Parallel Performance Analysis of Bacterial Biofilm Simulation Models

Sheraton M.V[1,2] and Peter M. A. Sloot[2,3,4]

[1] HEALTHTECH NTU, Interdisciplinary Graduate School, Nanyang Technological University, Singapore
[2] Complexity Institute, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore
[3] Institute for Advanced Studies, University of Amsterdam, Amsterdam, Netherlands
[4] National Research University ITMO, St. Petersburg, Russia
p.m.a.sloot@uva.nl

Abstract. Modelling and simulation of bacterial biofilms is a computationally expensive process necessitating use of parallel computing. Fluid dynamics and advection-consumption models can be decoupled and solved to handle the fluid-solute-bacterial interactions. Data exchange between the two processes add up to the communication overheads. The heterogenous distribution of bacteria within the simulation domain further leads to non-uniform load distribution in the parallel system. We study the effect of load imbalance and communication overheads on the overall performance of simulation at different stages of biofilm growth. We develop a model to optimize the parallelization procedure for computing the growth dynamics of bacterial biofilms.

Keywords: Load imbalance, communication overhead, biofilm.

1 Introduction

Computational models involving grid based or lattice-based systems are solved in parallel to reduce the overall computation time. In cases of uneven spatial distribution of grids or non-homogenous presence of model objects such as cells, catalysts or solid structures in the domain, the allocation of computational load to the processors may not be uniform. Such discrepancies will result in decrease of parallel computing efficiency. In multiphysics systems comprising of fluid flow, solute diffusion, reaction (or consumption) and cell growth, multiple methods of solving the models need to be implemented. For instance, Finite Element based Method (FEM) [1] or Lattice Boltzmann Method (LBM) [2] can be used to solve fluid dynamic equations, FEM or Finite Volume Method (FVM) [3] to solve the Fick's Equation of diffusion and solute consumption and Agent Based Method (ABM) [4] to handle the cell behavior. When combining these methods, there always exists a communication channel between them. This contributes to communication overhead in parallel computations. In addition, there will be fractional communication overhead [5] within a method resulting from memory access

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

(gathering and scattering) between each processor. Therefore, it is necessary to estimate the communication overhead between the methods, fractional overhead, and the parallel execution durations to optimize the parallel computation process.

In nature, bacteria exhibit two modes of growth, planktonic and biofilm. During their planktonic form of growth, bacteria exist as individual cells that float around in a fluid medium. Due to their direct exposure to ambient environmental conditions, planktonic bacteria are susceptible to antibiotics, bacteriophages, and other chemicals. In contrast, during the biofilm mode of growth, the bacteria adhere to a solid surface and to other bacteria near them, forming a large colony of bacteria confined within a structure known as biofilm. By shielding the bacteria from harsh environmental conditions, biofilms protect them from detrimental external factors and act as a platform for developing antibiotic drug resistance. Therefore, to tackle the health hazards and environmental issues arising from detrimental bacterial biofilms it is necessary to understand the dynamics of biofilm formation. Bacterial biofilm modelling has become an important tool in analyzing and predicting the quorum sensing [6] within the bacterial community, detachment of biofilms [7, 8], and phage-bacteria interactions [9]. Bacterial biofilms are complex systems that require multiphysics based models to effectively describe their evolution process. In most studies [10-12], proliferation of bacteria is modelled by considering the diffusion of essential nutrients such as oxygen or glucose around them. The individual bacterial cells are commonly represented as 'point sinks' or reaction zones within the diffusion domain. Thus, bacteria consume diffusing nutrients and proliferate based on the rate of consumption governed by Monod kinetics [13], Tessier kinetics [14] or other rate equations. The diffusion process is usually solved using grid-based methods, which can also be parallelized. Bacterial distribution on the grids is non-homogenous and localized to regions where biofilms are present. This leads to variable load allocation on the processors, with maximum load on the processor solving the grid points comprising most bacteria. In addition, bacterial biofilms in experiments are grown in flow cells [15], which have fluid flowing within the chambers growing biofilm. Here, computational fluid dynamics (CFD) needs to be implemented to model the effect of fluid on the mass transfer of nutrients. Such complex model system with CFD and solute mass transfer necessitates parallelization and optimization of the solving process. However, there are only a few studies that address the concerns of parallel computation of biological models [16, 17]. These studies are restricted to analysis of parallel efficiency in a single method (either CFD or solute mass transfer) and ignore the communication overhead arising from coupling multiple methods.

