Teaching Heart Modeling and Simulation on Parallel Computing Systems

Andrey Sozykin[1,2(✉)], Mikhail Chernoskutov[1,2], Anton Koshelev[1,2], Vladimir Zverev[1,2], Konstantin Ushenin[1,2], and Olga Solovyova[1,2,3]

[1] Institute of Mathematics and Mechanics UrB RAS, Ekaterinburg, Russia
[2] Ural Federal University, Ekaterinburg, Russia
Andrej.Sozykin@urfu.ru
[3] Institute of Immunology and Physiology UrB RAS, Ekaterinburg, Russia

Abstract. High Performance Computing (HPC) is an interdisciplinary field of study, which requires learning a number of topics, including not only parallel programming, but also numerical methods and domain science. Stand-alone parallel computing courses are insufficient for thorough HPC education. We present an interdisciplinary track of coherent courses devoted to modeling and simulation of the heart on parallel computing systems for master students at the Ural Federal University. The track consists of three modules: parallel and distributed computing, heart modeling, and numerical methods. Knowledge of numerical methods and heart modeling provides the students with the ability to acquire profound parallel programming skills by working out on the comprehensive programming assignment and complex heart modeling projects. Interdisciplinary approach also increases students' motivation and involvement.

Keywords: High performance computing · Distributed computing · HPC education · Heart simulation · Living system simulation

1 Introduction

High Performance Computing (HPC) is an interdisciplinary field of study, which requires learning a number of topics. To be able to solve state-of-the-art scientific and engineering problems on parallel computing systems, students have to study not only parallel programming, but also applied mathematics, numerical methods, and domain science. Therefore, stand-alone parallel computing courses are insufficient for thorough HPC education. Instead, the interdisciplinary training programs for various domain sciences are needed.

We present an interdisciplinary track of courses devoted to modeling and simulation of the heart on parallel computing systems. The track consists of the following modules: parallel and distributed computing, heart modeling, and numerical methods. Each module includes both theoretical and hands-on courses.

The course track is primarily aimed at master students in Computer Science, but it is also available for other students. The courses are taught by the Institute of Mathematics and Computer Sciences of the Ural Federal University in

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 102–113, 2015.
DOI: 10.1007/978-3-319-27308-2_9

cooperation with the Institute of Immunology and Physiology UrB RAS, and the Institute of Mathematics and Mechanics UrB RAS.

The initiative of creating the track was promoted by the scientific laboratory "Modeling of living systems on supercomputers", which carries out research on cardiac modeling [9,14–16]. At present, computer simulations in living systems have become an essential instrument of the medical science and practice, pharmacy, and education. As a part of the global international projects the Physiome [6] and the Virtual Physiological Human [10], the computer models of various systems, organs, and tissues are developed. The HPC is required to tackle biomedical problems due to the complexity of living system models, the enormous amount of input data, and the number of computational operations.

The important goal of the course track is to cultivate student interest in computational modeling and simulation using HPC, and to involve them into research work.

2 Related Work

Nowadays, the interdisciplinary HPC courses, workshops, and educational programs are beginning to emerge. In summer 2011, the City University of New York conducted a three week workshop on cardiac electrophysiology modeling [3]. The workshop combined lectures, hands-on lab experiments, and simulation on the graphics processor units (GPU). Students studied initiation of the spiral waves of electrical activity in the heart and the effect of various parameter values on the dynamics of these reentrant waves. The lecture topics included the current research in cardiac modeling, numerical methods, and tutorial on the compute unified device architecture (CUDA) programming. During the last week of the workshop, the in-depth lectures on GPU implementation of an electrophysiological model of a cardiac cell [5] was presented. The students successfully completed the workshop program and provided positive feedback. Some students continued to work with the researchers who conducted the workshop during the summer.

The interdisciplinary course "Finite Element Methods in Scientific Computing" [18] has been developed at the Texas A&M University. This project-based course covers parallel computing topics (pthreads, MPI), software engineering topics (using compilers, build tools, version control, debugging, *etc.*), and practical applications of the finite element method (FEM). The distinctive feature of the course is the usage of the flipped classroom format. The course authors recorded video lectures and posted them on YouTube [2]; therefore, students were able to watch the lectures outside the class. As a result, the teachers had more time in the class for interaction with students and for providing assistance to them on the projects. The flipped classroom format allows one to balance effectively competitive needs to present new material and to provide feedback to students.

