Hybrid Architecture for Simulation of Blood Flow with Foreign Bodies

Łukasz Faber, Krzysztof Boryczko, Marek Kisiel-Dorohinicki
AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: {faber,boryczko,doroh}@agh.edu.pl

KEYWORDS

agent-based simulation, dissipative particle dynamics

ABSTRACT

The new methods of diagnosis are often more sensitive and speed up its process allowing for a more effective treatment. Yet they need to be carefully tested before they can be used in practice. The paper concerns the problems of developing nanorobots, which would circulate in the bloodstream of the human body gathering information about its condition. Simulation of these new ways of diagnosis is the main motivation behind the presented work. Introduction of such nanorobots into the bloodstream puts several requirements on their mechanical properties. Thus, our first goal is to build a model to simulate the blood flow with nanorobots present in the capillary vessels and medium-sized blood vessels. We split it into two separate initial models and implementations. The first one is a simulation of the blood flow using particle-based methods in order to determine the appropriate mechanical parameters of nanorobots and verify their behavior. The second one is a multi-agent simulation that will allow to evaluate the usefulness of the data collected and prototype the system performing functions of nanorobots within the specified constraints.

INTRODUCTION

Early and accurate diagnosis is one of the most important factors that make it possible to provide a patient with an effective treatment. This leads to constant improvements to traditional and popular methods of diagnosis – usually noninvasive (e.g., diagnostic imaging). However, technological development makes it possible to introduce new, better, more sensitive methods that speed up the process of diagnosis (e.g., using the optoelectronics for marker detection in exhaled air). Moreover, it is not an end of possible ways to perform an early diagnosis. The next step is nanotechnology and particularly nanorobotics. In our work, we consider the existence of nanorobots able to circulate along with blood cells in the circulation system of the human body. These nanorobots would have the ability to gather and transmit data about the (internal) state of the body. On the basis of this data it would be possible to identify a possible condition of a patient.

The goal of the paper is to elaborate on possible techniques and tools for simulation of nanorobots circulating in the vessels of the human body and gathering information about its state. We try to identify the best solutions covering different aspects of the simulation and propose an architecture with their cooperation in mind. Simulation of blood flow and capillary vessels is usually performed using particle-based methods [1]. However, it is usually difficult to extend these models with more 'intelligent' behavior. When using the term *'intelligent' behavior* we think of some specific aspects: *1*) generation of the biological properties of the vessels and blood (e.g., temperature, permeation of other substances); *2*) behavior of injected nanorobots, and *3*) external components of the system (e.g., external sensors or data collectors). On the other hand, agent-based simulations are perfect for such work. Therefore, we want to add agent-based simulations to the particle models.

From this background we can separate two aspects of our work:

1) Simulation of the blood flow in capillary vessels of the human body with the additional presence of foreign bodies using particle-based methods.
2) Estimation of the scope and usefulness of the data obtainable from such an environment using a distributed multi-agent system.

These aspects, although they touch completely different fields, are strictly related. The former will help to specify *physical* parameters and constraints. The latter will possibly lead to extending the particle simulation in order to emulate the environment properties in a more realistic way.

In order to reach these goals we are building two simulation applications:

1) one based on particle models (Dissipative Particle Dynamics – DPD, Smoothed Particle Hydrodynamics – SPH, Smoothed DPD – SDPD, TC-FPM – Thermodynamically Consistent Fluid Particle Model) for simulating the blood flow with nanorobots present,
2) one agent-based that will be able to simulate behaviors of nanorobots and their environment for testing the data collection possibilities; this application will be also responsible for extending the previous one with biological data.

Both of these applications require construction of the model, software implementation, testing and running large simulations.

For the first application, we built the model of the blood flow in capillaries, based on modern particle methods: SDPD and TC-FPM. The software is built using the C language. In the later stages the OpenCL reimplementation is planned.

Similarly, for the second component, we developed the agent model for the simulation of nanorobots as a preliminary

Proceedings 28th European Conference on Modelling and Simulation ©ECMS Flaminio Squazzoni, Fabio Baronio, Claudia Archetti, Marco Castellani (Editors)
ISBN: 978-0-9564944-8-1 / ISBN: 978-0-9564944-9-8 (CD)

requirement for the implementation of the simulation platform. We should note, that the characteristics of these models are quite specific for the problem: the agents are very small but, on the other hand, quite capable in terms of their functions. They can perform full range of complicated behavior: sensing, acting, communication, movements. To the best of authors knowledge, there are no similar, ready-to-use models.

Apart from discussing the techniques for both simulation models, in the paper we present the architectural description of both components and propose the mechanism of their interaction. The actual evaluation of the architecture is based on the experiences gained during independent runs of the simulations of the blood flow and the agents, yet the results encourage further work in this direction.