We develop a model to analyze and optimize parallel computations in biofilm growth simulations. In the model, we extend the load balancing model proposed by Alowayyed et al. [16] to include the communication overhead between the methods. The effects of domain size, bacterial cell distribution and mesh element size on the parallelization efficiency are analyzed. Also, we develop a simplified function based on the above parameters to obtain the optimal number of processors required to simulate different stages of biofilm growth.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

2 Methodology

2.1 Computational Methods

We have two processes involved in the biofilm model, (m1) fluid dynamics simulation and (m2) solute simulation. To model the fluid dynamics of the growth medium in the simulation domain, we solve the incompressible Navier-Stokes (NS) equation and continuity equation listed in Eq.1 and Eq.2 respectively. In Eq.1 $\boldsymbol{u}$ is the velocity vector, $\boldsymbol{p}$ is the fluid pressure, ν is the kinematic viscosity and g is the external force (gravity) acting on the fluid. In cases of biofilm growth, the knowledge of steady-state nutrient concentration is required to model the cell proliferation. There are two ways to predict the steady state velocity profiles, solve the NS and continuity equations assuming no change of velocity with time, ie., $\frac{\partial \boldsymbol{u}}{\partial t} = 0$, or solve the equations taking small time steps 'dt' until the spatial velocity values converge. In our study, for numerical stability and accuracy we use the latter method of solving the transient state flow to arrive at steady state velocity. For simulating the flow, we use FENICS [18, 19], an open source finite element based partial differential equation solver. NS and continuity equations in FENICS were implemented using Incremental Pressure Correction Scheme (IPCS) [20]. The meshing for the fluid flow domain was done using GMSH [21]. GMSH is an open source mesh generation tool. We generate adaptive meshes to simulate the flow, that is, the mesh elements get finer as they approach the surface of biofilm.

$$\frac{\partial \boldsymbol{u}}{\partial t} = \nu \nabla^2 \boldsymbol{u} - \nabla \mathbf{p} + \boldsymbol{g} \tag{1}$$

$$\nabla . \boldsymbol{u} = 0 \tag{2}$$

The second simulation (m2) is the solute convection-diffusion-consumption (CDC) simulation, modelled using Eq.3 and Eq.4. The solute concentration evolution is defined by Eq.3. where, C is the concentration of glucose, D is the diffusivity of glucose, r is the rate of consumption of glucose by the cells. The steady state velocity for estimating the convection-diffusion is obtained from the FENICS solution. This solution is coupled with the Finite Volume (FV) mesh generated in FiPy [22]. FiPy is a partial differential solver based on (FV). To solve the equations in parallel we use the solver module, PyTrilinos, a python wrapper for open source Trilinos modules [23].

$$\frac{\partial C}{\partial t} = D\nabla^2 C - \boldsymbol{u}.\nabla \mathrm{C} - \mathrm{r} \tag{3}$$

$$r = \left(\frac{\mu_m}{Y} + m\right) B \frac{C}{K + C} \tag{4}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

2.2 Modelling set-up and assumptions

The bacteria in the biofilm are modelled to occupy a set of connected grid points with the simulation domain. In this study, we analyze three different biofilm settings, (i) The initial adhesion stage where only a few cells are present, (ii) Intermediate growth stage with a hemispherical structure and (iii) A final mushroom shaped structure as shown in fig 1. For the boundary conditions in FENICS, we assume a constant velocity inlet, atmospheric pressure boundary condition at the outlet and no slip boundary conditions near the bacterial cells in the domain as mentioned in equations 5,6 and 7 respectively. The mesh is refined near the bacterial cells to improve numerical accuracy. All the simulations are carried out for a Reynold's number, Re, of 100. A fixed number of iterations is carried out such that the solution converges to a steady state.