At the University at Buffalo, an interdisciplinary computational and data-enabled science and engineering Ph.D. program has been developed [4]. The program included courses in three areas: applied mathematics and numerical

methods, high performance and data intensive computing, and data science. In addition, only students with master degree in any domain science, such as engineering, computer science, and applied mathematics, are eligible for this Ph.D. program. Deep knowledge of the domain science provides the context for training and allows them to quickly apply acquired knowledge for profitable use.

3 The Course Track "Heart Modeling and Simulation on Parallel Computing Systems"

3.1 General Course Track Description

To meet the need for interdisciplinary training in HPC, the course track "Heart Modeling and Simulation on Parallel Computing Systems" has been developed at the Institute of Mathematics and Computer Science of Ural Federal University. The course track brings together researches from a variety of discipline, not only from the Ural Federal University, but also from the Institute of Immunology and Physiology UrB RAS, and the Institute of Mathematics and Mechanics UrB RAS.

The course track includes three modules: parallel and distributed computing, numerical methods, and heart modeling. The complete list of the courses is presented in Table 1.

Table 1. The list of courses in the "Heart Modeling and Simulation on Parallel Computing Systems" track

No	Course name	ECTS credits
1	**Parallel and distributed computing module**	
1.1	Parallel and distributed computing	4
1.2	GPU programming	2
1.3	Xeon Phi programming	2
2	**Numerical methods module**	
2.1	Parallel numerical methods	2
2.2	Science hackathon	4
3	**Heart modeling module**	
3.1	Simulation of living systems	4
3.2	Modeling heart dynamics on parallel computing systems	2
4	**Optional prerequisite courses**	
4.1	Scientific computing in C	2
4.2	Software performance optimization	2

The courses are primarily designed for master students in computer science, but bachelor students and master students with other background are also eligible. Typical class consists of 12–15 students.

Table 2. Recommended sequence of study for master students in computer science

Semester	Course name
1	Parallel and distributed computing
	Simulation of living systems
2	GPU programming
	Modeling heart dynamics on parallel computing systems
	Parallel numerical methods
3	Xeon Phi programming
	Science Hackathon

The recommended sequence of study for Computer Science Master students is presented in Table 2. Most of courses in the track are elective. Students are not required to pass all the courses; they are able to choose ones according to their individual needs and preferences.

3.2 Prerequisite Courses

In addition to three main modules, the course track also includes the optional prerequisite courses: "Scientific computing in C" and "Software performance optimization". The courses are intended for students with non computer science related bachelor degree and limited software development experience. The aim of the module is to provide students with good skills for programming in C language, which are required for parallel computing.

The "Scientific computing in C" is a hands-on course on software development for science. It covers usage of compilers, build tools, mathematical libraries (BLAS, FFTW, MKL, *etc.*), Linux command line, debugging, testing, version control, and teamwork in software development.

The "Software performance optimization" course is devoted to sequential program optimization. The course contributes significantly to the track because parallelization of a poor performing sequential program is often useless and frequently leads to performance degradation instead of improvement. For many real-world applications, it is necessary to achieve maximum speedup for sequential program and only then proceed to parallelization. The course covers contemporary CPU architecture and key factors affecting CPU performance, tools for performance analysis, optimizing compilers, and SIMD vectorization. The course is presented in flipped format. We use the video lectures from the online course "Application optimization using Intel compilers" provided by Intel [1]. During the classroom sessions, students have programming assignments, do hands-on laboratory classes and small team projects.

3.3 Computational Resources

Two parallel computing systems are available for students when working on laboratory classes, doing home programming assignments, and projects. The first

system is educational HPC cluster of the Ural Federal University. The cluster has 12 nodes, 144 CPU cores, 12 Intel Xeon Phi 5110P accelerators, and 12 GPU NVIDIA Tesla K20X. The second system is the "URAN" supercomputer of the Institute of Mathematics and Mechanics UrB RAS. The supercomputer consists of 54 nodes with 648 CPU cores and 370 GPU NVIDIA Tesla.

4 Parallel and Distributed Computing Module

4.1 Parallel and Distributed Computing

The "Parallel and Distributed Computing" is the introductory HPC course. It consists of four parts: parallel computing theory, concurrency, parallel programming, and distributed computing using MapReduce. The course includes both lectures and laboratory classes on parallel computing systems. Students are required to hand in six home assignments.

The first part of the course covers architecture of parallel computing systems, theoretical limits to parallelization (Amdahl's and Gustafson's laws), approaches to parallel algorithms design, and popular patterns for parallel programming [12].