BACKGROUND AND TECHNICAL SOLUTIONS

In this section we will review both discussed simulation models and most popular tools used with them.

Blood Flow Simulations

Blood is a highly complex tissue with its cells suspended in a liquid environment known as blood plasma. An important fact is that the blood and the vascular bed are the integral parts of all tissues of the body. Thus, disorders in physiology of individual body organs result in changes to the composition of the blood and vice versa – blood disorders and changes in its composition have a significant effect on the entire body.

The physiological task of the blood is transport. The blood transports oxygen, carbon dioxide, nutrients, hormones, metabolic products and cells. The task of the oxygen carrier is fulfilled by erythrocytes which are the most numerous blood cells and constitute about 45% of its volume. Their shape resembles a biconcave lens with a diameter of about 8 μm. This shape ensures a relatively high surface area to volume ratio and allows to adjust size to the capillaries. Research has shown that erythrocytes are able to flow through the vessels with the diameter of about 4 μm.

Number, shape and flexibility of erythrocytes have a direct impact on macroscopic parameters describing the hydrodynamic properties of blood. They have been a subject of numerous studies and simulations. A lot of models were developed in order to study the flow of blood in large blood vessels as well as the behavior of its cells when traveling through the capillaries. It is worth to note that there is a need to adjust the time-space scale of the used simulation method to the simulated phenomenon. Thus, in flow simulations in large (macroscopic) blood vessels the CFD methods or the Smoothed Particle Hydrodynamics (SPH) method are used. Simulation of the blood flow in capillaries, in mesoscale, was for a long time quite troublesome as there was no adequate model for this scale. First models were relatively large simplifications of the phenomenon [2], [3], [4]. Only in 1992, the Dissipative Particle Dynamics (DPD) method was proposed [5]. It was designed to simulate phenomena happening in the mesosopic scale. This method had several artifacts that were later removed in a method called Fluid Particle Model (FPM). As shown in [6] these methods and their combinations are well-suited for the accurate modeling of complex fluid flows in the size range 10 nm to 100 μm. On this basis, the model of blood flow in the capillaries was proposed that took into account the flexibility of erythrocytes, their interactions with the walls of the capillaries and the presence of fibrinogen [7], [8].

Multi-Agent Systems

In a Multi-Agent System (MAS), the process of solving a computational problem is decomposed into autonomous, intelligent entities called agents. Agents can actively follow some goals, based on their beliefs (perceived environment), plan their actions ahead and learn from experiences.

In order to achieve this, MAS are often used along with other methods from Artificial Intelligence or Machine Learning. Agents may also own resources and interact with each other. The structure of the system is expected to emerge from these decentralized agent interactions. MAS are thus a bottom-up and decentralized approach to system design and problem solving.

Multi-agent systems and, more generally, the concept of an intelligent agent have found multiple applications, both as a way to model complex systems and as a programming paradigm to implement them. Several established agent-based technological solutions exist, including FIPA compliant, fully-fledged environments like JADE or JADEX, where agents are a basic unit of software abstraction. As an example of another approach, there are also minimalistic and easy-to-use tools like NetLogo, where agents are only present at a domain level, as means of problem decomposition.

The first class of systems can be used to solve any problem that benefits from using an agent-oriented approach. However, in some particular classes of applications, this can be very inefficient. This happens especially in the case of simulation and computational applications, where agents and their behaviors are well defined. Such MAS might not need to be open to other systems, require the possibility of code migration nor support FIPA-compliant communication.

In these cases, systems of the second class are more efficient and a much better choice. However, they suffer from other drawbacks, as they do not support component-oriented approach, which affects code reusability and make more complex problems hard to program. Also, these systems are usually not suited for running in distributed environments. In other words, they do not scale well with bigger problems or more complex systems.

Usually, Multi-Agent Systems have following properties [9]: agents are autonomous and independent, agents are aware only of their local environment; in other words, no agent has all the data in the system, there is no central control component (i.e., an agent that would be able to supervise the whole system). Some researchers add also asynchronicity of the system to this set of properties [10]. Moreover, to execute their tasks in such a system agents usually need to have means of communication and mutual interactions. The way they are handled depends on the environment and system requirements.

Agent-based (and multi-agent) systems are well-researched and extensively used solutions to a very wide range of different problems: transportation, metaheuristic problem solving [11], multi-robotics, social simulations, etc. Their properties make

them well-suited for distributed systems [12]: they offer modularity and scale efficiently.

The question arises: why did we choose multi-agent systems? The simple answer is: agents are successfully used in multi-robotics scenarios due to their properties [13].

