J. Voss, and M. Webster, *The rate of convergence for approximate Bayesian computation*, Electronic Journal of Statistics, **9**, no. 1, 80–105, 2015.
12. P. Del Moral, A. Doucet, and A. Jasra, *Sequential Monte Carlo samplers*, J. R. Stat. Soc. B, **68**, no. 3, 411–436, 2006.
13. S. A. Sisson, Y. Fan, and M. M. Tanaka, *Sequential Monte Carlo without likelihoods*, Proc. Natl. Acad. Sci., **104**, no. 6, 1760–1765, Jan. 2007.
14. E. Klinger and J. Hasenauer, *A scheme for adaptive selection of population sizes in Approximate Bayesian Computation - Sequential Monte Carlo*, in: Computational Methods in Systems Biology. CMSB 2017, J. Feret and H. Koeppl, (Eds.), vol. 10545 of *Lecture Notes in Computer Science*, Springer, Cham. 2017.
15. Y. Schälte and J. Hasenauer, *Efficient exact inference for dynamical systems with noisy measurements using sequential approximate Bayesian computation*, Bioinformatics, **36**, no. Supplement 1, i551–i559, 7 2020.
16. E. Klinger, D. Rickert, and J. Hasenauer, *pyABC: distributed, likelihood-free inference*, Bioinformatics, **34**, no. 20, 3591–3593, 10 2018.
17. L. Schmiester, Y. Schälte, F. T. Bergmann, T. Camba, E. Dudkin, J. Egert, F. Fröhlich, L. Fuhrmann, A. L. Hauber, S. Kemmer, P. Lakrisenko, C. Loos, S. Merkt, W. Müller, D. Pathirana, E. Raimúndez, L. Refisch, M. Rosenblatt, P. L. Stapor, P. Städter, D. Wang, F.-G. Wieland, J. R. Banga, J. Timmer, A. F. Villaverde, S. Sahle, C. Kreutz, J. Hasenauer, and D. Weindl, *PEtab – Interoperable specification of parameter estimation problems in systems biology*, PLOS Computational Biology, **17**, no. 1, 1–10, January 2021.

18. K. Meyer, H. Morales-Navarrete, S. Seifert, M. Wilsch-Braeuninger, U. Dahmen, E. M. Tanaka, L. Brusch, Y. Kalaidzidis, and M. Zerial, *Bile canaliculi remodeling activates YAP via the actin cytoskeleton during liver regeneration*, Molecular systems biology, **16**, no. 2, e8985, 2020.
19. A. Imle, P. Kumberger, N. D. Schnellbächer, J. Fehr, P. Carrillo-Bustamante, J. Ales, P. Schmidt, C. Ritter, W. J. Godinez, B. Müller, et al., *Experimental and computational analyses reveal that environmental restrictions shape HIV-1 spread in 3D cultures*, Nature Communications, **10**, no. 1, 2144, 2019.
20. P. Kumberger, K. Durso-Cain, S. Uprichard, H. Dahari, and F. Graw, *Accounting for Space – Quantification of Cell-To-Cell Transmission Kinetics Using Virus Dynamics Models*, Viruses, **10**, no. 4, 200, Apr 2018.
21. Y. Schälte, E. Alamoudi, and J. Hasenauer, *Robust adaptive distance functions for approximate Bayesian inference on outlier-corrupted data*, bioRxiv, 2021.
22. C. Maier, C. Loos, and J. Hasenauer, *Robust parameter estimation for dynamical systems from outlier-corrupted data*, Bioinformatics, **33**, no. 5, 718–725, Mar. 2017.
23. D. Prangle, *Adapting the ABC Distance Function*, Bayesian Analysis, **12**, no. 1, 289–309, 2017.
24. P. Fearnhead and D. Prangle, *Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation*, J. R. Stat. Soc. B, **74**, no. 3, 419–474, 2012.
25. B. Jiang, T.-Y. Wu, C. Zheng, and W. H. Wong, *Learning summary statistic for approximate Bayesian computation via deep neural network*, Statistica Sinica, 1595–1618, 2017.
26. A. Borowska, D. Giurghita, and D. Husmeier, *Gaussian process enhanced semi-automatic approximate Bayesian computation: parameter inference in a stochastic differential equation system for chemotaxis*, Journal of Computational Physics, **429**, 109999, 2021.
27. Y. Schälte and J. Hasenauer, *Informative and adaptive distances and summary statistics in sequential approximate Bayesian computation*, bioRxiv, 2022.