The second part of the course is devoted to concurrency. Students study multi-threaded programming for multicore and multiprocessor systems using C++11 concurrency, which now is a part of the standard C++ library. The C++ was used as a programing language for the first time in the spring of 2015. Earlier, the Java was used to study concurrency. During this part of the course, students also examine various pitfalls of concurrent and parallel programming: non-deterministic behavior, race conditions, deadlocks, livelocks, *etc.* Topics of home programming assignments for this part of the course are Dining Philosophers Problem and multi-threaded web crawler.

During the third part of the course, parallel programming using OpenMP and MPI is considered. The C was used as a programming language. Home programming assignments are k-means clustering using OpenMP and Conways Game of life using MPI.

The last part on the distributed computing has been recently added to the course. It is devoted to BigData, which is becoming popular nowadays. Students learn the MapReduce programming model, its implementation in the Apache Hadoop, and various tools from Hadoop ecosystem (Hive, Pig, Mahout, *etc.*). In addition, other popular frameworks for distributed data processing are considered, such as Apache Spark for fast distributed in-memory data processing and Apache Storm for streaming data processing. Home programming assignments for this part of course are analysis of Twitter graph using Hadoop and computing the term frequency-inverse document frequency (TF-IDFs) for Wikipedia articles using Spark.

4.2 GPU Programming

The "GPU Programming" is the advanced HPC course. The GPU offers to its users tremendous computational resources compared to CPU. However, efficient

usage of such computational power requires advanced skills from the developers. The course "GPU programming" was developed to meet this problem and to provide students with the basic principles of parallel software development on the CUDA platform.

The course begins with basics of GPU architecture and "hello world" on the CUDA platform. Then, students are introduced to the CUDA memory hierarchy, principles of building grids with blocks of threads. Other topics of specific GPU features and tips for its efficient usage to get maximum performance from underlying hardware are also studied. Another crucial topic within the course is multi-GPU programming using OpenMP and MPI. Students have to understand that even contemporary GPUs have limited memory resources in comparison with CPU RAM. Therefore, they should be able to get profit from using multiple GPUs for solving big real-world computational tasks that require large amount of memory. The course ends with such interesting and complicated topics like performance profiling, optimization, and using various libraries for scientific computing.

The course includes a series of short lectures and a considerable amount of practice with the CUDA software. Participants gain experience in parallel programming by completing GPU programming assignments, which include matrix multiplication, parallel reduction, and multi-GPU sorting.

4.3 Xeon Phi Programming

The second advanced HPC course is devoted to the Intel Xeon Phi programming, which is another popular type of computational accelerators. As well as GPU, the Xeon Phi provides large computational resources, which are difficult to use efficiently. However, Xeon Phi architecture and programming tools differ from GPU ones significantly.

The course covers Xeon Phi architecture, programming models (native, offload, and MPI), and software development tools and techniques. Special attention is paid to Xeon Phi SIMD vectorization capabilities as a crucial tool to achieve the best accelerator performance. Students learn about various approaches to use vectorization: auto-vectorization by compiler, Intel Cilk Plus (array notation and elemental functions), and OpenMP *simd* directive. The techniques for performance profiling, tuning, and optimization for the Xeon Phi are considered. In addition, students explore libraries with the Xeon Phi support, such as Intel MKL and MAGMA [7].

The course is organized in the same way as the "GPU programming" one. It includes 20–30 min lectures, hands-on laboratory classes, and programming assignments. The topics of home programming assignments are matrix multiplication, European option pricing, and molecular dynamics simulation.

5 Numerical Methods Module

5.1 Parallel Numerical Methods

The course "Parallel Numerical Methods" was developed to bridge the gap between courses on various parallel programming technologies (such as Pthreads, OpenMP, and MPI) and high level courses devoted to solving complex real-world scientific problems on parallel computing systems. The course helps students to understand which algorithmic building blocks they should use to build scalable HPC applications. Students should realize that introducing parallelism into some types of algorithms instead of overall performance improvement might lead to its degradation.

The course consists of two parts devoted to using parallel computing for solving compute-intensive and data-intensive tasks. Compute-intensive tasks are bounded by amount of computations and use regular memory access pattern. The examples of such tasks are various physical and mathematical simulations (such as heart simulation). While data-intensive tasks need relatively small number of computational operations and use irregular memory access pattern. Hence, such tasks are bounded by input-output operations. The example of data-intensive task is processing large amount of unstructured data, such as graph of social network.