The MAS properties, that we described above, correspond clearly with the properties of the nanorobotic system that could be located in the blood vessels. In such conditions: nanorobots need to be autonomous as there is no possibility for all of them to communicate, nanorobots are unable to 'know' the whole environment as they are constrained in terms of computational power and storage capabilities, there is no possibility to introduce one central component that would be able to supervise all nanorobots. Moreover, the system is completely asynchronous. It also needs to adapt to constantly changing properties in terms of agents relative and absolute spatial positions, their communication and interaction partners.

As a second phase requires a simulation of a patient body properties, we will need to use an existing or implement a new agent-based simulation platform. For this purpose we should briefly review some of the possible choices of simulation platforms.

There exists a plethora of multi-agent frameworks which may be used to support the construction of simulation systems. Some of them are oriented to specific kinds of simulation (see [14], [15]): e.g., simulating of movement of entities with 3D visualization (e.g., breve [16]), possibility of visual programming (e.g., SeSam [17]). When looking for universal agent-based simulation frameworks (especially in open-source software domain), one should consider such products as, for example, Galatea [18], RePast [19], Mason [20]. Other ones are general-purpose agent-based programming frameworks (e.g., JADE [21]) that may be of course adapted to any kind of simulation. MadKit [22] should also be mentioned as a framework for simulating complex populations (following Agent/Group/Role paradigm).

The frameworks described in the next paragraphs were selected as the most promising examples of general-purpose agent-based simulation frameworks in the open-source market.

MASON is an agent-oriented simulation framework developed at George Mason University. It is advertised as fast, portable, 100% Java based. The multi-layer architecture brings complete independence of the simulation logic from visualization tools. The models are self-contained and may be included in other Java-based programs.

The programming model of MASON follows the basic principles of the object-oriented design. An agent is instantiated as an object of a class, added to a scheduler and its *step* method is called during the simulation. There are no predefined communication nor organization mechanism. There are neither ready-to-use distributed computing facilities nor component-oriented solutions.

RePast is a widely used agent-based modeling and simulation tool. It has multiple implementations in several languages and built-in adaptive features such as genetic algorithms and regression. The framework uses fully concurrent discrete event scheduling. Dynamic access to the models in the runtime (introspection) is possible using a graphical user interface.

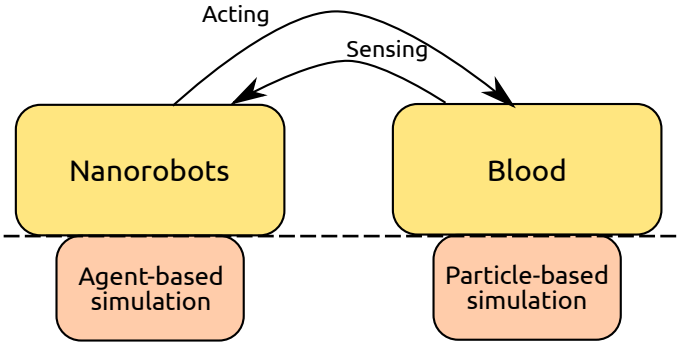


Fig. 1: Overview of the architecture. We can pinpoint two layers: conceptual (upper) and implementational (lower). The communication in the former is related to the way agents obtain and generate information. Agents sense the environment and (possibly) act on it.

In Repast Simphony, a new organizational concept called 'context' was introduced. It consists of a group of unorganized agents (they may be organized using so-called projections) and may create a hierarchical structure (context can have many sub-contexts and so on). This idea affects the perception of agents in such way, that an agent in the sub-context also exists in the parent context, but the reverse is usually not true.

MadKit is a modular and scalable multi-agent platform written in Java, aimed at modeling different agent organizations, groups and roles in artificial societies. It is built based on a so called Agent/Group/Role organizational model, using a plugin-based architecture. The architecture of MadKit is based on micro-kernels which provide only the basic facilities: local messaging, management of groups and roles, launching and killing of agents. Other features (remote messages, visualization, monitoring and control of agents) are performed by agents. Both thread based and scheduled agents may be developed.

Simulations do not require any particular structure or model to run. However, it is possible to add an arbitrary scheduler or create complex agent communities and relationships. Agents can locate other agents having some specific role or belonging to some particular group. Agents can communicate with each other using these roles or group membership (i.e. using broadcast messages).

ARCHITECTURE

The system has two heterogeneous components that can be discussed independently. We have built two separate models for initial tests. The basic idea behind the system is to use particle-based simulation techniques for the environment and agent-based ones for the foreign bodies 'injected' into the flow. In our particular case we define environment as capillary vessels, and the external bodies are nanorobots. An overview of the architecture is shown in Figure 1.