$$\boldsymbol{u} = \boldsymbol{u_o}\,, \qquad \text{at} \quad x=0 \tag{5}$$

$$\boldsymbol{p} = 0, \qquad \text{at} \quad x=nx \tag{6}$$

$$\boldsymbol{u} = 0, \qquad \text{along biofilm surface} \tag{7}$$

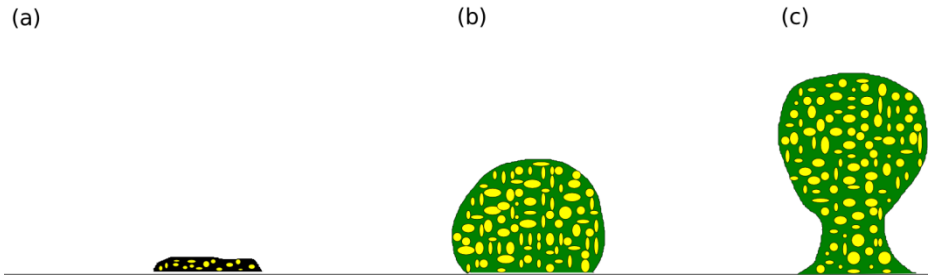

Fig. 1. Schematic of various stages of bacterial biofilm growth, (a) Stage 1: The initial adhesion stage, (b) Stage 2: Intermediate growth stage and (c) Stage 3: Mature mushroom shaped biofilm structure. Yellow color indicates the bacterial cells and dark green color indicates the extracellular polymeric substances.

We model growth dynamics of the bacteria using single substrate Monod kinetics given by Eq.4 Here, μ_m, is the maximum specific growth rate, *Y* is the mass yield coefficient, m is the metabolic maintenance coefficient, *B* is the biomass present at the grid and *K* is the saturation coefficient. Multiple studies involving the bacteria, *Pseudomonas aeruginosa*, have used Monod kinetics due to its simplicity and the availability of literature data [11, 24]. Here, Glucose (*C*) is assumed to be the critical nutrient

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

for the bacterial growth and survival. The convection-diffusion-consumption is solved for steady state by assuming $\frac{\partial C}{\partial t} = 0$. A fixed concentration inlet 'G_{ini}' is used at the inlet boundary, x = 0 and at all other boundaries no-flux boundary condition is used. We use a fixed number of iterations, large enough to let the solutions converge. The values used in the simulation are listed in table 1.

Table 1. Parameter values used in the biofilm simulations. (g_b is the quantity of biomass, expressed in grams)

Parameter	Value
Length of domain [11]	750 x 10^{-6} m
Height of domain [11]	450 x 10^{-6} m
Number of grids in FiPy simulation	1250 x 750
Initial glucose concentration, G^{ini} [11]	3 $g\ m^{-3}$
Initial mass of bacteria, B_C [11]	1.315 x 10^{-13} g_b
Half-saturation coefficient, K_s [11]	2.55 $g\ m^{-3}$
Diffusion coefficient, D_s	2.52 x 10^{-6} $m^2 h^{-1}$
Specific growth rate, μ_m [11]	0.3125 h^{-1}
Mass yield coefficient, Y [11]	0.45 $g_B\ g^{-1}$
Metabolic maintenance coefficient, m [11]	0.036 $g\ g_b^{-1}\ h^{-1}$
Reynold's Number, Re	100

To estimate the parallel performance, we adapt the models developed by Axner et al. [25] and Fox [5]. We use Eq. 8 to estimate the time taken to complete the computation through parallel execution, T_{mi}, from number of processors (P), the time for sequential computation ($T_{i,s}$), and the overheads arising within the individual process ($T_{overheads}$). The term $T_{overheads}$ does not include the communication overhead between the processes m1 and m2. We introduce an additive term T_{comm} which considers the overhead from communication between the two processes m1 and m2. Thus, Eq. 8 is now modified as Eq.9 which estimates the total time 'T' taken for the computation of both the processes, where the i in T_{mi} indicates the process number.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

$$T_{mi} = \frac{T_{i,s}}{P} + T_{overheads} \tag{8}$$

$$T = T_{comm} + \sum_{1,2} T_{mi} \tag{9}$$

Now we estimate the fractional load imbalance on each processor using the model developed by Alowayyed et al [16]. Consider $t_{j,i}$, the time taken by processor j working on process i to complete the computation. When the load is distributed properly, that is when the domain decomposition and cell data allocation to processors is done evenly, we have $t_{1,i} = t_{2,i} = t_{3,i} = \ldots = t_{P,i}$. However, due to heterogenous cell distribution in the domain and differences in spatial grid smoothness such a scenario is not possible. Thus, the fractional load imbalance $f_{l,i}$ is calculated depending on the average execution time, $<t_i>$ and maximum processor execution time t_i^m using equation 10. The speed up and parallel efficiency are quantified using Eq.11 and Eq.12 respectively.