During the first part of the course, students work on various parallel versions of numerical methods of linear algebra, numerical integration and differentiation, as well as numerical solving of mathematical physics equations. The techniques of effective parallelization of compute-intensive tasks, such as efficient cache usage, data partitioning among the processes, and reducing the communication between the processes are considered.

The second part of the course covers the data-intensive tasks. Special attention is paid to graph algorithms. The main obstacle to efficient parallel processing of large graphs is their irregular structure. Hence, sequential graph algorithms may often outperform their parallel versions. The course provides examples of various parallel graph algorithms, and useful techniques and methods to improve their performance.

The course consist of twelve lectures and three home programming assignments. The programming assignments require not only implementation of the parallel algorithms, but also carrying out the simple researches to investigate their performance and scalability. Such assignments are aimed at providing the students with experience of using various types of parallel algorithms and understanding how to choose the most efficient algorithms for solving real-world scientific challenges.

5.2 Science Hackathon

The "Science Hackaton" is the project-oriented hands-on course on scientific software development focused on HPC. The course is conducted on Saturdays and lasts all the day. During the day, a group of students works on a project

under the guidance of a tutor. Usually, the goal of the projects is to produce some usable software. However, some projects may have educational purpose, such as examining the capabilities of particular parallel library or exploring the parallel version of some numerical method.

The objective of the course is to provide students with opportunity to work together with the tutor for a long time. Hackathon participants are able to quickly gain an experience of successfully resolving the real-world scientific problems due to regular guidance from the tutor. Students obtain not only technical knowledge, but also experience in team software development, problem solving skills, and professional confidence.

The topics for Hackathon projects are chosen by students. The course primarily aimed at topics from the heart modeling track, but students are free to suggest other themes for projects. The examples of the spring 2015 projects are working with computational cluster in command line, using auto-vectorization for Xeon Phi, evaluating Intel VML performance on Xeon Phi, and simulating the muscle cube on parallel computing system using FEM and mass-spring method.

6 Heart Modeling Module

6.1 Simulation of Living Systems

The "Simulation of living systems" is the interdisciplinary theoretical course aimed at introducing students to the core concepts of biophysics, biomedical engineering, and biology, which use methods of mathematical modeling and bioinformatics [17]. Students study several classical mathematical models of biological process based on nonlinear theory of dynamic systems. These models represent the characteristic features of biological process and demonstrate the effectiveness of application of mathematical models application to understanding the mechanisms of biological systems.

The course participants become familiar with a number of biological processes (such as transport of substances, chemical kinetics, types of interaction in biological systems), mathematical concepts for describing these processes, a variety of techniques for modeling complex biological systems, and methods of models evaluation. Applications of various numerical schemes to living system simulation are considered.

During the course, the heart modeling is discussed in details. The main function of the heart is to pump blood throughout the body using contractions. Mechanical contraction of the heart is triggered by electrical activation of the myocardium. This process is known as excitation-contraction coupling. In the turn, the mechanical deformation influences the cardiac electrical activity. This process is designated as cardiac mechano-electric feedback. Hereby, the functional heart model consists of three main components: an anatomical model, an electrophysiological model, and a mechanical one. Students study various heart models [13] and the methods of integration of these models into one multiscale model of the heart.

An important advantage of the course is in examining the state-of-the-art heart models developed at the Institute of Immunology and Physiology UrB RAS. Such models include Ekaterinburg-Oxford electromechanical cell model [9,15,16] and mathematical model of the anatomy and fibre orientation field of the left ventricle (LV) of the heart [14].

The course consists of twelve lectures and twelve seminars. The prerequisite courses are applied mathematics and numerical methods. No biology, chemistry, or medicine courses are required.

6.2 Modeling Heart Dynamics on Parallel Computing Systems

The main objective of the course is to provide the students with the hands-on introduction to the contemporary methods of mathematical modeling of complex (multi-level) living systems. As an example of such task, the simulation of electromechanical function of the cardiac LV of the mammal is used during the course.

Application of FEM to multi-level (cell-tissue-organ) modeling of electromechanical contraction of the heart LV is considered. Currently, the FEM is the most widely used method of computer simulation of compound systems with complex geometry.

The course consists of eight lectures, ten workshops, and home programming assignments. During the lectures, the common approaches to the computer modeling of living systems are described. As an example of the heart simulations, the general sequence of computer simulation is demonstrated. Then the structure and physiology of the mammal heart, particularly the LV, are briefly described. Various methods and models of digital representation of the LV by magnetic resonance imaging and ultrasound are considered. The foundation of FEM meshing, meshing quality criteria, and methods of anisotropy vector field construction are presented. After that, FEM simulation of the electrodiffusion and mechanical contraction of the heart LV is examined. Theoretical part of the course ends with discussion of various tools for visualizations of simulation results.