The system is parallel by the nature of the performed computations. The current implementations are based on C (Message Passing Interface – MPI) and Java, but future implementations will be prepared mostly for highly-parallel environments like GPGPU.

Particle-based Component

For the first phase, the model of the blood flow in capillary vessels will be built. It will be based on modern particle methods that take into consideration internal state of particles (thermodynamically consistent) – SDPD (Smoothed DPD) [23] and TC-FPM (Thermodynamically Consistent FPM) [24]. Simulations will make it possible to set the mechanical parameters of the introduced objects that are required for their proper functioning in the circulatory system. Moreover, the physical parameters of the environment that should be possible to determine around the flowing object will be defined. Application of these simulation methods will allow to determine more parameters, including the internal temperature of tissues as its differences, in the context of other information, may indicate the presence of cancer.

During the first phase we will implement the simulation application. Then, we will perform extensive tests in order to obtain proper parameters of the simulation and to ensure high reliability and optimization (in terms of simulation speed). As mentioned earlier, we plan to run the simulation in a highly parallel environment. The obtained parameters will be used for specifying constraints of an agent operating 'inside' a nanorobot.

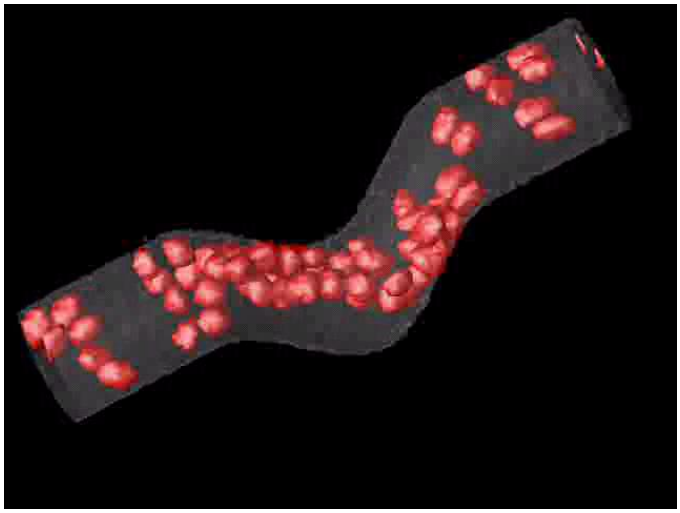


Fig. 2: Flow of the bodies in a larger vessel.

Figures 2 and 3 show a flow of the bodies in larger vessel during simulations with aim to determine how their shapes influence viscosity. Figure 4 is the part of determining mechanical properties of the bodies. We try minimize their influence on the flow of blood. In our simulations we use properties similar to erythrocytes but we omit proportions of the volume to the area as it is relevant only for an oxygen transport.

Our goal is to specify shape of a nanorobot. Some of the sample, initial results include the computation of Reynolds numbers for flat and biconcave bodies in relation to the number of simulation steps. As mentioned before, we do not have to keep proportions that are normally required for erythrocytes because there is no oxygen transport involved. If possible, we would prefer to increase the volume but keep all of the mechanical properties. Thus, our first attempt is to use not

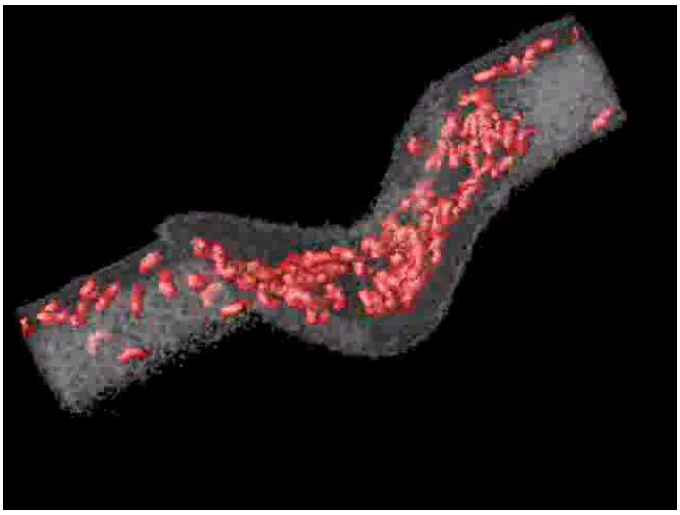


Fig. 3: Flow of the bodies in a larger vessel.