$$f_{l,i} = \left(\frac{\left(t_i^m - \left(\frac{T_{i,s}}{P} \right) \right)}{\frac{T_{i,s}}{P}} \right) = \frac{t_i^m}{< t_i >} - 1 \tag{10}$$

$$S_p = \frac{T_{i,s}}{T_p} \tag{11}$$

$$E_p = \frac{S_p}{P} \tag{12}$$

3 Results and Discussion

Initially, we fix the domain size, the mesh smoothness and run the simulations on a single processor (sequentially) to analyze the velocity patterns and concentration contours developed in the domain containing a mature biofilm structure shown in figure 1c. As shown in figure 2, the simulations can predict the changes in velocity and glucose concentration in the vicinity of the cells.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

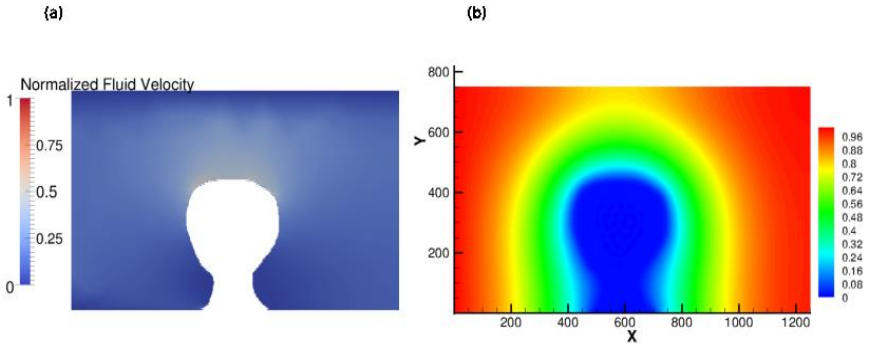

Fig. 2. Simulations on mature biofilm structure, (a) fluid dynamics simulation result showing the normalized velocity within the domain and (b) CDC simulations showing the normalized glucose concentration distribution in the domain.

All the simulations in the study were carried out on 3.20 GHz Intel® Core™ i7-6900K CPU running Ubuntu Linux 14.04. The parameters shown in table 1 were used for all the simulations, hence, the effects of change in domain size or change in fluid flow characteristics were not analyzed in the study. The total time taken for the simulations to converge to steady state were 845 s and 145 s for the fluid dynamics and CDC simulations respectively. There will be a communication overhead between the processes even when running sequentially, as indicated by the additive term in Eq. 9. In the next step, we simulated the fluid flow and nutrient diffusion patterns for the various stages of biofilm developments shown in figure 1. We restrict ourselves to these three stages of growth since after stage 3, due to nutrient depletion, there is a possibility of bacterial dispersion from the biofilm. In this study, parallel performance analysis during the biofilm dispersion process is not included due to the possibility of multiple structural configurations during the dispersion process. We varied the number of processors P from 1 to 16. The results of the simulations are shown in figure 3. We observed a plateauing of the computation time as the number of processors increased. This is due to the increase in overhead between the individual processors with increase in parallelization. Also, an interesting observation is that the stage 2 biofilms required longer processing time than stage 3 due to the larger number of mesh elements required to simulate stage 2 as shown in fig. 3c. The effect arises solely from the quantity of the mesh elements and not from the quality of the elements, since all the meshes had the same minimal element radius of 0.18. The increase in number of mesh elements could be due to the meshing algorithm being dependent on the geometry of the biofilm area. However, the communication time between the processes did not follow an established trend. Since there is always a load imbalance when using parallel processors as shown in fig. 4a, the heterogenous distribution of mesh elements would result in variable response duration for each processor to the communication signal, thereby causing inefficient inter-process communication. This inefficient communication is evident in the mesh-dense stage 2 biofilm simulations, where the mesh decomposition is much more heterogeneous.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

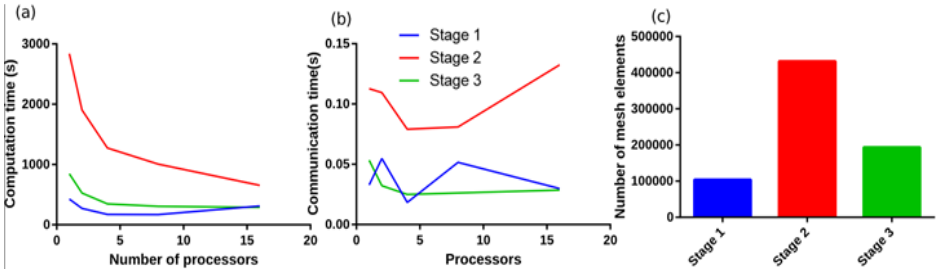

Fig. 3. Parallel performance at different stages of biofilm growth (a) change in computational time with increase in parallel processors, (b) change in communication time between processes m1 and m2 with increase in parallel processors and (c) number of mesh elements (N_e) used in the fluid dynamics simulation.