The amount of time allotted for lectures certainly is not enough to present the above-mentioned topics thoroughly, but it is not necessary. The course is aimed at rapid transition from the basic theoretical knowledge to its practical usage.

During the workshops, students build the model of LV using the tetrahedral 3D-network with the help of the “GMsh” library [8]. After that, students utilize FEM implementation from the “FENICS” library [11] to simulate muscle cube on the MPI clusters. Various techniques of optimization by the FEM application on parallel computer systems are examined. To visualize the simulation results, students use the “Paraview” software.

7 Discussion

Our experience demonstrates that teaching HPC in the interdisciplinary education program is very effective. Knowledge of numerical methods and the heart

modeling provides the students with the ability to solve nontrivial tasks on parallel computing systems. In turn, comprehensive programming assignments force students to explore the most efficient way of parallel computing technologies usage. For example, the course "Parallel numerical methods" includes two different assignments: numerical integration and single source shortest paths (SSSP) graph problem. To parallelize numerical integration students can use simple "master-slave" or "point-to-point" communication schemes. In both cases, they will achieve good speedup. At that point, many of students may think that parallelization is quite an "easy" research topic: to get performance improvement some MPI_Send and MPI_Recv functions must be inserted in the code to the appropriate places. Situation changes when students encounter the SSSP problem. Firstly students try to use well-known for them "point-to-point" parallelization, but achieve rather small (or even no) speedup due to irregular memory access pattern of the SSSP problem. Efficient solution requires usage of MPI collective communications function MPI_Allgather. Although the MPI_Allgather is heavyweight function and has big overhead, in this particular case of the SSSP problem, the MPI_Allgather allows balancing communications both across the various communication processes within single iteration and across all iterations of the entire SSSP algorithm. Thus, the MPI_Allgather function usage make possible performance speedup and scaling of SSSP graph problem.

Interdisciplinary approach also helps to increase students' motivation. It is very interesting for students to use parallel computing system not only for simple programming assignments, such as π calculation or matrix multiplication, but also for the real-world problems of heart simulation, which are very intensive in computations. Students willingly work on such tasks and try to squeeze last drop of performance from the hardware to speed up the simulation.

8 Conclusion

The interdisciplinary track of coherent courses on using HPC for the heart simulation is presented. The track consists of three modules: parallel computing technologies, heart modeling and simulation, and parallel numerical methods. Now, the track is conducted at the Ural Federal University.

The interdisciplinary approach allows students to gain deep knowledge in HPC by working on nontrivial programming assignments and participation in complex heart simulation projects. In addition, such approach increased students' motivation.

Several students decided to continue working on the heart modeling and obtained research positions at the laboratory "Modeling of living systems on supercomputers". Hence, the goal of involving students in research is achieved.

In the future, we plan to increase application of video lectures and flipped classroom format in hands-on courses. Another important direction of future works is designing interdisciplinary educational tracks for other domains, such as data science.

Acknowledgments. The work is supported by the Programme of Presidium of RAS no. II.4P (PI O.Solovyova). Our study was performed using the "Uran" supercomputer from Institute of Mathematics and Mechanics UrB RAS.