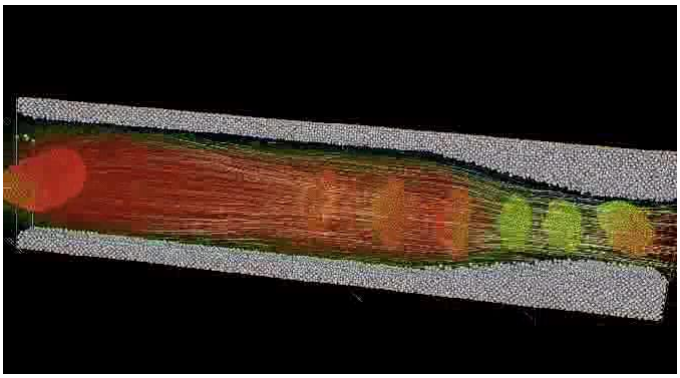


Fig. 4: Visualization of the flow in simulations leading to determining of mechanical properties of the bodies.

biconcave but flat nanorobots. We compared the behavior of these two possibilities (flat and biconcave nanorobots). Figures 6, 7 and 8 show shapes of the nanorobots successively in: 0th, 2500th and 4000th steps of the simulation. It is noticeable that flat ones are less flexible and this reflects on the blood flow. This fact is also visible in Figure 5.

Agent-based Component

The agent-based component should have following general properties:

- It should consist mostly of small agents strictly constrained in terms of computational capacity and the available memory size.
- It should be able to simulate agents separation in human body and spatial distribution.
- It should be possible to introduce larger entities into the system (but external to the body) that will be able to execute more complicated tasks (like data gathering).
- There is a need to introduce spatial properties (positions) and neighborhood for both the agents playing the role of the environment and those controlling nanorobots. For the former, because they are parts of a specific spatial

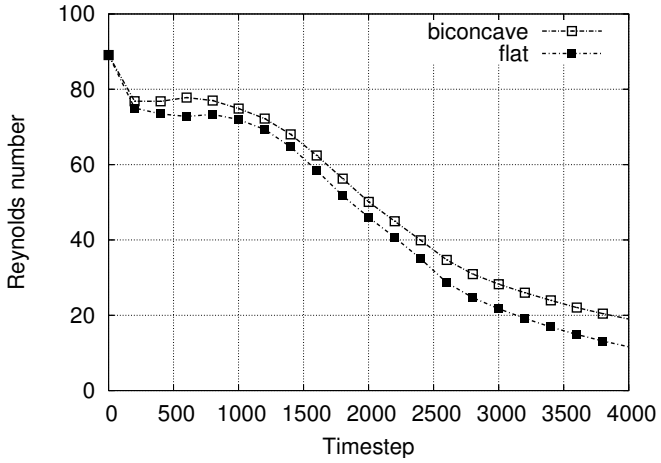


Fig. 5: Value of the Reynolds number (in program-dependent units) in consecutive steps of the simulation.

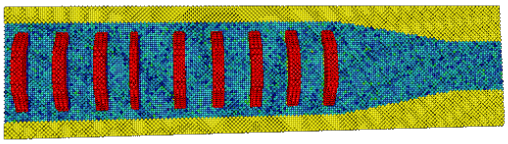


Fig. 6: Visualization of the 0th step of the simulation (initial configuration).

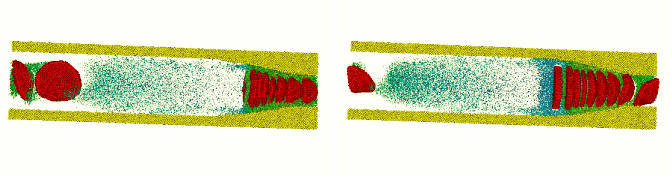
structure, and for the latter because there is a need to know which agents can communicate in order to simulate aforementioned communication problems.

- The nanorobots need to pass the collected data to an external component that will be able to process it having a more broad view (as it will have more information about the whole system and thus about the body).

There are several possibilities how to implement such an agent-based simulation platform. Initially, we test our requirements using the AgE platform [25]. The AgE platform includes two types of agents: heavyweight agents and lightweight ones.

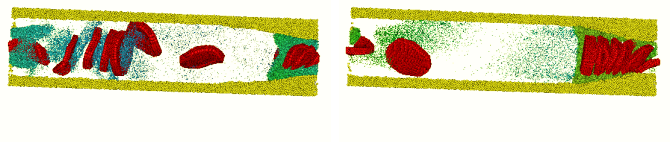
Heavy agents are realized as separate threads, as in the JADE platform. They communicate through asynchronous message passing. This is an effective model when the number of agents is small or agent interactions are sparse (i.e. each agents communicates only with a small number of other agents).

Moreover, AgE introduces lightweight agents. Populations of such agents are thread-contained and execute pseudo-concurrently, one step each at a time. Some restrictions on how agents may change each other state, along with communication constraints, described further below, emulate concurrency effects and execution interleaving. From these agents perspective, they are effectively processed in parallel.