The estimated fractional load imbalance from the simulations is shown in figure 4(a). In general, the load imbalance increased with increasing number of processors, and followed a sigmoidal curve pattern indicating the asymptotic nature of the load imbalance. The asymptotic behavior can be explained from the fact that, as the number of processors increase, the heterogeneity between the meshes allotted to the individual process decreases, resulting in an equilibrium value for fractional load imbalance. Figure 4(b) shows a decrease in efficiency of parallel computation at higher processor counts. This trend is expected since there is always an efficiency loss from intra-communication overheads between the processors. We also infer that, efficiency is a function of mesh elements and number of parallel processors. The geometry of the stage 2 biofilm necessitates use of large number of mesh elements to have a refined mesh boundary. Therefore, stage 2 biofilm with large number of mesh elements operates at a higher efficiency with large number of processors (>8) and underperforms with lesser number of processors than its counterparts. Although the fractional load imbalance for stage 1 biofilms is significantly higher than stage 2 and 3 biofilms using 4 processors, the efficiency for stage 1 biofilms is marginally higher than stage 2 and 3 biofilms due to the presence of fewer meshing elements and homogenous element distribution. Thus, the average number of mesh elements per processor (N_p) determines rate of decrease in parallel efficiency. We could therefore write a simplified function,

$$E_p = E_p(P, N_e, N_P) \tag{13}$$

Increase in N_p while using large number of processors will therefore result in increased processor efficiency. Practically, this could be done by refining the fluid dynamics mesh. However, the mesh refinement should be optimized such that the trade-off between parallel efficiency and total computation time 'T' stays optimal. A similar trend is observed with the speed up values since it is indirectly proportional to the parallel computation time as shown in Eq. 12.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

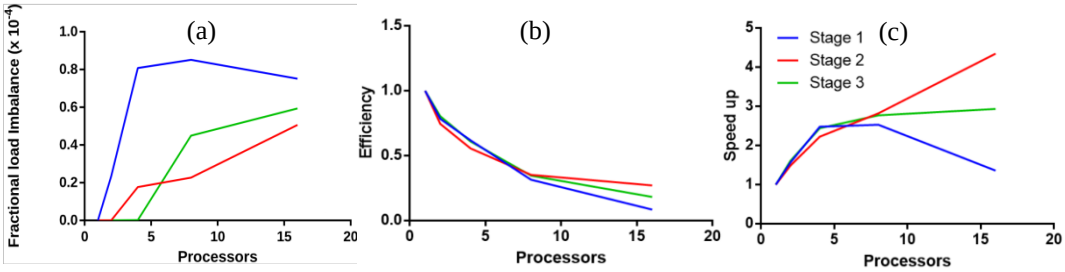

Fig. 4. Parallel efficiency test results, (a) estimate of fractional load imbalance on the processors, (b) change in parallel processing efficiency with increase in parallel processors and (c) Speed up resulting from change in number of processors.

4 Conclusion

We modeled the parallel computation efficiency at different stages of a multi-physics implementation of biofilm growth. It was found that high parallelization, at initial stages of biofilm growth simulations is not needed, since the computational efficiency from parallelization is offset by the intra-process overheads. The intermediate stage requires more parallel processors to decrease the overall computation time. This is due to the presence of large number of mesh elements at this stage. Therefore, as a rule of thumb, the number of processors needed to optimize the speed of execution of the entire biofilm growth simulation is, $(N_p)_{stage1} < (N_p)_{stage2} > (N_p)_{stage3}$. We have developed a simplified function (E_P) dependent on the number of processors, total number of mesh elements and the mesh elements per processor for optimizing the parallel efficiency in simulating bacterial biofilm growth.

5 Acknowledgment

P.S. acknowledges the Russian Science Foundation for support under RSCF #14-21-00137.