References

1. Anuferenko, A., Idrisov, R., Kasyanov, V., Vladimirovich, N.: Intel academy. Application optimization using intel compilers. http://www.intuit.ru/studies/courses/660/516/info
2. Bangerth, W.: 48 video lectures on computational science (2013). http://www.math.tamu.edu/~bangerth/videos.html
3. Bartocci, E., Singh, R., von Stein, F.B., Amedome, A., Caceres, A.J.J., Castillo, J., Closser, E., Deards, G., Goltsev, A., Ines, R.S., Isbilir, C., Marc, J.K., Moore, D., Pardi, D., Sadhu, S., Sanchez, S., Sharma, P., Singh, A., Rogers, J., Wolinetz, A., Grosso-Applewhite, T., Zhao, K., Filipski, A.B., Gilmour, Jr., R.F., Grosu, R., Glimm, J., Smolka, S.A., Cherry, E.M., Clarke, E.M., Griffeth, N., Fenton, F.H.: Teaching cardiac electrophysiology modeling to undergraduate students: laboratory exercises and gpu programming for the study of arrhythmias and spiral wave dynamics. Adv. Physiol. Educ. **35**(4), 427–437 (2011). http://dx.doi.org/10.1152/advan.00034.2011
4. Bauman, P.T., Chandola, V., Patra, A., Jones, M.: Development of a computational and data-enabled science and engineering Ph.d. program. In: Proceedings of the Workshop on Education for High-Performance Computing, EduHPC 2014, pp. 21–26. IEEE Press, Piscataway (2014). http://dx.doi.org/10.1109/EduHPC.2014.8
5. Bueno-Orovioa, A., Cherryb, E.M., Fenton, F.H.: Minimal model for human ventricular action potentials in tissue. J. Theoret. Biol. **263**(3), 544–560 (2008). http://dx.doi.org/10.1016/j.jtbi.2008.03.029
6. Crampin, E.J., Halstead, M., Hunter, P., Nielsen, P., Noble, D., Smith, N., Tawhai, M.: Computational physiology and the physiome project. Exp. Physiol. **89**(1), 1–26 (2004). http://dx.doi.org/10.1113/expphysiol.2003.026740
7. Dongarra, J., Gates, M., Haidar, A., Jia, Y., Kabir, K., Luszczek, P., Tomov, S.: Portable HPC programming on intel many-integrated-core hardware with MAGMA port to Xeon Phi. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2013, Part I. LNCS, vol. 8384, pp. 571–581. Springer, Heidelberg (2014)
8. Geuzaine, C., Remacle, J.F.: GMSH: a 3-d finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Meth. Eng. **79**(11), 1309–1331 (2009). http://dx.doi.org/10.1002/nme.2579
9. Katsnelson, L.B., Vikulova, N.A., Kursanov, A.G., Solovyova, O.E., Markhasin, V.S.: Electro-mechanical coupling in a one-dimensional model of heart muscle fiber. Russ. J. Numer. Anal. Math. Modell. **29**(5), 275–284 (2014)
10. Kohl, P., Noble, D.: Systems biology and the virtual physiological human. Mol. Syst. Biol. **5**(1), 292 (2009). http://dx.doi.org/10.1038/msb.2009.51
11. Logg, A., Mardal, K.A., Wells, G.N.: Automated Solution of Differential Equations by the Finite Element Method. Lecture Notes in Computational Science and Engineering, vol. 84. Springer, Heidelberg (2012)
12. McCool, M., Reinders, J., Robison, A.: Structured Parallel Programming: Patterns for Efficient Computation. Elsevier, Waltham (2012)

13. Pfeiffer, E.R., Tangney, J.R., Omens, J.H., McCulloch, A.D.: Biomechanics of cardiac electromechanical coupling and mechanoelectric feedback. J. Biomech. Eng. **136**(2), 021007 (2014)
14. Pravdin, S., Berdyshev, V., Panfilov, A., Katsnelson, L., Solovyova, O., Markhasin, V.: Mathematical model of the anatomy and fibre orientation field of the left ventricle of the heart. BioMed. Eng. Online **12**(1), 54 (2013). http://www.biomedical-engineering-online.com/content/12/1/54
15. Pravdin, S.F., Dierckx, H., Katsnelson, L.B., Solovyova, O., Markhasin, V.S., Panfilov, A.V.: Electrical wave propagation in an anisotropic model of the left ventricle based on analytical description of cardiac architecture. PLoS ONE **9**(5), e93617 (2014). http://dx.doi.org/10.1371/journal.pone.0093617
16. Solovyova, O., Katsnelson, L., Konovalov, P., Kursanov, A., Vikulova, N., Kohl, P., Markhasin, V.: The cardiac muscle duplex as a method to study myocardial heterogeneity. Prog. Biophys. Mol. Biol. **115**(23), 115–128 (2014). http://www.sciencedirect.com/science/article/pii/S007961071400073X. Novel Technologies as Drivers of Progress in Cardiac Biophysics
17. Solovyova, O.E., Markhasin, V.S., Katsnelson, L.B., Sulman, T.B., Vasilyeva, A.D., Kursanov, A.G.: Mathematical Modeling of Living Systems. Urals University Press, Ekaterinburg (2013)
18. Zarestky, J., Bangerth, W.: Teaching high performance computing: Lessons from a flipped classroom, project-based course on finite element methods. In: Proceedings of the Workshop on Education for High-Performance Computing, EduHPC 2014, pp. 34–41. IEEE Press, Piscataway (2014). http://dx.doi.org/10.1109/EduHPC.2014.10