(a) Biconcave nanorobots. (b) Flat nanorobots.

Fig. 7: Visualization of the 2500th step of the simulation.



(a) Biconcave nanorobots. (b) Flat nanorobots.

Fig. 8: Visualization of the 4000th step of the simulation.

Thus, the platform API available to heavyweight and lightweight agents is very similar. The main difference is in efficiency and determinism, as needed in a particular case.

Each agent in AgE belongs to some environment. Agents can communicate with each other within their environment or query it to acquire some information. In the case of heavy agents, the environment simply consists of multiple distributed nodes. All heavy agents on all connected nodes belongs to it. When it comes to lightweight agents, environments are treated as agents themselves. These *aggregate* lightweight agents also have properties, behavior, and also execute in some higher level environment. Thus lightweight agents form hierarchies, as shown in Figure 9. The root agent is a heavy one and is called a *workplace*. A workplace encompasses all the hierarchy within its thread and initiates step based, pseudo-concurrent execution.

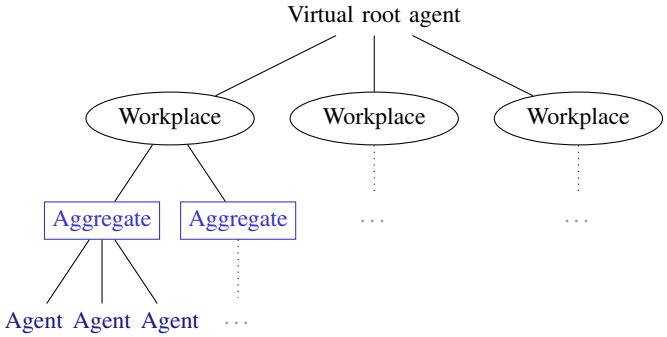


Fig. 9: Agent tree structure in AgE. Workplaces are units of distribution among separate AgE nodes. Each workplace may consist of any number of aggregates and simple agents. The *virtual root agent* represents a common root for all workplaces which handles communication facilities on their level.

In future, additional functionalities of the platform should include monitoring of the system for purpose of having an online information about its state and distribution of the simulation.

At least three broad kinds of agents are introduced:

1) one for simulation of the environment,
2) one for simulation of the nanorobots, and
3) one for being in role of possible external components.

Their general relations are shown in Figure 10. The red circles are agents that act as body tissues. They are spatially close to each other and can communicate locally in a peer-to-peer fashion. The blue circles are external component agents that also are spatially close and have efficient and a reliable peer-to-peer or leveled communication. The yellow circles are the agents controlling nanorobots. They can communicate with each other when they are in close neighborhood. They can also communicate with the external components (the bottom part of the figure) or with the body tissues (the upper part of the figure). Obviously, the interaction with the body tissues is regarded as 'communication' only from the simulation point of view. In the model such interactions are analogous to 'perceiving of' and 'acting on' the environment.

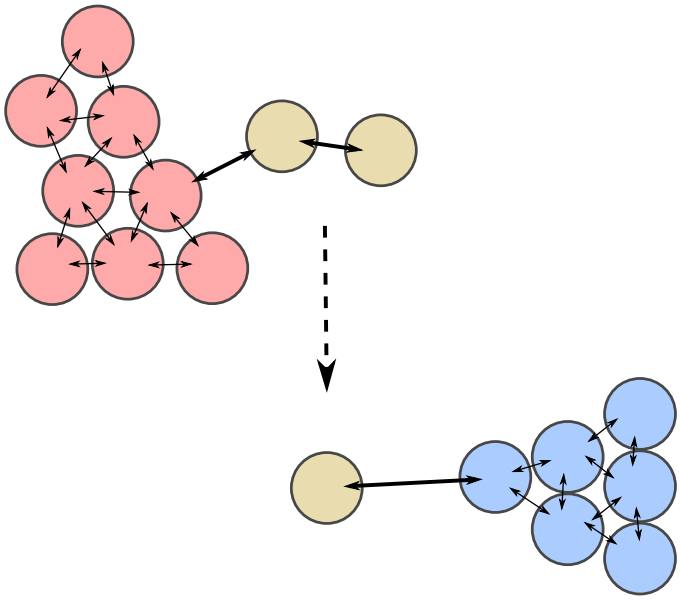


Fig. 10: Communication between agent types.

Communication between both models may be realized in many ways. Our first, the simplest, attempt was to perform simulations separately (we call it *offline* version). We run the particle-based simulation and we obtain physical properties of the system which we then submit to the agent-based simulation. Data acquisition can be done in two ways: we can dump all simulation data in every step or we can interpolate physical properties (like temperature, speed, etc.) during the run and dump only these values.