References

1. Dhatt, G., Lefrançois, E., Touzot, G.: Finite element method. John Wiley & Sons (2012)
2. Chen, S., Doolen, G.D.: Lattice Boltzmann method for fluid flows. Annual review of fluid mechanics 30, 329-364 (1998)
3. Versteeg, H.K., Malalasekera, W.: An introduction to computational fluid dynamics: the finite volume method. Pearson Education (2007)
4. Zhang, L., Wang, Z., Sagotsky, J.A., Deisboeck, T.S.: Multiscale agent-based cancer modeling. Journal of mathematical biology 58, 545-559 (2009)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

5. Fox, G.C., Johnson, M.A., Lyzenga, G.A., Otto, S.W., Salmon, J.K., Walker, D.W.: Solving problems on concurrent processors. Vol. 1: General techniques and regular problems. Prentice-Hall, Inc. (1988)
6. Fozard, J.A., Lees, M., King, J.R., Logan, B.S.: Inhibition of quorum sensing in a computational biofilm simulation. Biosystems 109, 105-114 (2012)
7. Morgenroth, E., Wilderer, P.A.: Influence of detachment mechanisms on competition in biofilms. Water Research 34, 417-426 (2000)
8. Picioreanu, C., Van Loosdrecht, M.C., Heijnen, J.J.: Two-dimensional model of biofilm detachment caused by internal stress from liquid flow. Biotechnology & Bioengineering 72, 205-218 (2001)
9. Weitz, J.S., Hartman, H., Levin, S.A.: Coevolutionary arms races between bacteria and bacteriophage. Proceedings of the National Academy of Sciences of the United States of America 102, 9535-9540 (2005)
10. Picioreanu, C., Vrouwenvelder, J., Van Loosdrecht, M.: Three-dimensional modeling of biofouling and fluid dynamics in feed spacer channels of membrane devices. Journal of Membrane Science 345, 340-354 (2009)
11. Fagerlind, M.G., Webb, J.S., Barraud, N., McDougald, D., Jansson, A., Nilsson, P., Harlén, M., Kjelleberg, S., Rice, S.A.: Dynamic modelling of cell death during biofilm development. Journal of theoretical biology 295, 23-36 (2012)
12. Popławski, N.J., Shirinifard, A., Swat, M., Glazier, J.A.: Simulation of single-species bacterial-biofilm growth using the Glazier-Graner-Hogeweg model and the CompuCell3D modeling environment. Mathematical biosciences and engineering: MBE 5, 355 (2008)
13. Han, K., Levenspiel, O.: Extended Monod kinetics for substrate, product, and cell inhibition. Biotechnology and bioengineering 32, 430-447 (1988)
14. Beyenal, H., Chen, S.N., Lewandowski, Z.: The double substrate growth kinetics of Pseudomonas aeruginosa. Enzyme and Microbial Technology 32, 92-98 (2003)
15. Sternberg, C., Tolker-Nielsen, T.: Growing and analyzing biofilms in flow cells. Current protocols in microbiology 1B. 2.1-1B. 2.15 (2006)
16. Alowayyed, S., Závodszky, G., Azizi, V., Hoekstra, A.: Load balancing of parallel cell-based blood flow simulations. Journal of Computational Science 24, 1-7 (2018)
17. Cytowski, M., Szymanska, Z.: Large-scale parallel simulations of 3d cell colony dynamics. Computing in Science & Engineering 16, 86-95 (2014)
18. Logg, A., Mardal, K.-A., Wells, G.: Automated solution of differential equations by the finite element method: The FEniCS book. Springer Science & Business Media (2012)
19. Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M.E., Wells, G.N.: The FEniCS project version 1.5. Archive of Numerical Software 3, 9-23 (2015)
20. Guermond, J.-L., Minev, P., Shen, J.: An overview of projection methods for incompressible flows. Computer methods in applied mechanics and engineering 195, 6011-6045 (2006)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38

21. Geuzaine, C., Remacle, J.F.: Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. International journal for numerical methods in engineering 79, 1309-1331 (2009)
22. Guyer, J.E., Wheeler, D., Warren, J.A.: FiPy: Partial differential equations with Python. Computing in Science & Engineering 11, (2009)
23. Heroux, M.A., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., Phipps, E.T.: An overview of the Trilinos project. ACM Transactions on Mathematical Software (TOMS) 31, 397-423 (2005)
24. Picioreanu, C., Kreft, J.-U., Klausen, M., Haagensen, J.A.J., Tolker-Nielsen, T., Molin, S.: Microbial motility involvement in biofilm structure formation–a 3D modelling study. Water science and technology 55, 337-343 (2007)
25. Axner, L., Bernsdorf, J., Zeiser, T., Lammers, P., Linxweiler, J., Hoekstra, A.G.: Performance evaluation of a parallel sparse lattice Boltzmann solver. Journal of Computational Physics 227, 4895-4911 (2008)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_38