In order to allow feedback from the agent-based simulation to the particle-based one, we need to implement concurrently running simulations that will be exchanging data *online*. In such scenario, the blood flow simulation will be forwarding data (full or selected interpolated properties) to the agent system in every step. The second system will be performing its own computations and will return results to the first one. As the complexity and execution time of the particle-based simulation is higher than that of the agent-based simulation, both systems will be able to run simultaneously. Sample realization is shown in Figure 11.

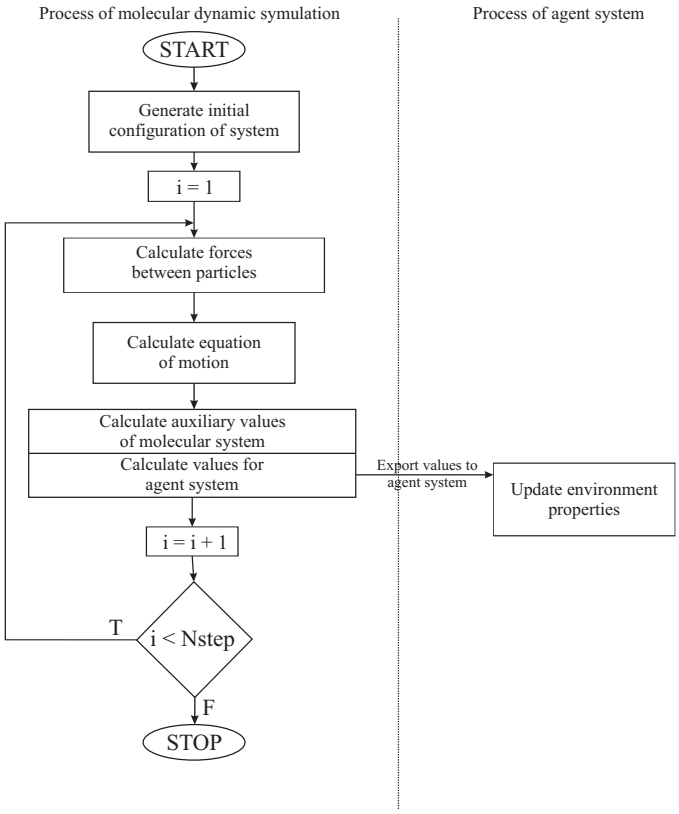


Fig. 11: Processing of particle simulation with data forwarding to the agent system.

CONCLUSIONS AND FUTURE WORK

We have presented the initial model and implementation of the hybrid simulation system for blood flows with foreign bodies. The system comprises of two heterogeneous simulation: the particle-based one and the agent-based one. The former is used as a tool to simulate physical properties of the environment (i.e. capillary vessels) whilst the latter is responsible for the generation of the additional, biological properties of the environment.

Achievement of the aforementioned objectives opens up several possibilities for further research in all of related areas. For example, for agent systems it could be improving the behavior of the nanorobots, diagnosis deduction, etc. In long-term it should also lead to creation of working prototypes.

The architecture was described in the context of the work on the system for obtaining data about nanorobots in the bloodstream of the human body.

The most important aspect of future work is a merge of these two, currently separate models and applications into

one, heterogeneous model possibly working as one application or data-exchanging cluster of applications. Moreover, our implementation goals are related to creating high-performance simulations using GPGPU.

ACKNOWLEDGEMENT

The research reported in the paper was supported by the grant "Hybrid model of the early detection of internal diseases based on the paradigm of interacting particles and multi-agent system" (No. UMO-2013/09/N/ST6/01011) from the Polish National Science Centre.

REFERENCES

[1] W. Dzwinel, K. Boryczko, and D. A. Yuen, "A discrete-particle model of blood dynamics in capillary vessels," *Journal of colloid and interface science*, vol. 258, no. 1, pp. 163–173, 2003.

[2] R. Hsu and T. Secomb, "Motion of nonaxisymmetric red blood cells in cylindrical capillaries," *Journal of biomechanical engineering*, vol. 111, no. 2, pp. 147–151, 1989.

[3] S. R. Keller and R. Skalak, "Motion of a tank-treading ellipsoidal particle in a shear flow," *Journal of Fluid Mechanics*, vol. 120, pp. 27–47, 1982.

[4] P. Mazeron, S. Muller, and H. El Azouzi, "Deformation of erythrocytes under shear: a small-angle light scattering study," *Biorheology*, vol. 34, no. 2, pp. 99–110, 1997.

[5] P. Hoogerbrugge and J. Koelman, "Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics," *EPL (Europhysics Letters)*, vol. 19, no. 3, p. 155, 1992.

[6] K. Boryczko, W. Dzwinel, and D. A. Yuen, "Parallel implementation of the fluid particle model for simulating complex fluids in the mesoscale," *Concurrency and computation: practice and experience*, vol. 14, no. 2, pp. 137–161, 2002.

[7] ——, "Dynamical clustering of red blood cells in capillary vessels," *Journal of Molecular Modeling*, vol. 9, no. 1, pp. 16–33, 2003.

[8] ——, "Modeling fibrin aggregation in blood flow with discrete-particles," *Computer methods and programs in biomedicine*, vol. 75, no. 3, pp. 181–194, 2004.

[9] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[10] R. A. Flores-Mendez, "Towards a standardization of multi-agent system framework," *Crossroads*, vol. 5, no. 4, pp. 18–24, 1999.

[11] A. Byrski and M. Kisiel-Dorohinicki, "Agent-based meta-heuristic approach to discrete optimization," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*. IEEE, 2011, pp. 508–512.

[12] H. S. Nwana, "Software agents: An overview," *The knowledge engineering review*, vol. 11, no. 03, pp. 205–244, 1996.

[13] A. Soriano, E. J. Bernabeu, A. Valera, and M. Vallés, "Multi-agent systems platform for mobile robots collision avoidance," in *Advances on Practical Applications of Agents and Multi-Agent Systems*. Springer, 2013, pp. 320–323.

[14] C. Nikolai and G. Madey, "Tools of the trade: A survey of various agent based modeling platforms." *Journal of Artificial Societies & Social Simulation*, vol. 12, no. 2, 2009.

[15] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: Review and development recommendations," *Simulation*, vol. 82, no. 9, pp. 609–623, 2006.

[16] J. Klein, "Breve: a 3d environment for the simulation of decentralized systems and artificial life," in *Proceedings of the eighth international conference on Artificial life*, 2003, pp. 329–334.

[17] N. Ventroux, A. Guerre, T. Sassolas, L. Moutaoukil, G. Blanc, C. Bechara, and R. David, "Sesam: An mpsoc simulation environment for dynamic application processing," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1880–1886.

[18] J. Dávila and M. Uzcátegui, "Galatea: A multi-agent simulation platform," in *Proceedings of the International Conference on Modeling, Simulation and Neural Networks*, 2000.

[19] N. Collier and M. North, "Parallel agent-based simulation with repast for high performance computing," *Simulation*, vol. 89, no. 10, pp. 1215–1235, 2013.

[20] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan, "Mason: A new multi-agent simulation toolkit," in *Proceedings of the 2004 SwarmFest Workshop*, vol. 8, 2004.

[21] F. Bellifemine, A. Poggi, and G. Rimassa, "Jade–a fipa-compliant agent framework," in *Proceedings of PAAM*, vol. 99, no. 97-108. London, 1999, p. 33.

[22] O. Gutknecht and J. Ferber, "The madkit agent platform architecture," in *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*. Springer, 2001, pp. 48–55.

[23] P. Espanol and M. Revenga, "Smoothed dissipative particle dynamics," *PHYSICAL REVIEW-SERIES E-*, vol. 67, no. 2; PART 2, pp. 026 705–026 705, 2003.

[24] M. Serrano and P. Espanol, "Thermodynamically consistent mesoscopic fluid particle model," *Physical Review E*, vol. 64, no. 4, p. 046115, 2001.

[25] Ł. Faber, K. Piętak, A. Byrski, and M. Kisiel-Dorohinicki, "Agent-based Simulation in AgE Framework," in *Advances in Intelligent Modelling and Simulation*. Springer, 2012, pp. 55–83.

ABOUT THE AUTHORS

Łukasz Faber obtained his M.Sc. in 2012 at AGH University of Science and Technology in Cracow and is currently a Ph.D. student at the Department of Computer Science of AGH-UST. His research interests include agent-based modeling and distributed systems.

Krzysztof Boryczko received his Ph.D. degree in computer science in 1992 and D.Sc. degree in computer science in 2004, both from the AGH University of Science and Technology, where he is now Associate Professor at the Department of Computer Science, Faculty of Computer Science, Electronics and Telecommunications. His research interests focus on large-scale simulations with particle methods and on implementations of particle methods on Graphical Processing Units. He is also interested in scientific visualisation, feature extraction and clustering algorithms for analysis of simulation data.

Marek Kisiel-Dorohinicki obtained his Ph.D. in 2001 at AGH University of Science and Technology in Cracow. He works as an assistant professor at the Department of Computer Science of AGH-UST. His research focuses on intelligent software systems, particularly using agent technology and evolutionary algorithms, but also other soft computing techniques.